International Journal of Digital Multimedia Broadcasting

# Advances in 3DTV: Theory and Practice

Guest Editors: Xenophon Zabulis, Irene Cheng, Nikolaos Grammalidis, Georgios A. Triantafyllidis, and Pietro Zanuttigh

# Advances in 3DTV: Theory and Practice

International Journal of Digital Multimedia Broadcasting

# Advances in 3DTV: Theory and Practice

Guest Editors: Xenophon Zabulis, Irene Cheng, Nikolaos Grammalidis, Georgios A. Triantafyllidis, and Pietro Zanuttigh

# Contents

Hindawi Publishing Corporation
International Journal of Digital Multimedia Broadcasting
Volume 2010, Article ID 579514, 2 pages
doi:10.1155/2010/579514

*Editorial*

# Advances in 3DTV: Theory and Practice

**Xenophon Zabulis,[1] Irene Cheng,[2] Nikolaos Grammalidis,[3] Georgios A. Triantafyllidis,[3] and Pietro Zanuttigh[4]**

[1] *ICS-FORTH, Greece*
[2] *University of Alberta, Canada*
[3] *ITI-CERTH, Greece*
[4] *University of Padova, Italy*

Correspondence should be addressed to Xenophon Zabulis, zabulis@ics.forth.gr

Received 21 March 2010; Accepted 21 March 2010



Being a goal over decades, the extension of television, and visual communications in general, to the third dimension (3DTV) has been almost reached. Currently, 3D motion pictures are shown in theatres and the first 3D broadcasts are being planned to initiate in the next few years. Albeit the progress, state of the art has not yet reached the goal of acquiring a three-dimensional scene in full detail and creating a precise optical duplicate at remote site in real-time. Limitations in reconstruction accuracy and visual quality as well as user acceptance of pertinent technologies have, to date, prevented the creation of infrastructures for the delivery of 3D content to mass markets. Thereby, relevant scientific research is at the center of interest regarding the improvement of current 3DTV technologies.

The goal of 3DTV involves the collaboration of researchers in multiple scientific disciplines as well as the integration of pertinent results, in order to provide end-users with a fully functional system. The tasks of "capture and representation of 3D scene information," "representation of the digital 3DTV signal," "storage and transmission of the digital 3DTV signal," as well as "display of the reproduced 3D scene" constitute a chain of operations that are essential for reaching this goal. In this context, this special issue presents contributions along the whole chain of computational tasks that are required to implement 3DTV. In addition, the last two papers provide with contributions that go beyond the simple presentation of 3D visual data and propose ways of interacting with them.

The paper "Multi-view shooting geometry for multiscopic rendering with controlled distortion" introduces a complete analysis of the geometrical quality of 3D content. The article compares the perceived depth with scene depth obtained from the viewing and shooting (acquisition) geometries, yielding a depth distortion model whose parameters are expressed by the geometrical characteristics of shooting and rendering devices. These expressions are then inverted, in order to design convenient shooting layouts yielding chosen distortions on specific rendering devices.

The paper "Near real time estimation of super-resolved depth and all-in-focus images from a plenoptic camera using graphic processing units (GPU)," provides a near real-time method for the employment of plenoptic video-cameras in the acquisition of all-in-focus color images with accompanying depth values. The proposed method overcomes low-resolution deficiencies of previous methods, while its output can be directly provided as input to autostereoscopic displays.

Occlusions are a characteristic problem in disparity estimation, in binocular and multiview stereo. The paper "An occlusion approach with consistency constraint for multiscopic depth extraction" addresses this issue, handling occlusions for improved disparity estimation from multiple images acquired by a parallel camera geometry. The proposed technique ensures the consistency of disparity maps and its application demonstrated on

both a correlation-based and a graph-cut based stereo approach.

The need for fast and compact representations for 3D animations raises two issues: (1) how to classify portions of an object into moving and stationary and (2) how to represent nonrigid motion for moving portions. In this context, the paper "A compact representation for 3D animation using octrees and affine transformations" deals with a new and compact 3D representation for nonrigid objects using motion vectors between consecutive frames. The proposed method employs an affine-based motion representation that uses a rigid octree structure for partitioning.

Multiple Description Coding is one of the most promising approaches for the transmission of video signals over unreliable networks. The paper "A scalable multiple description scheme for 3D video coding based on the inter layer prediction structure" presents a coding scheme that is based on H.264/SVC that enables the application Multiple Description Coding to 3D video. The proposed method is applicable on both stereo sequences and depth-image-based rendering schemes.

The application of digital holography as a viable solution to 3D capture and display technology is examined in the paper "Digital holographic capture and opto-electronic reconstruction for 3D displays." The proposed work addresses major challenges in digital holography including (1) the removal of the DC and conjugate image terms, which are features of the holographic recording process, (2) the reduction of speckle noise, a characteristic of a coherent imaging process, (3) increasing the spatial resolution of digital holograms, and (4) replaying captured and/or processed digital holograms using spatial light modulators.

The transmission of 3D scenes over unreliable networks is a challenging topic in digital broadcasting research. In order to maximize the rendered quality, the article entitled "Transmission of 3D scenes over lossy channels" introduces a novel Unequal Error Protection scheme to transmit texture and depth information. Various scene elements are assigned different predefined amounts of redundancy based on their visual impacts. Experimental results demonstrate the effectiveness of the proposed scheme.

A complete multicamera real-time 3D modeling system, Grimage, is introduced in "Multi-camera real-time 3D modeling for telepresence and remote collaboration," which is employed in the creation of new immersive and interactive environments. The system creates in real-time a textured 3D model of the observed scene and utilized this information to compute collisions and reaction forces with virtual objects, enforcing the mechanical presence of the user in the virtual world.

The paper "Stereo 3D mouse cursor: A method for interaction with 3D objects in a stereoscopic virtual 3D space" presents a novel approach that applies stereoscopic principles to the classical mouse cursor in order to build a 3D pointing device that allows to effectively navigate within 3D scenes. The outcome is a novel way to interact with the reconstructed 3D that matches the requirements of a fully immersive experience.

With research and development on 3D imaging and visual reproduction systems being more active than ever in the past, pertinent technologies are expected to be increasingly employed in the broader context of communication of 3D visual information.

*Xenophon Zabulis*
*Irene Cheng*
*Nikolaos Grammalidis*
*Georgios A. Triantafyllidis*
*Pietro Zanuttigh*

Hindawi Publishing Corporation
International Journal of Digital Multimedia Broadcasting
Volume 2010, Article ID 975674, 11 pages
doi:10.1155/2010/975674

*Research Article*

# Multiview Shooting Geometry for Multiscopic Rendering with Controlled Distortion

**J. Prévoteau,[1,2] S. Chalençon-Piotin,[1] D. Debons,[2] L. Lucas,[1,2] and Y. Remion[1,2]**

[1] *CReSTIC-SIC EA3804, Université de Reims Champagne-Ardenne, IUT de Reims-Châlons-Charleville, rue des Crayères, BP 1035, 51687 Cedex 2, France*
[2] *TéléRelief, Pépinière d'Entreprises Henri Farman, 2 Allée Albert Caquot, 51100, France*

Correspondence should be addressed to J. Prévoteau, jessica.prevoteau@etudiant.univ-reims.fr

Received 30 April 2009; Revised 4 October 2009; Accepted 8 December 2009

Academic Editor: Georgios Triantafyllidis



A fundamental element of stereoscopic and/or autostereoscopic image production is the geometrical analysis of the shooting and viewing conditions in order to obtain a qualitative 3D perception experience. This paper firstly compares the perceived depth with the shot scene depth from the viewing and shooting geometries for a couple of shooting and rendering devices. This yields a depth distortion model whose parameters are expressed from the geometrical characteristics of shooting and rendering devices. Secondly, these expressions are inverted in order to design convenient shooting layouts yielding chosen distortions on specific rendering devices. Thirdly, this design scheme provides three shooting technologies (3D computer graphics software, photo rail, and camera box system) producing qualitative 3D content for various kinds of scenes (real or virtual, still or animated), complying with any prechosen distortion when rendered on any specific multiscopic technology or device formerly specified.

## 1. Introduction

The three-dimensional display is on its way to becoming the next evolution in the image industry. These 3D displays hold tremendous potential for many applications in entertainment, information presentation, reconnaissance, telepresence, medicine, visualization, remote manipulation, and art. Both research and business in multiscopic display are increasing. We dispose now of numerous multiscopic rendering systems with or without glasses. Different technologies support all these systems; stereoscopy with colorimetric or temporal mixing such as anaglyph [1, 2], occultation and polarization [3], for example for projections with glasses as in some movie theaters; autostereoscopy [4, 5] such as parallax barrier and lenticular lens; for example, for 3D advertising billboards, autostereoscopic displays or lenticular printing.

As shown in Figure 1, the different rendering modes and kinds of scenes to be shot are well-known, but all these systems need content; up to now, there is no 3D shooting system specifically designed to acquire a qualitative 3D content. Some works [6–9] consist in calculating a right eye image from a left eye image and so obtain a 3D content from a 2D-plus-depth. The main disadvantage of these methods lies in the lack of information in occluded areas which is impossible to overcome in a generic way. Yet, to comply with our demand of qualitative 3D content we focus on multi-shooting technologies. Other works [10, 11] define the projective relations between the images shot by multicameras in order to calibrate the different cameras and then to reconstruct the 3D shot scene from these multiple views. There is no link with any viewing device since the target is a reconstruction module. In our case, flat multiscopic viewing requires a simplified shooting layout also called "rectified geometry".

Moreover, some works have been done to improve the control of the viewer's 3D experience in stereoscopy and computer graphics fields [12, 13]. They usually compare shooting and viewing geometries in order to choose a shooting layout fitting a given depth range in virtual space to the "comfortable" depth range of the display. We believe that choices that can be made in the shooting design may be richer than a simple mapping of depth and could differ for each observation position in the multi-view case. This

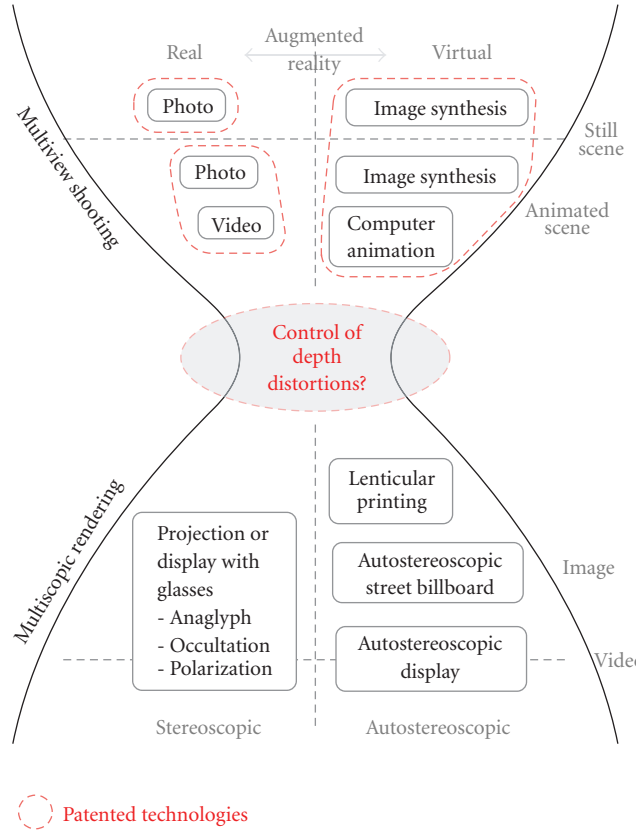



Figure 1: Shooting to rendering process.

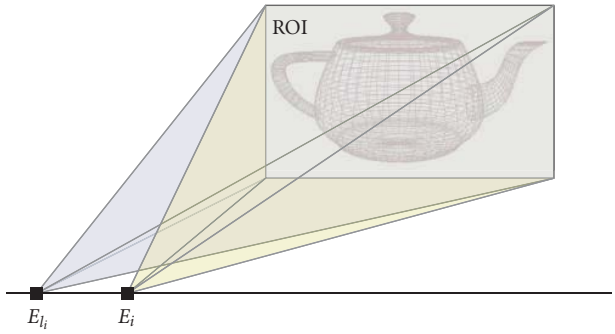
requires a detailed model and a precise analysis of possible distortions for multiscopic shooting/viewing couple. Indeed, such a model will provide the characteristics of shooting which will ensure the chosen distortions on the chosen viewing device. If some authors have described the transformation between the shot and the real scene [12] in the stereoscopic case, none of them has been interested in producing an analytic multi-observer and reversible model allowing to pilot the shooting for all kind of possible distortions. Thus, we suggest a solution to produce 3D content according to the chosen rendering mode and the desired depth effect.

So we will explain how to model and quantify the depth distortion from given rendering and shooting geometries and also from a chosen rendering device and a desired depth effect, how to design the appropriate shooting layout.

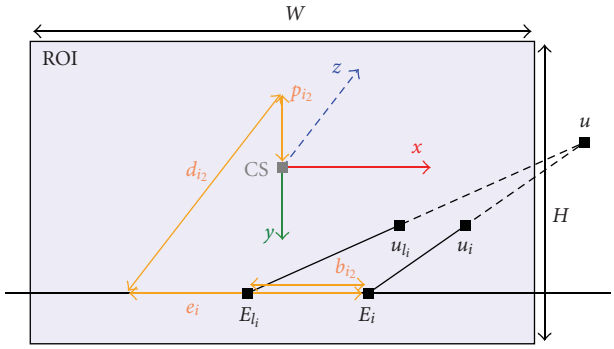
This article introduces an complete analysis of the geometrical quality of 3D content based on distortion analysis by linking shooting and viewing geometries. Starting from the prior and related knowledge (i.e., viewing and shooting geometries), we will show remaining problems and model the possibilities of depth distortions between the scene perceived by a viewer and the initially shot scene. Next, we will present a shooting layout design scheme ensuring a desired depth effect (controlled depth distortion or perfect depth effect) upon a pre-determined rendering device. Finally, we will introduce three shooting technologies (which are patent pending) complying to this scheme and, thus, achieving qualitative 3D content on formerly given rendering device: 3D computer graphics software, photo rail and camera box systems. We will present these prototypes and some of their results.

## 2. Prior and Related Knowledge

*2.1. Viewing.* 3D image rendering, with or without glasses, is known to require "stereoscopic" or "autostereoscopic" devices. All these devices make a spatial, colorimetric and/or temporal mixing over a single region of interest (ROI area physically filled by the displayed image on the rendering device) of $n \times m$ so-called "initial images" of one scene shot from several distinct viewpoints. These systems allow to optically and/or temporally separate the images reaching each eye of one or more viewers. In case of stereoscopic systems, both images are emitted in a single optical beam independently of the viewer's position in this beam [1, 2, 14]. However, autostereoscopic systems separate the images in several distinct optical beams, organized for example, in horizontal "range" of $n$ images ($n \geq 2$ and $m = 1$) [4, 5]. We can also imagine optical beams organized in both horizontal and vertical ranges. Then we dispose of a matrix disposition of $n \times m$ optical beams ($n \geq 2$ and $m \geq 2$), each one transporting a different image. Thus, all known devices broadcast alternately and/or simultaneously $n \times m$ images ($n \geq 2$ and $m \geq 1$) within one or several optical beams in such a way that both eyes of a correctly-placed viewer

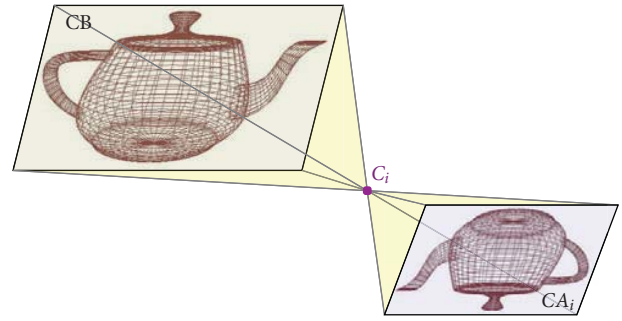



(a) Generic description of preferred position for multiscopic rendering device

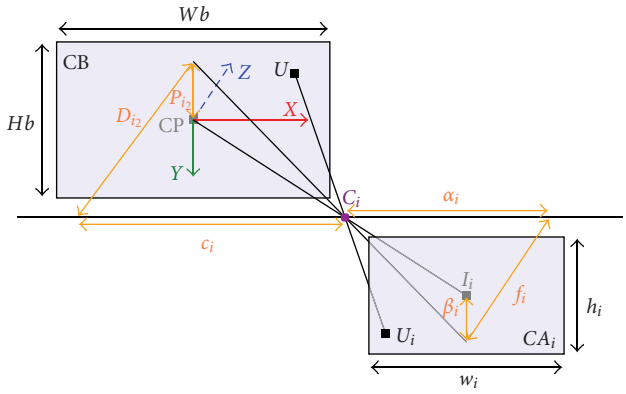



(b) Detailed description of the depth perception on multiscopic rendering device

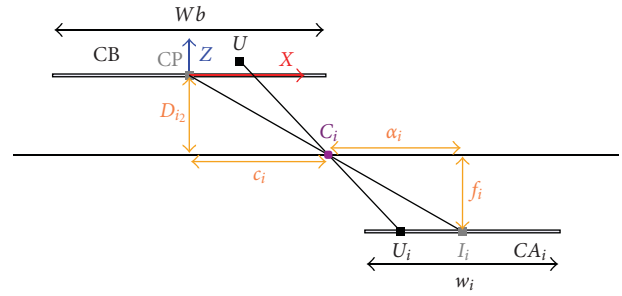
Figure 2: The viewing geometry.



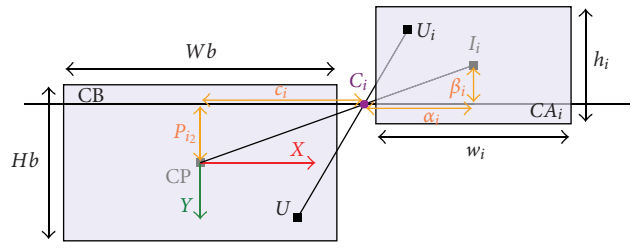

(a) Generic description of a non-convergent parallel camera set-up




(b) Detailed perspective view




(c) Detailed top view




(d) Detailed full-face view

Figure 3: Implied shooting geometry.

get different consistent images (i.e., initial images and not combinations of them). Thereby the viewer's brain rebuilds his depth perception by stereoscopy [15]. Even if the human visual system has a tolerance as for epipolar alignment, ideal positions within this tolerance correspond in particular to eyes line parallel to the display's rows. Despite this human tolerance, we calculate our images in such a way that they have a perfect epipolar alignment for well-placed eyes line.

So let's analyse the geometry of these devices (the "viewing geometry" (Figure 2) which will constrain the compatible shooting layout.

A 3D rendering device mixes $n \times m$ images sharing out a ROI of dimension $W$ (width) and $H$ (height). Each image (image's index $i = (i_1, i_2) \in \{1, n\} \times \{1, m\}$) is supposed to be "correctly" visible (without much mixing with others) at least from the chosen preferential position $E_i$. These positions are aligned upon $m$ lines parallel to the rows of the ROI located at distance $d_{i_2}$, $i_2 \in \{1, \ldots m\}$, from the device ROI. The preferential positions $E_i$ are placed on those lines according to their second index $i_2$ in order to guarantee that a viewer whose binocular gap is $b_{i_2}$ (often identical to human medium binocular gap 65 mm, but possibly different according to the expected public: children, etc.), with eyes line parallel to the device rows, would have his right eye in $E_i$ and his left eye in $E_{l_i}$. The right eye in $E_i$ would catch image number $i$, while the left eye in $E_{l_i}$ would catch image number $l_i$ knowing that $l_i = i - (q_{i2}, 0)$ with $q_{i_2}$ being the way gap between the indexes of images composing the visible consistent stereoscopic couples with binocular gap $b_{i_2}$ at distance $d_{i_2}$. Hence, associated left and right eye preferential positions $E_{l_i}$ and $E_i$ verify $E_i = E_{l_i} + b_{i_2}x$ and $e_i = e_{l_i} + b_{i_2}$.

We also define the lines positions vertically (because viewers of various sizes use the device) by $p_{i_2}$ which represents the "overhang", that is, vertical gap of eyes positioning compared with the ROI center $CS$. If we do not know

$p_{i_2}$, we use a medium overhang corresponding to a viewer of medium size, which has to be chosen at design stage. Assuming $u_i$ and $u_{l_i}$ are stereoscopic homologous for images $i$ and $l_i$, their perception by the right and left eye of a viewer from $E_i$ and $E_{l_i}$ leads this spectator's brain to perceive a 3D point $u$. The viewing geometry analysis is expressed thanks to a global reference frame $r = (CS, x, y, z4x \times y)$, chosen at the ROI center $CS$, with $x$ parallel to its rows and turned towards the right of the spectators, and $y$ parallel to its columns and turned towards the bottom.

*2.2. Shooting.* In order to "feed" such devices with 3D content, we need sets of $n \times m$ images from a single scene acquired from several distinct and judicious viewpoints and with specific projective geometry as the rendering upon flat multiscopic devices involves coplanar mixing of these images. This major issue is well known in multiscopy.

The image viewing is achieved according to distorted pyramids whose common base corresponds to the device ROI and the tops are the viewer's eyes or $E_i$ positions. Given that vision axes are not necessarily orthogonal to the observed images area (ROI), the viewing of these images induces trapezoidal distortion if we don't take into account this slanted viewing during the shooting. This has an immediate consequence in order to achieve depth perception. If the trapezoidal distortions are not similar for the two images seen by a spectator, the stereoscopic matching by the brain is more delicate, or even impossible. This reduces or cancels the depth perception. This constraint, well-known in stereoscopy is called the "epipolar constraint".

Solutions (also called toe-in camera model) of convergent systems have been proposed [16, 17], but such convergent devices manifest the constraint presented above. So, unless a systematic trapezoidal correction of images is performed beforehand (which might not be desirable as it loads down the processing line and produces a qualitative deterioration of the images) such devices do not afford to produce a qualitative 3D content. As demonstrated by [18, 19], we must use devices with shooting pyramids sharing a common rectangular base (off-axis camera model) and with tops arranged on a line parallel to the rows of this common base in the scene. For example Dodgson et al. use this shooting layout for their time-multiplexed autostereoscopic camera system [20].

Thus, aiming axes are necessarily convergent at the center of the common base and the tops of the shooting pyramids must lie on $m$ lines parallel to the rows of the common base. Figure 3(b) shows a perspective representation of such a shooting geometry. This figure defines the layout of the capture areas ($CA_i$), and the centers ($C_i$) and specifies a set of parameters describing the whole shooting geometry completely. Figures 3(c) and 3(d) show top and full-face representations of this geometry, respectively.

The shooting geometry analysis is expressed using a shooting global reference frame $R = (CP, X, Y, Z \equiv X \times Y)$ chosen centered at the desired convergence point $CP$ (which is also the center of the common base $CB$ of the scene) and oriented in such a way that the two first vectors of the reference frame are parallel to the main directions of the common base $CB$ of the scene and so, parallel to the main directions of the capture areas. The physical size of $CB$ is $Wb$ and $Hb$. Furthermore, the first axis is supposed to be parallel to the rows of the capture areas and the second axis is supposed to be parallel to the columns of these areas.

The $n \times m$ pyramids, representative of a shooting layout, according to the principles previously explained to resolve the known issue, are specified by

- an optical axis of $Z$ direction,
- optical centers $C_i$ (i.e.,: principal points) aligned on one or more ($m$) line(s) parallel to the rows of the common base (so on $X$ direction),
- rectangular capture areas $CA_i$.

These capture areas must be orthogonal to $Z$, so parallel one to another and parallel to $CB$ and to centers lines (which are defined by their distances from $CB$, $D_{i_2}$ along $Z$, $P_{i_2}$ along $Y$ and $c_i$ along $X$). These capture areas are also placed at distances $f_i$ along $Z$, $\beta_i$ along $Y$ and $\alpha_i$ along $X$ from their respective optical center $C_i$. Their physical size is given by $w_i$ and $h_i$. They are centered on points $I_i$ in such a way that lines $I_iC_i$ defining the axes of sight are convergent at $CP$. The centers $C_i$ and $C_{l_i}$ must be on same "centers line" and with a spacing of $B_i$ ($C_i = C_{l_i} + B_iX$ and $c_i = c_{l_i} + B_i$).

Such a shooting layout is necessary to obtain a depth effect on multiscopic device. Nevertheless, its does not ensure that the perceived scene will not be distorted relative to the shot scene. Non distortion implies that viewing pyramids are perfect counterpart of shooting pyramids (i.e., have exactly same opening and main axis deviation angles in both horizontal and vertical directions). In case of pyramids dissimilarity, the 3D image corresponds to a complex distortion of the initially acquired scene. This can be desirable in some applications to carry out some special effects, as it can be undesirable in others. This requires, that shooting and viewing must be designed in a consistent way whether we desire a depth distortion or not. Let's now model those distortion effects implied by a couple of shooting and viewing geometries.

## 3. Distortion Analysis and Model

In this section, we consider that we use perfect sensors and lenses, without any distortion. This assumption implies some issues which will be presented for each derived technology.

Thanks to the two previous sections we can link the coordinates $(X, Y, Z)$, in the reference frame $R$, of the point $U$ of the scene, shot by the sensors previously defined, with the coordinates $(x_i, y_i, z_i)$ in the reference frame $r$, of its counterparts $u$ seen by an observer of the viewing device placed in an preferential position.

Assuming that the scene point $U$ is visible on image number $i$, its projection $U_i$ verifies

$$C_iU_i = \frac{-f_i}{Z + D_{i_2}} C_iU \quad i \in \{1, n\} \times \{1, m\}. \tag{1}$$

Knowing that $I_i$, center of $CA_i$, verifies

$$C_iI_i = \frac{-f_i}{D_{i_2}} C_iCP \quad i \in \{1,n\} \times \{1,m\}, \tag{2}$$

the relative position of the scene point $U$'s projections in the various images are expressed as

$$I_iU_i = \frac{f_i}{Z + D_{i_2}} \begin{bmatrix} -X - Z\dfrac{c_i}{D_{i_2}} \\ -Y + Z\dfrac{P_{i_2}}{D_{i_2}} \\ 0 \end{bmatrix}_R \quad i \in \{1,n\} \times \{1,m\}. \tag{3}$$

As the images are captured behind the optical centers, the projection reverses up/down and left/right axes, and the implicit axes of the images are opposites of those of the global shooting reference frame $R$. Moreover, the images are then scaled on the whole ROI of the rendering device. This relates $U_i$ projections of $U$ to their "rendered positions" $u_i$ on the ROI:

$$CSu_{i|_r} = -\begin{bmatrix} \dfrac{W}{w_i} & & \\ & \dfrac{H}{h_i} & \\ & & 1 \end{bmatrix} I_iu_{i|_R} \quad \forall i. \tag{4}$$

Remarking that $f_iWb = D_{i_2}w_i$ and $f_iHb = D_{i_2}h_i$, $u_i$ is expressed in the reference frame $r$ as

$$u_{i|_r} = \frac{D_{i_2}}{Z + D_{i_2}} \begin{bmatrix} \left(X + Z\dfrac{c_i}{D_{i_2}}\right)\dfrac{W}{Wb} \\ \left(Y - Z\dfrac{P_{i_2}}{D_{i_2}}\right)\dfrac{H}{Hb} \\ 0 \end{bmatrix} \quad \forall i. \tag{5}$$

By this time, and assuming $U$ was visible on both images $l_i$ and $i$, we notice that $u_{l_i}$ and $u_i$ lie on the same row of the ROI. This fulfills the epipolar constraint and thus permits stereoscopic reconstruction of $u = [x_i, y_i, z_i]_r^t$ from $E_{l_i}$ and $E_i$ according to

$$\begin{aligned} u_{l_i}u_i &= \frac{z_i}{z_i + d_{i_2}} b_{i_2}x, \quad \text{which yields } z_i \text{ and} \\ E_iu &= \frac{z_i + d_{i_2}}{d_{i_2}} E_iu_i, \quad \text{which then gives } x_i, y_i. \end{aligned} \tag{6}$$

Thus, after some calculus, the relation between the 3D coordinates of the scene points and those of their images perceived by a viewer may be characterized under homogeneous coordinates by

$$a_i \begin{bmatrix} x_i \\ y_i \\ z_i \\ 1 \end{bmatrix} = \left[\begin{array}{ccc|c} & \mu_i & \gamma_i & 0 \\ k_{i_2} & \rho\mu_i & \delta_i & 0 \\ & & 1 & 0 \\ \hline 0 & 0 & \dfrac{k_{i_2}(\epsilon_i - 1)}{d_{i_2}} & \epsilon_i \end{array}\right] * \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}. \tag{7}$$

The above equation can be seen as the analytic distortion model for observer position $i$ which matches the stereoscopic transformation matrix given in [12]. As such this model clearly exhibits the whole set of distortions to be expected in any multiscopic 3D experience, whatever the number of views implied or the very nature of these images (real or virtual). It shows too that these distortions are somehow independent one from another and may vary for each observer position $i$. The following detailed analysis of this model and its further inversion will offer a novel multiscopic shooting layout design scheme acting from freely chosen distortion effects and for any specified multiscopic rendering device.

The above model exhibits some new parameters quantifying independent distortion effects. Those parameters may be analytically expressed from geometrical parameters of both shooting and rendering multiscopic devices. Their relations to geometrical parameters and impact on distortion effects are now presented

– $k_{i_2}$ control(s) the global enlarging factor(s),

$$k_{i_2} = \frac{d_{i_2}}{D_{i_2}}. \tag{8}$$

– $\epsilon_i$ control(s) the potential nonlinear distortion which transforms a cube into a pyramid trunk according to the global reducing rate $a_i = \epsilon_i + k_{i_2}(\epsilon_i - 1)(Z/d_{i_2})$ possibly varying along $Z$,

$$\epsilon_i = \frac{b_{i_2}}{B_i}\frac{W_b}{W}. \tag{9}$$

– $\mu_i$ control(s) width over depth relative enlarging rate(s), or horizontal/depth anamorphose factor,

$$\mu_i = \frac{b_{i_2}}{k_{i_2}B_i}. \tag{10}$$

– $\rho$ control(s) height over width relative enlarging rate(s), or vertical/horizontal anamorphose factor,

$$\rho = \frac{W_b}{H_b}\frac{H}{W}. \tag{11}$$

– $\gamma_i$ control(s) the horizontal "shear" rate(s) of the perceived depth effect,

$$\gamma_i = \frac{c_ib_{i_2} - e_iB_i}{d_{i_2}B_i}. \tag{12}$$

– $\delta_i$ control(s) the vertical "shear" rate(s) of the perceived depth effect by an observer whose overhang complies to what is expected,

$$\delta_i = \frac{p_{i_2} B_i - P_{i_2} b_{i_2} \rho}{d_{i_2} B_i}. \tag{13}$$

Thus we have defined the depth distortion possibilities using the previously established shooting and viewing geometries. Moreover, this model makes possible to quantify those distortions for any couple of shooting and viewing settings by simple calculus based upon their geometric parameters.

## 4. Shooting Design Scheme for Chosen Distortion

One can use any multiscopic shooting device with any multiscopic viewing device while giving an effect of depth to any well-placed viewer (3D movie theater for example) but Section 3 shows that distortions will not be similar for each couple of technologies. In this section, we will design the shooting geometry needed to obtain, on a given viewing device, a desired distortion: whether perfect depth or chosen distortion effect of shot scene.

Knowing how distortions, shooting and viewing parameters are related, it becomes possible to derive shooting layout from former distortion and viewing choices.

We will describe two shooting layout design schemes complying to this use of the distortion model:

– a generic scheme allowing for a precise control of each distortion parameter,

– a more dedicated one of huge interest as it is focused on "non distortion" or "perfect depth", allowing at most control of global enlarging factor(s) $k_{i_2}$ as any other distortion parameter is set to its "non distortion value".

*4.1. Controlled Depth Distortion.* To define the shooting layout using this scheme, we control global enlargement (by $k_{i_2}$) and 4 potential depth distortions:

(1) when $\epsilon_i \neq 1$, a global nonlinearity which results in a deformation of the returned volume onto a "pyramid trunk" (as $a_i$ varies along $Z$ axis),

(2) when $\gamma_i \neq 0$, a sliding or "horizontal shear" of the returned volume according to the depth,

(3) when $\delta_i \neq 0$ and/or when the real overhang of the observer differs from the optimal overhang, a sliding or "vertical shear" of the returned volume according to the depth and

(4) when $\mu_i \neq 1$ and/or $\rho \neq 1$, an anamorphose producing uneven distentions of the 3 axis ($X$ versus $Z$ for $\mu_i$ and $Y$ versus $X$ for $\rho$).

The depth controlled-distortion is obtained by adjusting the enlarging factor(s) $k_{i_2}$ and the distortion parameters $\epsilon_i$ (and so $a_i = \epsilon_i * k_{i_2}(\epsilon_i - 1)/d_{i_2}$), $\mu_i$, $\rho$, $\gamma_i$ and $\delta_i$. This last condition on $\delta_i$ is more delicate because it depends on the height of the viewer which inevitably affects the effective overhang towards the device. So the chosen vertical sliding $\delta_i$ can be reached only for observer whose overhang is defined in the viewing settings for this observation position.

Table 1: Proposed solutions according to the main features of scenes to be shot.

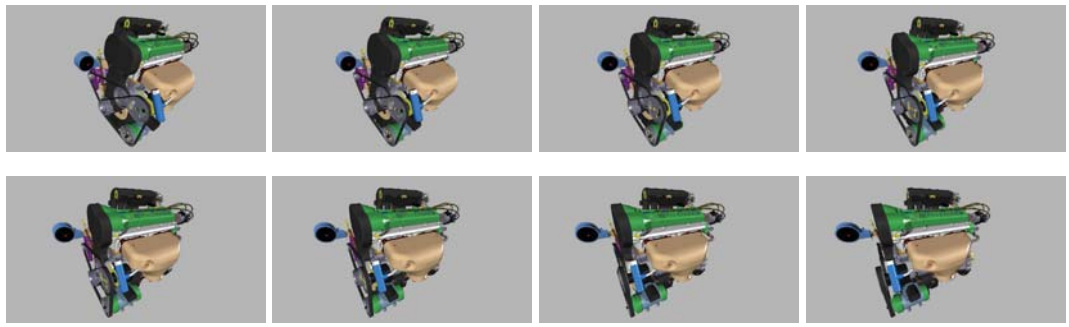
| | Photo | | Video |
|---|---|---|---|
| Scene | Still | Animated | Animated |
| Real | 3D-CAM<br>Photo Rail | 3D-CAM | 3D-CAM |
| Virtual | 3D computer graphics software | | |

Thus, given the viewing settings and the desired distortion parameters, the shooting parameters can be calculated as follows:

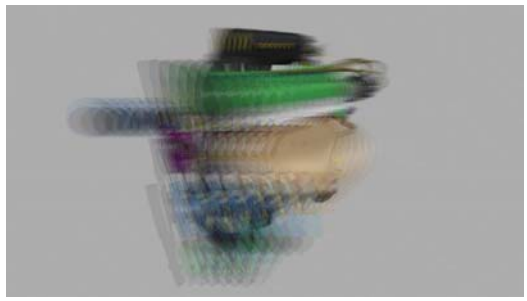
$$\begin{gathered}
P_{i_2} = \frac{p_{i_2} - \delta_i d_{i_2}}{k_{i_2} \rho \mu_i}, \qquad D_{i_2} = \frac{d_{i_2}}{k_{i_2}}, \\
W_b = \frac{W \epsilon_i}{k_{i_2} \mu_i}, \qquad H_b = \frac{H \epsilon_i}{k_{i_2} \rho \mu_i}, \\
c_i = \frac{e_i + \gamma_i d_{i_2}}{k_{i_2} \mu_i} \\
f_i \text{ imposed or chosen, individually } \forall i \in \{1 \dots n\} \times \\
\{1, \dots, m\}, \text{ by lot } \forall i_2 \in \{1, \dots, n\} \text{ or on the whole :} \\
w_i = \frac{W_b f_i}{D_{i_2}} = \frac{W f_i \epsilon_i}{\mu_i d_{i_2}}, \\
h_i = \frac{H_b f_i}{D_{i_2}} = \frac{H f_i \epsilon_i}{\mu_i \rho d_{i_2}}, \\
\alpha_i = \frac{c_i f_i}{D_{i_2}} = \frac{f_i (e_i + \gamma_i d_{i_2})}{\mu_i d_{i_2}}, \\
\beta_i = \frac{P_{i_2} f_i}{D_{i_2}} = \frac{f_i (p_{i_2} - \delta_i d_{i_2})}{\mu_i \rho d_{i_2}}.
\end{gathered} \tag{14}$$

This depth controlled-distortion scheme allows to obtain the parameters of a shooting layout producing desired 3D content for any rendering device and any depth distortions combination.

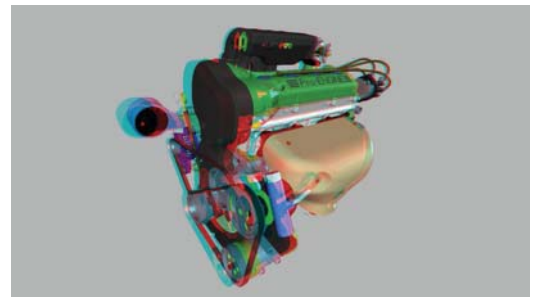
*4.2. Perfect Depth Effect.* A particular case of the depth controlled-distortion is a perfect depth effect (depth perception without any distortion compared with the depth of the shot scene). To produce a perfect depth effect (whatever the enlarging factor(s) $k_{i_2}$), we should configure the shooting in order to avoid the 4 potential distortions. This is obtained by making sure that the distortion parameters verify $\epsilon_i = 1$, $\mu_i = 1$, $\rho = 1$, $\gamma_i = 0$ and $\delta_i = 0$. The last condition $\delta_i = 0$ is more delicate, as it can be assured only for an observer complying to the defined overhang.



(a) One temporal frame: a set of 8 images



(b) Autostereoscopic mix



(c) Anaglyph of images 3 and 4 (Note 1)

Figure 4: Images synthesis of engine photo with courtesy of PI3C, partner of PTC.

In case of shooting for perfect depth effect, the shooting parameters can be calculated as below:

$$P_{i_2} = \frac{p_{i_2}}{k_{i_2}}, \qquad D_{i_2} = \frac{d_{i_2}}{k_{i_2}},$$

$$W_b = \frac{W}{k_{i_2}}, \qquad H_b = \frac{H}{k_{i_2}},$$

$$c_i = \frac{e_i}{k_{i_2}}$$

$f_i$ imposed or chosen, individually $\forall i \in \{1 \dots n\} \times \{1 \dots m\}$, by lot $\forall i_2 \in \{1 \dots n\}$ or on the whole. (15)

$$w_i = \frac{W_b f_i}{D_{i_2}} = \frac{W f_i}{d_{i_2}},$$

$$h_i = \frac{H_b f_i}{D_{i_2}} = \frac{H f_i}{d_{i_2}},$$

$$\alpha_i = \frac{c_i f_i}{D_{i_2}} = \frac{e_i f_i}{d_{i_2}},$$

$$\beta_i = \frac{P_{i_2} f_i}{D_{i_2}} = \frac{p_{i_2} f_i}{d_{i_2}}.$$

This particular case is very interesting for its realism, for example: in order to convince financiers or deciders, it may be important to give the real volumetric perception of a building, or a mechanical piece, in a computer aided design (CAD) application, or medical visualization software, in a surgical simulation application.

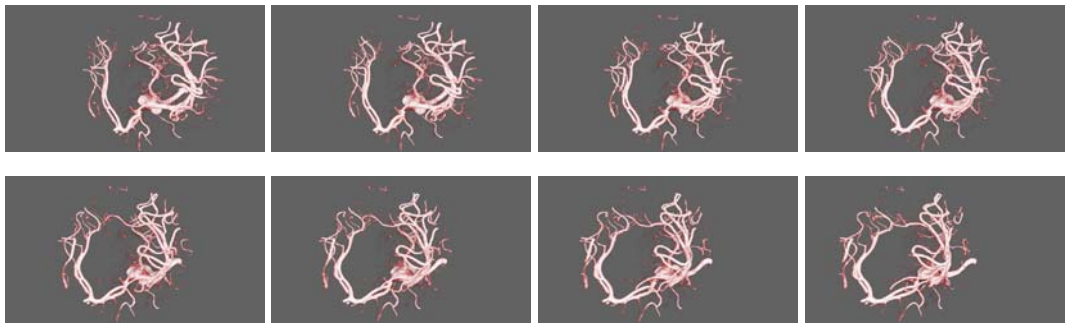
## 5. Derived Shooting Technologies

*5.1. 3D Computer Graphics Software.* Thanks to these design schemes, we have created 3 different technologies to shoot 3D scenes: multi-viewpoint computer graphics software, photo rail and camera box system. These products have been developed under the brand "3DT 5 Solutions" and patents are pending for each of then. As shown in Table 1, we have developed 3 solutions to obtain qualitative photo or video content for any relevant kind of scene still or animated, real or virtual). We use anaglyph to illustrate our results even if their viewing on paper or 2D screen media is not optimum because the images have been calculated to be rendered on specific devices.

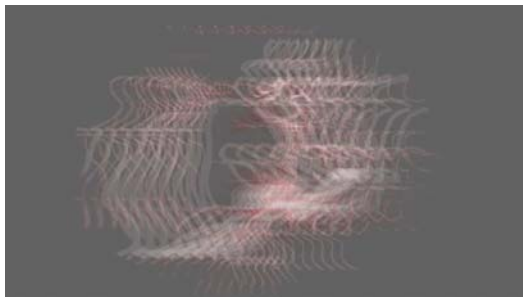
The common goal of our 3D computer graphics pieces of software is to virtually shoot different scenes, as photo of still or animated scene as well as computer animations of animated scene. Thanks to the previous shooting design scheme, we are able to place the virtual sensors around a usual monocular camera according to the chosen viewing device in order to obtain the desired depth effect. In this case, virtual cameras are perfect and there is no issue with distortions due to sensors or lenses.

Thus we have, by now, developed plugins and software to visualize and handle in real-time files from CAD software such as AutoCAD, Archicad, Pro/Engineer, etc. as well as medical data, such as MRI. We are going to apply this technology to other virtual scenes.

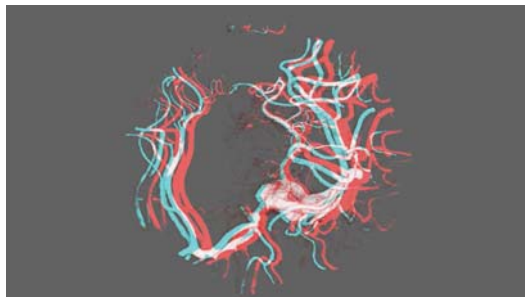
In those software pieces, we choose the rendering device parameters and the desired depth effect, and the software computes and uses its virtual shooting layout. It is possible to record different rendering devices and depth effect



(a) One temporal frame: a set of 8 images



(b) Autostereoscopic mix



(c) Anaglyph of images 3 and 4 (Note 1)

Figure 5: Images synthesis of image of an aneurysm.

distortions; then to switch easily between these devices and these distortions.

Those pieces of software currently handle scenes up to 7 million polygons at interactive rate.

Figure 4 shows an example of images of Parametric Technology Corporation (PTC) piece (a engine) shot as the software was tuned to achieve a perfect depth effect on an autostereoscopic parallax display 57″ (optimal viewing distance 4.2 m) (Note 1).

Figure 5 shows medical data (image of an aneurysm). The software was tuned for autostereoscopic parallax display 57″ (optimal viewing distance 4.2 m) and a perfect depth effect is obtained, as it is usual with medical data in order to allow the most definite and efficient interpretation possible by the surgeon (Note 1).

*5.2. Photo Rail.* The goal of this photo rail is to shoot a 3D photo of still scene. By using the photo rail (Figure 6) with its controlling software it is possible to control both the usual operations of a professional digital camera and its movement along a linear axis parallel to its sensor rows.

This allows us to carry out any shooting configuration whatever the chosen depth distortion settings and viewing device if we crop the needed capture area $CA_i$ in each digital photo in order to comply to the needed shooting geometry. With this Photo Rail, there is a possibility of distortion due to the digital camera, but distortions will be consistent for all images and of negligible magnitude, as it is professional equipment. We have not tried to correct those possible distortions but such a work could be done easily.

(a) (b)

Figure 6: Photo Rail

For example, Figure 7 illustrates the shooting of a room in "Palais du TAU" [21] in Reims with a perfect depth effect for autostereoscopic parallax display 57″ (optimal viewing distance 4.2 m). We made a 3D shooting of a big hall with a significant depth (Note 1). To realize this sequence, we positioned the convergence point at 4.2 m from the photo rail. Moving the digital camera along the rail, taking pictures and storing them took us 39.67 s.

*5.3. Camera Box System.* The goal of this camera box system is to shoot different scenes, as photo of still or animated scene as well as video of animated scene. Thanks to the previous shooting design method, we know how to realize a camera box system containing several couples of lenses and image sensors in order to produce simultaneously the multiple images required by an autostereoscopic display with a desired depth effect. As these couples are multiple, their induced

(a) One temporal frame: a set of 8 images

(b) Autostereoscopic mix

(c) Anaglyph of images 3 and 4 (Note 1)

Figure 7: "Palais du Tau" result of Photo Rail with courtesy of Monum.

distortions can be different. We have introduced a couple-by-couple process of calibration/correction based upon the model of Zhang [22].

We have already produced two prototypes of camera box system delivering multi-video stream in real-time (25). Their layout parameters have been defined for no distortion of specific scenes (see below) and set at manufacturing:

- The 3D-CAM1 (Figure 8(a)) which allows to shoot a life size scene (ratio $k_i = 1$) of the bust of a person to be viewed on an autostereoscopic parallax display 57″ (optimal viewing distance 4.2 m). We have chosen to obtain a perfect depth effect. According to numerous viewers both novice and expert, the 3D perception is really qualitative. Unfortunately, such an experiment is impossible to produce on 2D media (Note 1). Figure 9 shows some results. Figure 9(a) corresponds to a set of 8 simultaneously shot images, Figure 9(b) is the result after their autostereoscopic mixing and Figure 9(c) is anaglyph of images indexed 3 and 4.
- The 3D-CAM2 (Figure 8(b)) enables to shoot small size objects (in the order of 10–20 cm) and to display them on an autostereoscopic lenticular display 24" (optimal viewing distance 2,8) with an enlargement factor set to $k_i = 1,85$. We have chosen to obtain a perfect depth effect. Here again the 3D perception is really qualitative according to our numerous visitors. You will find some illustrations in Figure 10. Figure 10(a) corresponds to a set of 8 simultaneously shot images and Figure 10(b) gives the result in anaglyph of images indexed 3 and 4 (Note 1).

(a) 3D-CAM1

(b) 3D-CAM2

Figure 8: Camera box systems.

## 6. Conclusion

This work firstly models geometrical distortions between the shot scene and its multiscopically viewed avatar. These distortions are related to geometrical parameters of both the shooting and rendering devices or systems. This model enables quantitative objective assessments on the geometrical reliability of any multiscopic shooting and rendering couple.

The formulas expressing distortion parameters from geometrical characteristics of the shooting and rendering devices have afterwards been inverted in order to express the desirable shooting layout yielding a chosen distortion scheme upon a chosen rendering device. This design scheme *a priori* insures that the 3D experience will meet the chosen requirements for each expected observer position. Such a scheme may prove highly valuable for applications needing reliable accurate 3D perception or specific distortion effects.

This design scheme has been applied to 3 different kinds of products covering any needs of multi-viewpoint scene shooting (real/virtual, still/animated, photo/video).

(a) One temporal frame: a set of 8 images

(b) Autostereoscopic mix

(c) Anaglyph of images 3 and 4 (Note 1)

Figure 9: The results of 3D-CAM1.

(a) One temporal frame: a set of 8 images

(b) Anaglyph of images 3 and 4 (Note 1)

Figure 10: The results of 3D-CAM2.

This work proposes several perspectives. We take an interest in the combination of real and virtual 3D scene, that is 3D augmented reality. This allows to combine virtual and real elements or scenes. In the CamRelief project, Niquin et al. [23] work on this subject and present their first results concerning accurate multi-view depth reconstruction with occlusions handling.

We are developing a configurable camera with geometric parameters possibly flexible in order to fit to a chosen rendering device and a desired depth effect. Thus, we could test different depth distortions for a same scene. Moreover, we could produce qualitative 3D content for several rendering devices from a single camera box.

We will need to do some experiments on a fussy subject as we have to validate that the perception is geometrically conform to our expectations. This will require a significant panel of viewers but also to define and set up the perception test which will permit to precisely quantify the distances between some characteristic points of the perceived scene.

*Note 1.* The 3D content has been produced for autostereoscopic display. Obviously, it can only be experimented with the chosen device and in no way upon 2D media such as paper or conventional display. Nevertheless, anaglyphs help the reader to notice the depth effect on such 2D media.

## Acknowledgments

We would you like to thank the ANRT, the French National Agency of Research and Technology for its financial support. The work reported in this paper was also supported as part of the CamRelief project by the French National Agency of Research. This project is a collaboration between the University of Reims Champagne-Ardenne and TéléRelief. We would you like to thank Michel Frichet, Florence Debons, and the staff for their contribution to the project.

## References

[1] W. Sanders and D. F. McAllister, "Producing anaglyphs from synthetic images," in *Stereoscopic Displays and Virtual Reality Systems X*, vol. 5006 of *Proceedings of SPIE*, pp. 348–358, 2003.

[2] E. Dubois, "A projection method to generate anaglyph stereo images," in *Proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP '01)*, vol. 3, pp. 1661–1664, IEEE Computer Society Press, 2001.

[3] R. Blach, M. Bues, J. Hochstrate, J. Springer, and B. Fröhlich, "Experiences with multi-viewer stereo displays based on lc-shutters and polarization," in *Proceedings of IEEE VR Workshop: Emerging Display Technologies*, 2005.

[4] K. Perlin, S. Paxia, and J. S. Kollin, "An autostereoscopic display," in *Proceedings of the 27th ACM Annual Conference on Computer Graphics (SIGGRAPH '00)*, vol. 33, pp. 319–326, 2000.

[5] N. A. Dodgson, "Analysis of the viewing zone of multi-view autostereoscopic displays," in *Stereoscopic Displays and Virtual Reality Systems IX*, vol. 4660 of *Proceedings of SPIE*, pp. 254–265, 2002.

[6] K. Müller, A. Smolic, K. Dix, P. Merkle, P. Kauff, and T. Wiegand, "View synthesis for advanced 3D video systems," *EURASIP Journal on Image and Video Processing*, vol. 2008, Article ID 438148, 11 pages, 2008.

[7] U. Güdükbay and T. Yilmaz, "Stereoscopic view-dependent visualization of terrain height fields," *IEEE Transactions on Visualization and Computer Graphics*, vol. 8, no. 4, pp. 330–345, 2002.

[8] T. Yilmaz and U. Güdükbay, "Stereoscopic urban visualization based on graphics processor unit," *Optical Engineering*, vol. 47, no. 9, Article ID 097005, 2008.

[9] S. Fu, H. Bao, and Q. Peng, "An accelerated rendering algorithm for stereoscopic display," *Computers and Graphics*, vol. 20, no. 2, pp. 223–229, 1996.

[10] O. Faugeras, Q.-T. Luong, and T. Papadopoulou, *The Geometry of Multiple Images: The Laws That Govern the Formation of Images of a Scene and Some of Their Applications*, MIT Press, Cambridge, Mass, USA, 2001.

[11] R. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision*, Cambridge University Press, New York, NY, USA, 2000.

[12] G. Jones, D. Lee, N. Holliman, and D. Ezra, "Controlling perceived depth in stereoscopic images," in *Stereoscopic Displays and Virtual Reality Systems VIII*, vol. 4297 of *Proceedings of SPIE*, pp. 42–53, June 2001.

[13] R. T. Held and M. S. Banks, "Misperceptions in stereoscopic displays: a vision science perspective," in *Proceedings of the 5th Symposium on Applied Perception in Graphics and Visualization (APGV '08)*, pp. 23–32, ACM, New York, NY, USA, 2008.

[14] E. Peinsipp-Byma, N. Rehfeld, and R. Eck, "Evaluation of stereoscopic 3D displays for image analysis tasks," in *Stereoscopic Displays and Applications XX*, vol. 7237 of *Proceedings of SPIE*, 2009, 72370L.

[15] A. J. Hill, "A mathematical and experimental foundation for stereoscopic photography," *SMPTE Journal*, vol. 61, pp. 461–486, 1953.

[16] J.-Y. Son, Y. N. Gruts, K.-D. Kwack, K.-H. Cha, and S.-K. Kim, "Stereoscopic image distortion in radial camera and projector configurations," *Journal of the Optical Society of America A*, vol. 24, no. 3, pp. 643–650, 2007.

[17] H. Yamanoue, "The relation between size distortion and shooting conditions for stereoscopic images," *SMPTE Journal*, vol. 106, no. 4, pp. 225–232, 1997.

[18] H. Yamanoue, "The differences between toed-in camera configurations and parallel camera configurations in shooting stereoscopic images," in *Proceedings of IEEE International Conference on Multimedia and Expo (ICME '06)*, pp. 1701–1704, 2006.

[19] A. Woods, T. Docherty, and R. Koch, "Image distortions in stereoscopic video systems," in *Stereoscopic Dispalys and Applications IV*, vol. 1915 of *Proceedings of SPIE*, pp. 36–48, 1993.

[20] N. A. Dodgson, J. R. Moore, and S. R. Lang, "Time-multiplexed autostereoscopic camera system," in *Stereoscopic Displays and Virtual Reality Systems IV*, vol. 3012 of *Proceedings of SPIE*, 1997.

[21] http://palais-tau.monuments-nationaux.fr.

[22] Z. Zhang, "A flexible new technique for camera calibration," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, no. 11, pp. 1330–1334, 2000.

[23] C. Niquin, S. Prévost, and Y. Remion, "Accurate multi-view depth reconstruction with occlusions handling," in *Proceedings of the 3DTV-Conference: The True Vision—Capture, Transmission and Display of 3D Video (3DTV-CON '09)*, Postdam, Germany, May 2009.

Hindawi Publishing Corporation
International Journal of Digital Multimedia Broadcasting
Volume 2010, Article ID 942037, 12 pages
doi:10.1155/2010/942037

*Research Article*

# Near Real-Time Estimation of Super-Resolved Depth and All-In-Focus Images from a Plenoptic Camera Using Graphics Processing Units

**J. P. Lüke,[1] F. Pérez Nava,[2] J. G. Marichal-Hernández,[1] J. M. Rodríguez-Ramos,[1] and F. Rosa[1]**

[1] *Departamento de Física Fundamental y Experimental, Electrónica y Sistemas, Universidad de La Laguna, 38203 Canary Islands, Spain*
[2] *Departamento de Estadística, Investigación Operativa y Computación, Universidad de La Laguna, 38203 Canary Islands, Spain*

Correspondence should be addressed to J. P. Lüke, jpluke@ull.es

Received 30 April 2009; Revised 4 September 2009; Accepted 14 September 2009

Academic Editor: Xenophon Zabulis



Depth range cameras are a promising solution for the 3DTV production chain. The generation of color images with their accompanying depth value simplifies the transmission bandwidth problem in 3DTV and yields a direct input for autostereoscopic displays. Recent developments in plenoptic video-cameras make it possible to introduce 3D cameras that operate similarly to traditional cameras. The use of plenoptic cameras for 3DTV has some benefits with respect to 3D capture systems based on dual stereo cameras since there is no need for geometric and color calibration or frame synchronization. This paper presents a method for simultaneously recovering depth and all-in-focus images from a plenoptic camera in near real time using graphics processing units (GPUs). Previous methods for 3D reconstruction using plenoptic images suffered from the drawback of low spatial resolution. A method that overcomes this deficiency is developed on parallel hardware to obtain near real-time 3D reconstruction with a final spatial resolution of $800 \times 600$ pixels. This resolution is suitable as an input to some autostereoscopic displays currently on the market and shows that real-time 3DTV based on plenoptic video-cameras is technologically feasible.

## 1. Introduction

Since the introduction of television, a great effort has been made to improve the overall experience for viewers. Improvements in color, picture quality, and sound quality have contributed to an enhanced viewing experience. It is believed that three-dimensional television (3DTV) that enables people to watch content in three dimensions is the next step in the evolution of television. Proponents of 3DTV have argued that it will bring the viewer a more natural and life-like visual home entertainment experience [1]. Although there have been several attempts to initiate 3DTV [2] over the last decades, its introduction is considered as becoming increasingly feasible thanks to recent technologies and advances in the fields of camera development, image processing, and display design.

A complete technological solution for 3DTV must consider the whole production chain involving content creation, coding, transmission, and display [3]. In contrast to traditional television, 3DTV has to capture multiview information about the scene in order to reconstruct 3D information. Most 3D material is shot using a dual-camera configuration or using an array of cameras. The storage and transmission of this 3D material involves a large amount of data because one stereoscopic image consists of multiple views. Therefore, a considerable effort is involved in compressing the digital images in order to obtain savings in bandwidth and storage capacity. This is of particular relevance in the case of 3D HDTV, where a single uncompressed HDTV channel may consume up to one Gbit/s transmission bandwidth, or in the case of 3D video transmission over low-bandwidth transmission channels, such as the Internet [4, 5]. In terms of compatibility with existing broadcast systems, double the bandwidth would be needed for transmitting the left- and right-eye views of a dual camera. The use of a depth range camera recording the RGB image and accompanying

depth value per pixel (2D plus depth format) overcomes this bandwidth problem. There are several developments in this area: active cameras like Axi-Vision by NHK [6] or Z-cam by 3DV [7] or passive cameras like the CAFADIS plenoptic video-camera [8, 9]. Although this is a promising area, there are still some challenges to accurately recovering depth with enough spatial and range resolution such that the viewing comfort is at least comparable to that of conventional 2D television. Transmitted images are shown on stereoscopic or autostereoscopic displays [10]. Stereoscopic displays require the viewer to wear an optical device (e.g., polarized glasses, shutter glasses) to direct the left and right eye images to the appropriate eye, while the separation technique used in autostereoscopic displays is integrated into the display screen.

The development of autostereoscopic displays that accept the standard 2D plus depth format [11] makes possible a simple solution for the 3DTV production chain based on the recently developed depth range cameras. This solution is backward compatible with conventional broadcast television and ensures a gradual transition from one system to the other.

This paper addresses the creation of 3DTV content using a depth range plenoptic video-camera. It is clear that the introduction of 3DTV is conditioned on the existence of enough 3D content that can be displayed to viewers. The content creation process requires the capacity to capture scenes realistically or to recreate them using computer generated graphics. The capture of real scenes in 3DTV requires a significant technological effort. Multiview information has to be obtained with an array of two or more cameras and has to be processed in order to obtain the necessary 3D information. The use of camera arrays is not a portable solution and can thus hinder the filming process. Another disadvantage of camera arrays is the need for geometric and color calibration or frame synchronization between the cameras. It would be desirable for 3D capture to be done with a device of dimensions similar to those of traditional TV cameras. One such device for obtaining multiview information and which does not require calibration is a plenoptic video-camera [8, 9]. Plenoptic cameras use a microlens array behind the main lens to capture multiple views of the scene, generating a lightfield image that records the radiance and direction of all light rays in the scene. Using the lightfield image, it is possible to generate the focal stack using several approaches [12–15]. The focal stack serves as input for multiview stereo depth estimation algorithms. Using the estimated depth and the focal stack, it is straightforward to obtain the all-in-focus image and depth map of the scene.

An important drawback of plenoptic cameras is the generation of low-resolution images due to their spatioangular lightfield sampling [16]. The angular information recorded results in a small output image being produced in comparison with sensor size. Methods based on superresolution techniques that overcome this spatial resolution limitation have recently been presented [17, 18]; however, such methods are much too time intensive to be executed in real time on an ordinary CPU. A solution for achieving the real-time requirements of 3DTV is the use of parallelization techniques

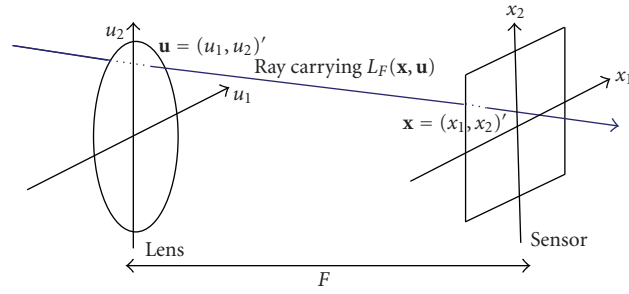



Figure 1: Two-plane parameterization of the lightfield.

that increase the computational processing power. In recent years, graphics processing units (GPU) have experienced enormous change, becoming parallel coprocessors that can be used to perform general purpose computations with a significant speedup [19]. In this article we describe an implementation on GPUs of the super-resolution and distance extraction algorithm presented in [18].

The rest of this paper is divided as follows: in Section 2 we present the super-resolution technique that obtains all-in-focus and depth images. Details of multi-GPU implementation are explained in Section 3. Section 4 shows some experimental results and finally Section 5 contains some concluding remarks and future work.

## 2. Simultaneous Superresolution Depth and All-In-Focus Images from Plenoptic Cameras

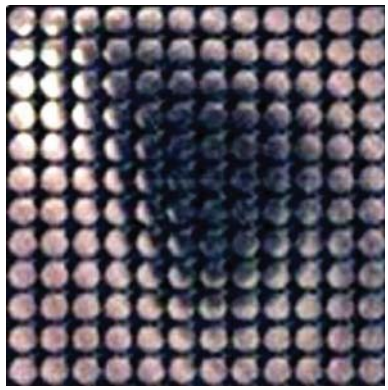
In this section we present a super-resolution technique [18] to produce depth and all-in-focus images from plenoptic cameras. This technique is based on the super-resolution discrete focal stack transform (SDFST) that generates a focal stack that is processed to estimate 3D depth. This estimated depth is then used to obtain the all-in-focus image. We present the entire process in this section.

*2.1. The Focal Stack Transform (FST).* The SDFST is an extension of the Focal Stack Transform [12, 13] that processes a lightfield image from a plenoptic camera and generates images focused on a desired frontal plane. The FST is based on the Photography transform [12] that generates conventional 2D photographs from a lightfield. To introduce the Photography transform it is necessary to parameterize the lightfield defined by all the light rays inside a camera. Using the two-plane parameterization we write $L_F(\mathbf{x}, \mathbf{u})$ as the radiance traveling from position $\mathbf{u} = (u_1, u_2)'$ (apostrophe means transpose) on the lens plane to position $\mathbf{x} = (x_1, x_2)'$ on the sensor plane. $F$ is the distance between the lens and the sensor (see Figure 1 adapted from [12]).

The lightfield $L_F$ can be used to compute conventional 2D photographs at any depth $\alpha F$. Let $\mathcal{P}_\alpha$ be the operator that transforms a lightfield $L_F$ at a sensor depth $F$ into a photograph at sensor depth $\alpha F$, then the Photography operator can be written as [12]

$$\mathcal{P}_\alpha[L_F](\mathbf{x}) = \frac{1}{\alpha^2 F^2} \int L_F\left(\mathbf{u}\left(1 - \frac{1}{\alpha}\right) + \frac{\mathbf{x}}{\alpha}, \mathbf{u}\right) d\mathbf{u}. \quad (1)$$

(a)



(b)

Figure 2: (a) Lightfield captured by a plenoptic camera. (b) Detail from the white square in the left image.

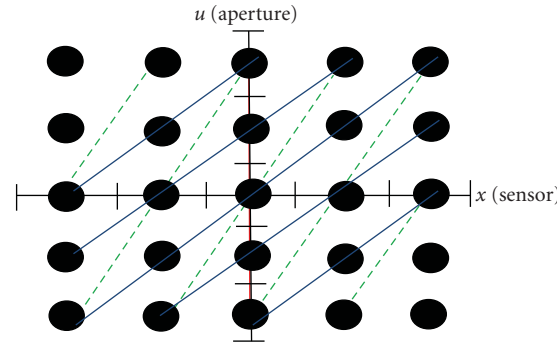



Figure 3: Photography transform for two particular $\alpha$ values.

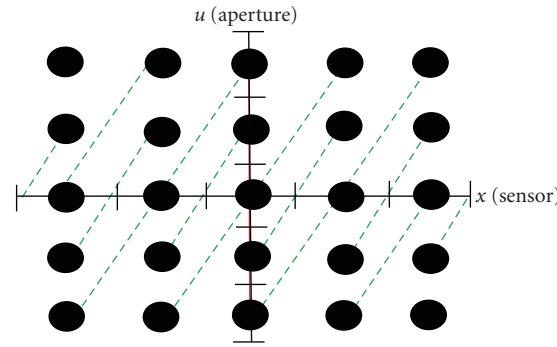



Figure 4: The lines used in the SDFST for a particular distance.

This equation shows how to compute $\mathcal{P}_\alpha[L_F]$ at different depths from the lightfield $L_F$. When photographs are computed for every sensor depth $\alpha F$, we obtain the focal stack transform $\mathcal{S}$ of the lightfield defined as [13]

$$\mathcal{S}[L_F](\mathbf{x}, \alpha) = \mathcal{P}_\alpha[L_F](\alpha\mathbf{x}). \tag{2}$$

A plenoptic camera only captures discrete samples of the lightfield. We will assume that the plenoptic camera is composed of $n \times n$ microlenses and each of them generates an $m \times m$ image. A discrete lightfield captured from a plenoptic camera is shown in Figure 2 [12]. To obtain a discretized version of the focal transform it is necessary to interpolate the lightfield $L_F$ and to discretize the integral in (1). There are several techniques for performing this interpolation [12–15]. A common result of these techniques is that the focal stack is composed of images with $n \times n$ pixels. Several focal stacks can be built depending on the application. We can use the radiance in the lightfield $L_F$ to obtain the usual focal stack, measures of focus to obtain the Laplacian focal stack [13, 15], or measures of photoconsistency to obtain the variance focal stack.

To introduce the super-resolution focal stack it is necessary to show how to compute the Photography transform on a discretized lightfield. To simplify the presentation, the super-resolution technique is shown in 2D instead of 3D. The extension of the results to 3D is straightforward. In the 2D case, to compute the Photography transform (1) at sensor depth $\alpha F$ it is necessary to compute a line integral on the lightfield. For each of the $n$ pixels in the $x$ axis, a sensor depth $\alpha F$ generates a line through $x$ whose slope is determined by $\alpha$ (see Figure 3). Since the line is continuous, to approximate the integral we have to sum all the values on the line and interpolate where pixels are not available.

*2.2. The Superresolution Discrete Focal Stack Transform .* The super-resolution extension to the Focal Transform (SDFST) is based on performing the Focal Stack Transform for a different set of lines. The selection of this new set of lines is explained in Figure 4.

The new set of lines for each slope is an extension of the initial set with lines of the same slope. This new set passes through every point in the discretized lightfield, not only through points on the $x$-axis. For the example above, intersecting these lines with the $x$-axis we see that we now have twice the resolution as before (see the dashed lines of Figures 3 and 4). The super-resolution algorithm only uses the samples that lie exactly on each line and obtains the extended resolution using the following theorem and corollary [18].

**Theorem 1.** *Given a lightfield* $L_F(x, u)$, $|x| \leq r|u| \leq s$, $n = 2r + 1$, $m = 2s + 1$, $0 < |\Delta x| \leq r$, $0 < |\Delta u| \leq s$, $\Delta u$, $\Delta x \in Z$, *if we pad* $L_F(x, u)$ *with* $|\Delta x_g|s$ *zero-columns on each side of the x-axis, the super-resolution focal stack image for slope* $\Delta u/\Delta x$

Input: Lightfield $L_F(\mathbf{x}, \mathbf{u})$, $\mathbf{x} = (x_1, x_2)'$, $\mathbf{u} = (u_1, u_2)'$, $|x| \leq r$ $|u| \leq s$.
Slope $\Delta u/\Delta x$ with $\Delta u, \Delta x \in Z$ as in the previous theorem
Output: Super-resolution focal stack image $F_{\Delta u/\Delta x}(\mathbf{k})$
Compute $g$ =g.c.d.$(\Delta u, \Delta x)$, $\Delta u_g = \Delta u/g$ and $\Delta x_g = \Delta x/g$
For each microlens $\mathbf{x} = (x_1, x_2)'$
  For each pixel in that microlens $\mathbf{u} = (u_1, u_2)'$
    Compute $\mathbf{k} = (k_1, k_2)'$, $\mathbf{k} = \Delta u_g \mathbf{x} - \Delta x_g \mathbf{u}$
    Update the mean value in $F_{\Delta u/\Delta x}(\mathbf{k})$ using $L_F(\mathbf{x}, \mathbf{u})$

Algorithm 1

*has* $|\Delta u_g| n + |\Delta x_g| m - |\Delta u_g| - |\Delta x_g| + 1$ *points with* $\Delta u_g = \Delta u/g$ *and* $\Delta x_g = \Delta x/g$, *where* $g$ *is the greatest common divisor (g.c.d.) of* $|\Delta u|$ *and* $|\Delta x|$.

**Corollary 2.** *Under the conditions of the preceding theorem the collection of slopes* $\{\Delta u/\Delta x\ 0 < |\Delta x| < s, \Delta u = s, \Delta u, \Delta x \in Z$ *with* $s$ *a prime number}* *can generate a super-resolution focal stack with* $\Delta u n - \Delta u + 1$ *pixels in each image.*

Therefore, the final resolution is approximately $\Delta u n \approx mn/2$, that is, half the full resolution of the sensor. The extension of the above results to 3D using squared aperture and microlenses is trivial. The final resolution in that case is approximately the full resolution divided by 4. In actual plenoptic images the resolution is somewhat lower due to the circular shape of microlenses and their edge effects. The algorithm for computing an image of the super-resolution focal stack is shown in Algorithm 1 [18]. In Algorithm 1, $\mathbf{x} = (x_1, x_2)'$ and $\mathbf{u} = (u_1, u_2)'$ are the coordinates of the microlens and the coordinates of the pixel within its microlens, respectively. $g$ is the g.c.d.$(\Delta u, \Delta x)$. As shown in the previous theorem, in order to maximize the spatial resolution the value of $g$ must be equal to 1. Each plane of the focal stack has associated with it a slope $\Delta u/\Delta x$ of integration so that Algorithm 1 has to be executed for each plane. $\Delta u, \Delta x \in Z$ are the discrete points in the sampled lightfield. $\mathbf{k} = (k_1, k_2)'$ is the coordinate of a pixel in the focal stack plane $F_{\Delta u/\Delta x}$ where lightfield data is accumulated to finally give the photograph focused at the depth associated with the slope $\Delta u/\Delta x$.

The computational complexity for the complete super-resolution focal-stack in the 3D case is $O(n^2 \times m^3)$.

The result of the algorithm for a region of the lightfield in Figure 2 is shown in Figure 5. The algorithm computes the super-resolved focal stack, generating several photographs of the same object focused at different depths. The resulting images have a 25× spatial resolution magnification with respect to previous approaches based on plenoptic cameras [12–15].

*2.3. Multiview Stereo over the Superresolution Focal Stack.* The multiview stereo algorithm operates on the variance focal stack and is able to obtain a $t \times t$ estimation of depth ($t = \Delta u n - \Delta u + 1$) with $m - 3$ different depth values and a $t \times t$ all-in-focus image. The variance focal stack is based on the photoconsistency assumption (PA) which states that the radiance of rays from a 3D point over all directions is the same (the Lambertian model). To test the PA assumption the variance of all points along every line is measured (see Figure 4). The multiview stereo algorithm uses this variance measure to choose the optimal depth. A straightforward approach for solving the multiview stereo problem would be to choose the line with less variance for each pixel. However it is well known that more assumptions are necessary if good results are to be obtained. The assumption that surfaces are smooth makes it possible to use the variance focal stack and the Markov Random Field (MRF) approach to obtain the optimal depth by minimizing an energy function [20]. The energy function is composed of a data term $E_d$ and a smoothness term $E_s$, $E = E_d + \lambda E_s$, where the parameter $\lambda$ measures the relative importance of each term. The data term is simply the sum of the per-pixel data costs, $E_d = \sum_{\mathbf{p}} c_{\mathbf{p}}(d)$, where $c_{\mathbf{p}}(d)$ is the variance measure for pixel $\mathbf{p}$ in the superresolved image and $d = \Delta u/\Delta x$ is a specific slope (see Figure 4). The smoothness term is based on the 4-connected neighbors of each pixel and can be written as $E_s = \sum_{\mathbf{p},\mathbf{q}} V_{\mathbf{pq}}(d_{\mathbf{p}}, d_{\mathbf{q}})$ where $\mathbf{p}$ and $\mathbf{q}$ are two neighboring pixels. $V_{\mathbf{pq}}(d_{\mathbf{p}}, d_{\mathbf{q}})$ is defined as

$$V_{\mathbf{pq}}\left(d_{\mathbf{p}}, d_{\mathbf{q}}\right) = \begin{cases} 0, & \text{if } d_{\mathbf{p}} = d_{\mathbf{q}}, \\ 1, & \text{otherwise.} \end{cases} \tag{3}$$

The hierarchical belief propagation (BP) approach [20] is used for optimizing the energy function $E$. This algorithm passes messages around the 4-connected image grid in an

Figure 5: Results of the super-resolution focal stack with a 25× resolution magnification.

iterative process using the following message updating rule:

$$M^{i}_{\mathbf{p}\to\mathbf{q}}\left(d_{\mathbf{q}}\right) = \min_{d_{\mathbf{p}}}\left(h_{\mathbf{p}\to\mathbf{q}}\left(d_{\mathbf{p}}\right) + \lambda \cdot V_{\mathbf{pq}}\left(d_{\mathbf{p}}, d_{\mathbf{q}}\right)\right), \quad (4)$$

where $M^{i}_{\mathbf{p}\to\mathbf{q}}(d_{\mathbf{q}})$ is the message passed from pixel $\mathbf{p}$ to pixel $\mathbf{q}$ for disparity $d_{\mathbf{q}}$ at iteration $i$ and

$$h_{\mathbf{p}\to\mathbf{q}}\left(d_{\mathbf{p}}\right) = c_{\mathbf{p}}\left(d_{\mathbf{p}}\right) + \mu \sum_{\mathbf{s}\in(\mathbf{p})} M^{i-1}_{\mathbf{s}\to\mathbf{p}}\left(d_{\mathbf{p}}\right) - M^{i-1}_{\mathbf{q}\to\mathbf{p}}\left(d_{\mathbf{p}}\right), \quad (5)$$

where $N(\mathbf{p})$ is the four-connected neighborhood of pixel $\mathbf{p}$ and $\mu \in (0, 1]$. Once the iterative process has finished after a fixed number of $I$ iterations, the belief vector for every pixel is computed as

$$b_{\mathbf{q}}\left(d_{\mathbf{q}}\right) = c_{\mathbf{q}}\left(d_{\mathbf{q}}\right) + \mu \sum_{\mathbf{p}\in N(\mathbf{q})} M^{I}_{\mathbf{q}\to\mathbf{p}}\left(d_{\mathbf{q}}\right), \quad (6)$$

and the value of $d_{\mathbf{q}}$ that leads to the minimum of $b_{\mathbf{q}}(d_{\mathbf{q}})$ is selected as the optimal depth.

The general belief propagation algorithm needs $O(k^2)$ computing time per message, where $k$ is the number of labels in each pixel ($m - 3$ depth levels in our case). Several techniques can be used to decrease the computing and memory requirements of the algorithm [21]. Messages can be computed in $O(k)$ time by taking into account the particular structure of $V_{\mathbf{pq}}(d_{\mathbf{p}}, d_{\mathbf{q}})$ function and rewriting the message update rule as

$$M^{i}_{\mathbf{p}\to\mathbf{q}}\left(d_{\mathbf{q}}\right) = \min\left(h_{\mathbf{p}\to\mathbf{q}}\left(d_{\mathbf{q}}\right), \min_{d_{\mathbf{p}}}\left(h_{\mathbf{p}\to\mathbf{q}}\left(d_{\mathbf{p}}\right) + \lambda\right)\right). \quad (7)$$

The computational complexity for one iteration of the belief propagation algorithm in the 3D case is $O(n^2 \times m^2 \times m)$ using this linear update rule.

A bipartite graph approach can also be used to compute messages faster and reduce the memory requirements in half. Finally a hierarchical approach is also used for computing message updates since it runs much faster than the general BP while yielding comparable accuracy. The main difference between the hierarchical BP and general BP is that hierarchical BP works in a coarse-to-fine manner, first performing BP at the coarsest scale and then using the output from the coarser scale to initialize the input for the next scale.

## 3. Implementation on Multi-GPU

In recent years, graphics hardware has evolved toward massively parallel coprocessors that can be used to perform general purpose computation. GPUs are specially suited to problems requiring data-parallel computations with high arithmetic intensity [19]. This means that the same program can be executed in parallel on different data elements using an SIMD (single instruction multiple data) programming model, speeding up computations. Also, multiple GPUs can be used to increase the benefits of parallel computing. Many algorithms have been implemented on the GPU, and the results often yield a significant speed-up over the sequential CPU implementation of the same algorithm. On the GPU each data element has to be mapped to parallel processing threads that execute on an SIMD architecture. This requires a different approach than for sequential implementations.

Until now, general purpose processing on GPU was done through the graphics API using shader languages. This made programming very tricky since graphic primitives were used to implement nongraphic algorithms, resulting in overhead. Recently, nVidia released CUDA (Compute Unified Device Architecture) [22], which is a general purpose parallel computing architecture with a scalable programming model that is available in the latest generations of nVidia's graphics hardware. This programming model is structured in such a way that each GPU processes a kernel function that executes multiple threads in parallel. CUDA kernels have many similarities with fragment shaders, which enable stream processing. However, CUDA is conceived for general purpose parallel processing, eliminating the overhead of graphics API and also including some features of parallel processing like thread synchronization and random-access write to memory, also known as scatter. To execute kernel functions, threads are grouped into a grid of thread blocks. Each thread is identified by a block identifier and a thread identifier within the block. Identifiers can be multidimensional, up to three dimensions. Inside the kernel function, the thread identifier is used to determine which input data portion will be processed.

Memory accesses are possible in multiple memory spaces during kernel execution. Each thread has a private local memory. Each block has a shared memory for all its threads within the lifetime of the block. Finally, threads can access a global memory shared between all threads. Global memory is not cached, but by using convenient memory access patterns to coalesce various memory accesses into one and bringing data to faster shared memory, performance can be improved significantly.

Using multiple GPUs to improve timing results is also possible with CUDA. However, it is the responsibility of the programmer to design how tasks are scheduled to GPUs. In the ideal case, the problem can be divided into separate parts that can be processed in parallel, thus eliminating the need for communication between GPUs. If data has to be transferred between GPUs, it causes overhead. It is important to minimize data transfers because they are very time consuming. As in traditional parallel programming, adding more processors (GPUs) also leads to more communications overhead.

We have used CUDA to implement the algorithm described in Section 2 on a multiGPU personal computer equipped with two nVidia 9800GX2 cards, each with two GPUs. The four GPUs make for a total of 512 processing cores.

In a multi-GPU implementation there are two levels of parallel execution: multithread executions within a GPU and parallelism between GPUs. Different parts of the input data have to be scheduled over the available GPUs so as to minimize data transfers between GPUs.

We decided to divide the input into parts with some overlap between areas. The scheduling scheme is depicted in

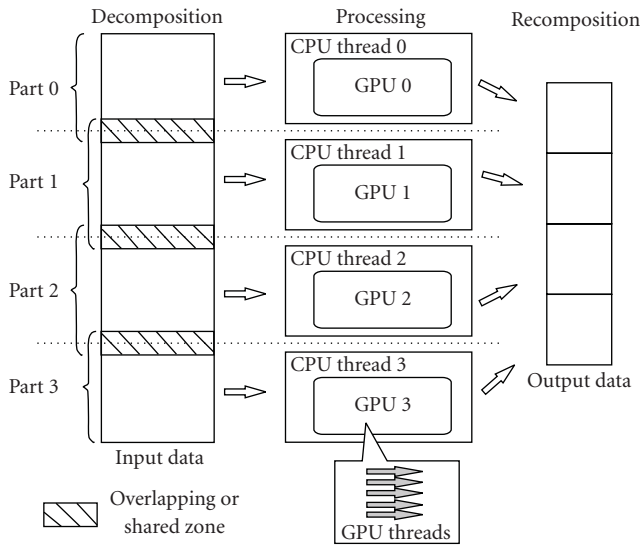



Figure 6: Data assignment scheme for multiGPU processing.

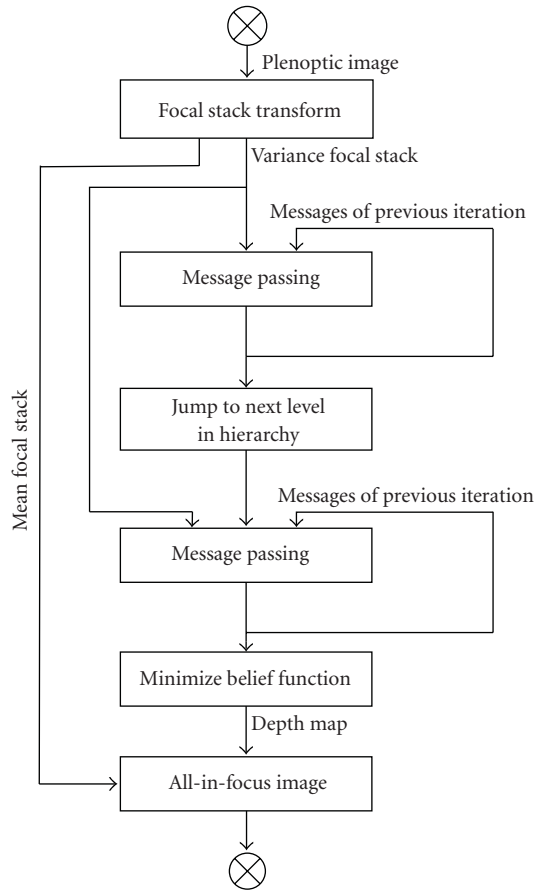



Figure 7: General flow of algorithm. Boxes represent kernel functions executing on the GPU.

Figure 6. It consists of a decomposition of the input data, a processing step that is mostly independent on each GPU, and a recomposition of the partial solution to obtain the global solution.

First, the lightfield input data is transferred from host to GPU with some overlap. This overlap allows us to compute the means and variance focal stacks without any data transfers between GPUs by performing some redundant computations. The result is a region of the focal stack with overlap on each GPU.

The same approach can now be followed by the belief propagation step, but the transfer of messages between GPUs is unavoidable. In each iteration, some messages corresponding to the edges of divisions have to be exchanged between GPUs. Once the iterative process is finished, a section of the depth map and all-in-focus image is obtained by each GPU. Finally the partial results are transferred back to the adequate location in host memory to yield the whole result.

Data processing on each GPU consists of several kernel executions as shown in Figure 7. The first step computes the super-resolved mean and variance focal stacks. Then a two-level hierarchical belief propagation is executed using the variance focal stack. Belief propagation has an iterative part consisting of the execution of the message passing kernel. After a certain number of iterations it jumps to the next level and an analogous iterative process takes place again. Then the belief function is minimized to obtain a super-resolved depth map. Finally, the all-in-focus image is generated using the depth map and focal stack.

*3.1. Parallelization of SDFST.* A close examination of the algorithm presented in Section 2.2 shows that the computation of each focal stack data element only requires a set of input pixels from the discrete lightfield captured with the plenoptic camera. This means that each focal stack data element can be computed in parallel if it is possible to simultaneously read the lightfield data.

To develop an efficient access scheme, it is necessary to find a mapping between the input and the output elements. The forward mapping relates the input elements from lightfield to output pixels of the focal stack, where accumulations for means and variances have to be done. Such an approach is used in the pseudocode presented in Section 2.2. The backward mapping, on the other hand, relates output memory positions to input memory positions of the lightfield, so that the necessary input pixels can be determined for a given output pixel position.

Global memory bandwidth is used more efficiently in CUDA when the thread memory accesses can be coalesced in a single memory access. nVidia GPUs with computing capability 1.1 have to access the memory addresses in sequence, so that the $j$th thread accesses memory position $j$ to achieve coalescence. Examining the mappings, it can be seen that sequential access is only possible for one side of the mapping. For the forward mapping, the input can be accessed with such a sequential pattern and output accumulations are updated with an uncoalesced read/write access. If backward mapping is used, the reading of input data is uncoalesced and one coalesced output write access can

be performed. Each thread can compute the accumulations in a register and then write the result to global memory. So backward mapping is more adequate because it needs fewer memory accesses at the uncoalesced side.

Given the forward mapping $\mathbf{k} = \Delta u_g \mathbf{x} - \Delta x_g \mathbf{u}$, the backward mapping is

$$\mathbf{x} = \frac{\mathbf{k} + \Delta x_g \mathbf{u}}{\Delta u_{\mathrm{g}}}, \quad x_i \in Z. \tag{8}$$

Once the backward mapping is determined, each CUDA thread will be responsible for computing the accumulations for one pixel of the mean and variance focal stacks. Each thread is identified by $\mathbf{k} = (k_1, k_2)$. Using the inverse mapping in (8), a set of $(\mathbf{x}, \mathbf{u})$ pairs, with $\mathbf{x} = (x_1, x_2)$ and $\mathbf{u} = (u_1, u_2)$, is determined from $\mathbf{k}$. Lightfield values for each $(\mathbf{x}, \mathbf{u})$ pair are loaded and accumulated. Squared values are also accumulated. Then the variance and mean focal stack is obtained.

*3.2. Parallelization of Belief Propagation.* The belief propagation algorithm is well suited for parallel implementation on a GPU. Several implementations exist using shader languages [23, 24]. CUDA implementations can also be found [25]. These GPU implementations show a significant speed up over CPU implementations.

Examining the linear time message updating rule (7), it can be seen that each output message only depends on the input messages of neighboring pixels in the previous iteration and on the data term for the current pixel. There is no dependence on other partial results. This means that message updating within the same iteration can be mapped easily on the CUDA execution model.

Special care has to be taken in the implementation of iterative message passing since the process is very time consuming.

The bipartite graph approach is used to increase speed up of the message passing step [21]. The image pixel set is divided into two disjoint subsets, A and B (checkerboard pattern), so that messages of A only depend on messages sent from B in the previous iteration and vice versa. This allows computing the messages of set A at odd iterations and messages of set B at even iterations. When the solution converges it can be assumed that $M^i_{p\to q}(d_q) = M^{i-1}_{p\to q}(d_q)$ so that a final solution can be constructed from both sets. This has the advantage that at each iteration only half of the messages have to be computed, which allows either doubling the speed or executing double the number of iterations in the same time. The memory requirements are also reduced by half since messages can be computed in place.

Messages have to be suitably organized so as to favor coalescent memory accesses. It is also convenient to operate in shared memory and once a result is obtained to write it back to global memory. To explain memory organization it is useful to describe memory addressing in multidimensional matrices. Incoming messages from the previous iteration are stored in a `float4` structure. Different memory organizations can be used to obtain linear addresses from a three-dimensional coordinate $(x, y, z)$, where $x$ and $y$ are spatial coordinates of the focal stack and $z$ represents the depth level. The differences in memory organization are essentially due to the order of coordinates when they are transformed into linear addresses.

During the iterative process, messages have to be exchanged between GPUs. This implies a data transfer from one GPU to host memory and from host memory to another GPU. It is convenient to reduce the number of calls to the `cudaMemcpy` function that performs the data transfers. This can be done by properly organizing messages in GPU memory. An adequate organization is the use of $(y, x, z)$ ordering of coordinates, because the partition between GPUs is done by rows and messages for all depth levels have to be transferred for each row. Any other ordering requires more memory copies because depth levels are not stored in adjacent positions.

Next, two possible implementation approaches of the message passing rule are discussed. The first one uses a parallel reduction scheme to compute the minimization of the messages during the update rule (4). The second approach is similar to that presented in [25] and performs minimization sequentially within the message passing kernel. However, in [25] a different expression for $V_{pq}$ is used.

In the first approach, each thread of the block computes (5) for its depth level and stores it in shared memory. After that, all threads in the block collaborate in a parallel reduction scheme to compute $\min_{d_p}(h_{p\to q}(d_p)) + \lambda$, which is a subexpression of (7) common to all depth levels. Then, each thread has to compute (7) for its depth level using the previously obtained global minimum. This implementation is limited by the amount of available shared memory, but it has the advantage that sequential execution of thread blocks in the grid is scheduled by CUDA, not by the programmer. From the view point of memory organization the $(y, x, z)$ ordering seems to be the most adequate since the messages for different depth levels are stored in adjacent memory positions and can be loaded with coalescent accesses. This memory organization also reduces the number of `cudaMemcpy` calls, reducing the overhead of data transfers between GPUs.

The second approach to implement the message updating rule is performing the global minimum sequentially [25]. On the GPU each thread is responsible for computing one message of the current iteration using messages from the previous one. After that, for each depth level $d_p$ the minimum between the global minimum and $h_{p\to q}(d_p)$ is computed. It offers the advantage of having an easier extension because the programming of the kernel function is more similar to traditional sequential programming. However, since each thread computes outgoing messages for each pixel, it does not make sense to share data between threads. Although shared memory can still be used to store intermediate results during message computation, this forces a reduction in the number of threads per block since the amount of memory required increases. This, along with the sequential coding style, has an adverse impact on performance. For this approach $(z, y, x)$ coordinate ordering is more suitable. However, the checkerboard pattern results in having only odd or even pixels used at each iteration for every row. This

does not favor coalescent memory accesses because the pixels to be read are not in adjacent memory positions.

For both approaches the results are written to the corresponding memory positions. This cannot be done with coalescent memory accesses because outgoing messages for each pixel have to be distributed to neighboring pixels accessing only one member of the `float4` structure. This problem could be solved by dividing the `float4` structure into individual memory zones. However, the checkerboard pattern used to implement the bipartite graph again breaks the coalescence.

After a certain number of iterations the algorithm jumps to the next hierarchy level and repeats the iterative message passing rule for the next level. When the iterative process is finished, the belief function has to be minimized to obtain the depth map.

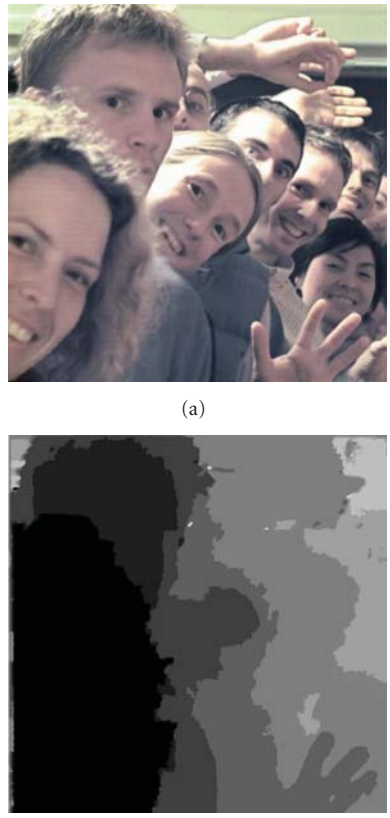
In the end it was decided to implement the first approach that consists in a message passing rule with a parallel reduction as this is expected to yield better performance.

## 4. Results

This section discusses some of the results obtained with the technique described in the preceding sections. First, some images and distance maps obtained from different plenoptic images are presented to show the effectiveness of the methods for high resolution images suitable for 3D HDTV. Since the objective was to generate images suitable for input to a commercial 3DTV autostereoscopic display, a more reduced lightfield of $160 \times 120 \times 15 \times 15$ was used to conduct performance tests of the GPU implementation. For this lightfield an $800 \times 600$ all-in-focus image and depth map for a Philips WOVvx 20″ display was generated [11]. Finally, the sensitivity of the parameter value for the belief propagation algorithm on the 3D reconstruction is studied.

*4.1. Experimental Results.* In Figure 8 we can see the all-in-focus image and depth estimation from the super-resolution focal stack and multiview stereo algorithms using the lightfield in Figure 2. Figure 9 shows an area from the plenoptic image [12] to compare the magnification results. The top image shows a nearest neighbor interpolation and the center image shows a bicubic interpolation of the $292 \times 292$ all-in focus image computed with previous approaches. Finally, the bottom image presents the results of the super-resolution technique. The results show an improvement in detail and noise reduction. It should also be noticed that from a theoretical point of view, super-resolution focal stack effectively increases resolution in lambertian scenes because each pixel in the final image is formed with data recorded directly from the scene using the plenoptic camera. In the case of interpolation techniques new pixel values are created from neighboring values without having direct information about them.

More results from other lightfield data [12] are shown in Figure 11 for the $256 \times 256 \times 15 \times 15$ lightfield of Figure 10. This shows that this method can also be used in wide range scenes.



(a)

(b)

Figure 8: Super-resolved depth and all-in-focus image.

The main drawback that can be observed in the results is the low depth resolution. This is due to the low number of pixels behind each microlens ($m = 15$). This could be solved by using a different plenoptic camera configuration with a higher value of $m$.

*4.2. Performance Tests.* To study the feasibility of real-time 3D reconstruction with a plenoptic video-camera, some performance measures were obtained for the multiGPU implementation that uses the parallel reduction technique during the message passing step of the belief propagation algorithm.

Four GPUs, each with 128 computing cores, were used to perform computations on the $160 \times 120 \times 15 \times 15$ lightfield shown in Figure 12, which is a portion of the lightfield in Figure 2 that guarantees an $800 \times 600$ pixel output. Table 1 shows speed up results obtained with up to 4 GPUs with respect to a sequential CPU implementation executed on an Intel Core 2 Quad at 2.5 GHz.

This sequential implementation executes on just one core and does not make use of special instruction set extensions, specific code optimizations, or external libraries. The program was compiled with GCC 4.1 setting the –O3 switch for code optimization. The multiGPU implementation uses a core with each GPU.

(a)

(b)

(c)

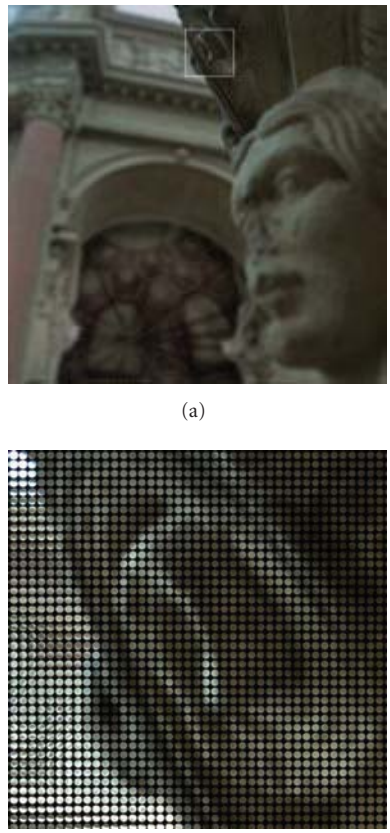
Figure 9: Nearest neighbor interpolation (a), bicubic interpolation (b), and super-resolved all-in-focus image (c).

Table 1: Frame rate and speedup over sequential implementation for up to 4 GPUs.

| Execution time on CPU (ms) | | | 4313 |
|---|---|---|---|
| Number of GPUs | Time (ms) | Frame rate (fps) | Speed up |
| 1 | 194.6 | 5.1 | 22.2 |
| 2 | 104.4 | 9.6 | 41.3 |
| 3 | 73.0 | 13.7 | 59.1 |
| 4 | 58.5 | 17.1 | 73.8 |

The results show that with four GPUs, near real-time performance can be achieved if we consider that the video frame rate is 24 frames per second.

Another feature to consider in parallel implementations is scalability. In the ideal case, the performance would increase linearly with the number of processors. However, in practice there is some overhead caused by data communication between processors (GPUs). It is important to study the effect caused by this overhead as the number of GPUs increases so as to decide on the optimum amount of GPUs. To show the scalability of the implementation, the frame rate with different numbers of GPUs was measured, as depicted in Figure 13. It shows that frame rate growth is quasilinear with respect to the amount of processors for up to four GPUs. However, as expected, the growth rate is smaller than in the ideal case. To increase the frame rate it would be more adequate to use GPUs with more processing cores and memory like nVidia Tesla because this requires less communications for the same processing power.



(a)

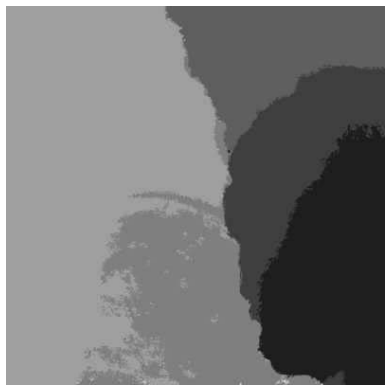
(b)

Figure 10: (a) Lightfield captured from a plenoptic camera. (b) Detail from the white square on the left image.

Performance also depends on the input size; so the frame rates necessary to generate output images with different spatial resolutions were measured and depicted in Figure 14 to show the achievable speed.

The theoretical complexity of the algorithm is $O(n^2 \times m^3 + n^2 \times m^3 \times BP\ iterations)$. Note that when $m$ and the number of *BP iterations* are fixed, the complexity is $O(n^2)$. In this case the algorithm is linear on the output size $n^2 \times m^2$ . This result also shows the feasibility of achieving satisfactory results for higher resolutions in the near future using more computing power.

(a)



(b)

Figure 11: Super-resolved all-in-focus image and depth map of lightfield in Figure 10.

Figure 12: Lightfield of $160 \times 120 \times 15 \times 15$ used for performance testing.

*4.3. Sensitivity to Parameter Values of the BP Algorithm.* The parameters of the belief propagation algorithm can have a significant influence on the final results. The number of iterations is imposed by real-time requirements and the value of $\lambda$ has to be adjusted to obtain satisfactory results. Smaller values of $\lambda$ lead to rough depth maps while higher values of $\lambda$ lead to smooth solutions.

Let us now consider the sensitivity of the algorithm to changes in $\lambda$. Since the real ground truth is unknown, the proportion of equal pixels between a chosen good quality reference depth map ($\lambda = 0.05$) and the depth maps obtained with different values of $\lambda$ is computed. Figure 15 shows how the parameter value influences the results obtained. Parameter changes in a small range around the selected optimal parameter do not generate significantly different results.

Figure 16 indicates the amount of depth values different from the chosen reference map for every depth map pixel. This allows us to see that border pixels are more influenced by the parameter.

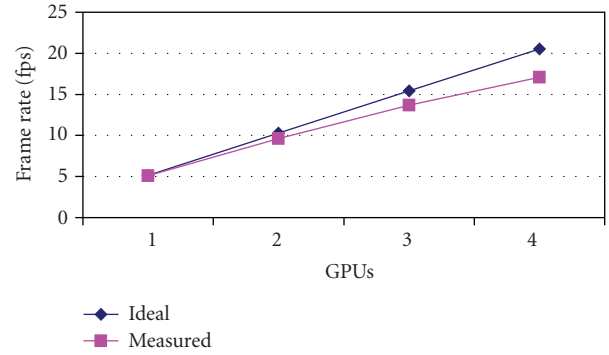



Figure 13: Ideal and measured speed up for up to 4 GPUs.

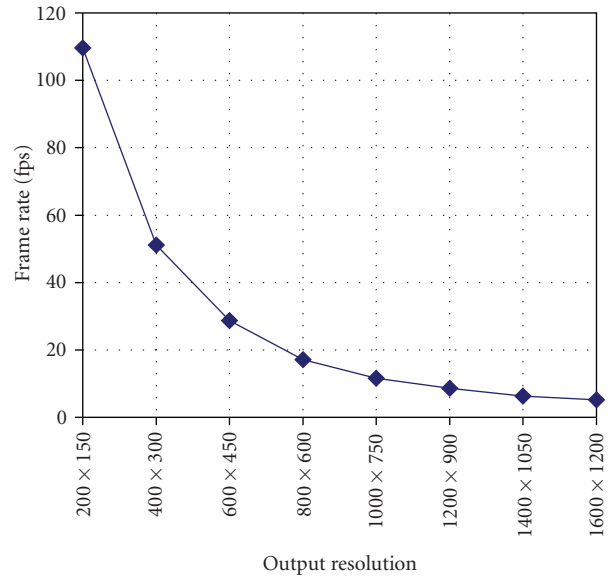



Figure 14: Frame rates obtained for different output resolutions.

## 5. Conclusions and Future Work

This paper presents a novel approach for obtaining a 3D reconstruction from a scene using a plenoptic video-camera. Simultaneous super-resolved depth maps and all-in-focus image estimation solve the spatial resolution drawback of previous techniques based on plenoptic cameras. Results for several lightfields are shown.

The problem of real-time execution is also addressed with the use of multiple GPUs. Near real time for a lightfield

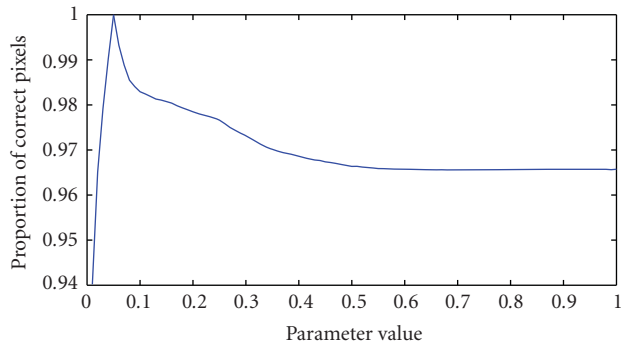



Figure 15: Proportion of correct pixels for different values of $\lambda$. Correctness was measured with respect to a reference depth map with good quality.

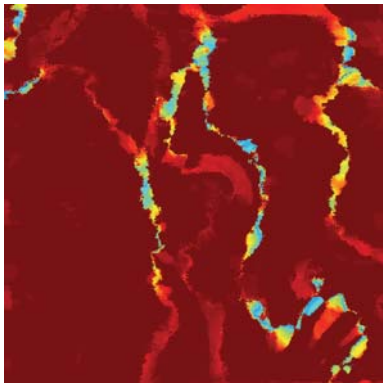


Figure 16: Number of depth values that are different from the chosen reference depth map when varying $\lambda$. Warm colors indicate an estimation independent of $\lambda$. Cold colors indicate more dependence.

of $160 \times 120 \times 15 \times 15$ is achieved. The resulting depth map and all-in-focus image are suitable as the input to a Philips WOVvx 20″ autostereoscopic display with a spatial resolution of 800×600. This shows that an adequate combination of processing algorithms with fast parallel hardware, like GPUs, makes it possible to develop a portable plenoptic video-camera with 2D color+depth output.

The use of plenoptic cameras for 3DTV has some benefits with respect to 3D capture systems based on dual stereo cameras since there is no need for geometric and color calibration or frame synchronization. Plenoptic cameras can also improve the postproduction process since special effects and synthetic content can be added easily to scenes because depths are known. This will make it easier to combine real and synthetic scenes. It should result in lower postproduction costs and time as well. There are also other features that make this format interesting for postproduction processes. Since the color image is fully focused, special effects may include defocusing on demand by blurring pixels with particular depth values. Other color effects can also be applied to particular depths in the scene (Figure 17).

Another advantage of depth range cameras is that their output can be seen as a compressed coding of 3D

Figure 17: Depth-based color manipulation of all-in-focus image.

information of a scene. Compared to other formats like dual stereoscopic video, where the amount of information with respect to conventional uncompressed 2D formats is doubled, the 2D color+depth format needs only 10%–20% more bit rate [4]. Notice that using a depth map and color image a stereo pair can be recomposed.

The use of compact designs like plenoptic video-cameras and adequate processing hardware enables the creation of passive portable 3D video capture devices. The creation of such devices will facilitate 3D content creation. Currently the lack of content is one of the main obstacles to the introduction of 3DTV in the home. Devices that generate contents for autostereoscopic displays in a simple way will help to mitigate this shortage.

Future extensions to our work will try to improve the processing speed and the super-resolution technique.

In order to increase the frame rate the use of CUDA 2.2 or higher will be considered. This new version of CUDA has some interesting features, like portable pinned memory and mapped memory. These can help to improve communication between GPUs. Also, for hardware with computing capability 1.2 the conditions for coalescent memory access are more flexible [22]. In addition, the OpenCL 1.0 framework will be an interesting candidate in the future, since it allows parallel computing on heterogeneous systems. Other improvement related with processing time will consist of porting this technique to FPGA (Field Programmable Gate Arrays) to obtain a low-cost processing device whose dimensions will allow it to be integrated as a image processing chip inside plenoptic cameras to encode the raw plenoptic image into a 2D color+depth format.

Improvements in the super-resolution algorithms will consist in the use of motion-based super-resolution techniques to achieve better resolution and to increase the robustness of depth maps and all-in-focus images. Finally, the depth resolution drawback will be addressed by using joint spatial-angular super-resolution techniques.

## Acknowledgments

This work was funded by the national R&D Program (Project DPI 2006-07906) of the Ministry of Science and Technology and by the European Regional Development Fund (ERDF).

## References


[1] L. Onural, "Television in 3-D: what are the prospects?" *Proceedings of the IEEE*, vol. 95, no. 6, pp. 1143–1145, 2007.

[2] C. Fehn, "3D TV broadcasting," in *3D Videocommunication*, O. Schreer, P. Kauff, and T. Sikora, Eds., John Wiley & Sons, Chichester, UK, 2005.

[3] L. M. J. Meesters, W. A. IJsselsteijn, and P. J. H. Seuntiens, "A survey of perceptual evaluations and requirements of three-dimensional TV," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 14, no. 3, pp. 381–391, 2004.

[4] A. Smolic, K. Mueller, N. Stefanoski, et al., "Coding algorithms for 3DTV—a survey," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 17, no. 11, pp. 1606–1620, 2007.

[5] G. B. Akar, A. M. Tekalp, C. Fehn, and M. R. Civanlar, "Transport methods in 3DTV—a survey," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 17, no. 11, pp. 1622–1630, 2007.

[6] M. Kawakita, K. Iizuka, T. Aida, et al., "Axi-vision camera: a three-dimensional camera," in *Three-Dimensional Image Capture and Applications III*, vol. 3958 of *Proceedings of SPIE*, pp. 61–70, San Jose, Calif, USA, January 2000.

[7] Z-cam, "3dv systems," http://www.3dvsystems.com/.

[8] F. P. Nava, J. P. Lüke, J. G. Marichal-Hernández, F. Rosa, and J. M. Rodríguez-Ramos, "A simulator for the cafadis real time 3DTV camera," in *Proceedings of 3DTV-Conference: The True Vision—Capture, Transmission and Display of 3D Video (3DTV-CON '08)*, pp. 321–324, Istanbul, Turkey, May 2008.

[9] The CAFADIS camera: international patent numbers, PCT/ES2007/000046 and ES2008/000126.

[10] P. Benzie, J. Watson, P. Surman, et al., "A survey of 3DTV displays: techniques and technologies," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 17, no. 11, pp. 1647–1657, 2007.

[11] Philips Wowx Display, http://www.businesssites.philips.com/3dsolutions/servicesupport/docs/docs.page.

[12] R. Ng, "Fourier slice photography," in *Proceedings of the International Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '05)*, pp. 735–744, Los Angeles, Calif, USA, 2005.

[13] F. Pérez, J. G. Marichal, and J. M. Rodríguez-Ramos, "The discrete focal stack transform," in *Proceedings of the 16th European Signal Processing Conference (EUSIPCO '08)*, Lausanne, Switzerland, August 2008.

[14] J. G. Marichal-Hernández, J. P. Lüke, F. Rosa, F. Pérez, and J. M. Rodríguez-Ramos, "Fast approximate focal stack transform," in *Proceedings of 3DTV-Conference: The True Vision—Capture, Transmission and Display of 3D Video (3DTV-CON '09)*, Potsdam, Germany, May 2009.

[15] F. Pérez and J. P. Lüke, "An $O(n^2 \log n)$ per plane fast discrete focal stack transform," in *Proceedings of the Optical 3D Measurement Techniques*, 2009.

[16] T. Georgiev, K. C. Zheng, B. Curless, D. Salesin, S. Nayar, and C. Intwala, "Spatio-angular resolution tradeoff in integral photography," in *Proceedings of the Eurographics Symposium on Rendering*, 2006.

[17] A. Lumsdaine and T. Georgiev, "Full resolution lightfield rendering," Tech. Rep., Adobe Systems, Inc., January 2008.

[18] F. Pérez and J. P. Lüke, "Simultaneous estimation of superresolved depth and all-in-focus images from a plenoptic camera," in *Proceedings of 3DTV-Conference: The True Vision—Capture, Transmission and Display of 3D Video (3DTV-CON '09)*, Potsdam, Germany, May 2009.

[19] J. D. Owens, D. Luebke, N. Govindaraju, et al., "A survey of general-purpose computation on graphics hardware," *Computer Graphics Forum*, vol. 26, no. 1, pp. 80–113, 2007.

[20] D. Scharstein and R. Szeliski, "A taxonomy and evaluation of dense two-frame stereo correspondence algorithms," *International Journal of Computer Vision*, vol. 47, no. 1–3, pp. 7–42, 2002.

[21] P. F. Felzenszwalb and D. P. Huttenlocher, "Efficient belief propagation for early vision," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR '04)*, vol. 1, pp. 261–268, Washington, DC, USA, July 2004.

[22] nVidia Corporation, "nVidia CUDA Programming Guide".

[23] Q. Yang, L. Wang, R. Yang, S. Wang, M. Liao, and D. Nistér, "Real-time global stereo matching using hierarchical belief propagation," in *Proceedings of the British Machine Vision Conference (BMVC '06)*, pp. 989–998, September 2006.

[24] A. Brunton, C. Shu, and G. Roth, "Belief propagation on the GPU for stereo vision," in *Proceedings of the 3rd Canadian Conference on Computer and Robot Vision (CRV '06)*, p. 76, June 2006.

[25] S. Grauer-Gray, C. Kambhamettu, and K. Palaniappan, "GPU implementation of belief propagation using CUDA for cloud tracking and reconstruction," in *Proceedings of IAPR Workshop on Pattern Recognition in Remote Sensing (PRRS '08)*, pp. 1–4, Tampa, Fla, USA, 2008.

Hindawi Publishing Corporation
International Journal of Digital Multimedia Broadcasting
Volume 2010, Article ID 857160, 8 pages
doi:10.1155/2010/857160

*Research Article*

# An Occlusion Approach with Consistency Constraint for Multiscopic Depth Extraction

**Cédric Niquin,[1,2] Stéphanie Prévost,[1] and Yannick Remion[1,2]**

[1] *CReSTIC SIC, Université de Reims Champagne-Ardenne, 51100 Reims, France*
[2] *3DTV Solutions, 8 rue Gabriel Voisin, 51100 Reims, France*

Correspondence should be addressed to Cédric Niquin, cedric.niquin@univ-reims.fr

Received 30 April 2009; Revised 31 August 2009; Accepted 28 November 2009

Academic Editor: Nikolaos Grammalidis



This is a new approach to handle occlusions in stereovision algorithms in the multiview context using images destined for autostereoscopic displays. It takes advantage of information from all views and ensures the consistency of their disparity maps. We demonstrate its application in a correlation-based method and a graphcuts-based method. The latter uses a new energy, which merges both dissimilarities and occlusions evaluations. We discuss the results on real and virtual images.

## 1. Introduction

Augmented reality has many applications in several domains such as games or medical training. On the other hand autostereoscopic display is an emergent technology, which adds a perception of depth enhancing the users immersion. Augmented reality can be applied to autostereoscopic display in a straightforward way by adding virtual objects on each image. However, it is much more interesting to use the depth-related information of the real scene so that virtual objects could be hidden by real ones.

To that end, we need to obtain one depth map for each view. The particular context of images destined to be viewed on autostereoscopic displays allows us to work on a simplified geometry (e.g., no rectification is needed, epipolar pairs are horizontal lines of the same rank, and disparity vectors are thus aligned along the abscissa). However, our aim is to obtain good assessment of depth in all kinds of scenes, without making any assumption on their contents. Indeed images may have homogeneous colors as well as they may have various colors. Also, due to the principle of autostereoscopic displays, the users can see two images at the same time. It is then crucial to have strongly consistent depth maps. For example if a virtual object is drawn in front of a real object in one view, it has to be drawn in the same order in all views. Therefore we introduce a new occlusion approach for multiview stereovision algorithms, which aims to ensure the consistency of the depth maps.

We propose an example of application of our approach in a correlation-based method and in a symmetrical graph-cuts-based method. Finally we discuss their results.

## 2. Related Work

Stereovision algorithms aim to find the disparity maps in order to deduce the depth maps. That is the reason why we will use the phrase "disparity maps" instead of "depth maps" in the following lines. Depth maps can be easily obtained from disparity maps using a triangulation step.

Let us admit we have a set of $N$ images, numbered from the left (0) to the right $(N-1)$, shot with a parallel capture system specifically designed for autostereoscopic displays. Figure 1 illustrates these shooting conditions with $N = 4$. $f$ is the focal distance and *dioc* is the Distance Intra Optical Centers. $I_i$ designates the set of pixels of the $i$th image. $d_i(p)$ is a function associating a disparity vector with any pixel $p$ from $I_i$. This vector is the difference between the coordinates of the corresponding pixel of $p$ in image $i+1$ and those of $p$ in image $i$. The corresponding pixel of $p$ in the next image is then at position $p + d_i(p)$. Since we are using a simplified geometry, $d_i(p)$ is of dimension one. It is an integer too

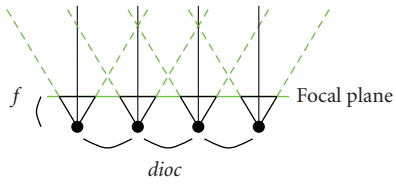



Figure 1: Illustration of the parallel camera configuration with four optics (illustrated by black dots).

($d_i(p) \in \mathbb{Z}$), so that we do not have to deal with subpixels. Moreover it is possible to find the corresponding pixel of $p$ in any image using the only disparity $d_i(p)$. In any $k$ image, the corresponding pixel is given by $p + (k - i)d_i(p)$. For example the corresponding pixel of $p$ in the previous image is $p - d_i(p)$.

Optical flow algorithms are based on a cost $L(p, q)$, which evaluates the color dissimilarity between two pixels $p$ and $q$. In the case of color images, it is given by

$$L(p,q) = \left|r_p - r_q\right| + \left|g_p - g_q\right| + \left|b_p - b_q\right|, \quad (1)$$

where $r_p$, $g_p$, and $b_p$ (resp., $r_p$, $g_p$, and $b_p$) are the red, green, and blue components of $p$ (resp., $q$). Several methods use the multiview aspect in order to make such algorithms more robust and less sensitive to noise. The local cost $M_\alpha^i$ of a pixel $p$ ($\in I_i$) according to disparity $\alpha$ is then given by

$$M_\alpha^i(p) = \sum_{k=0}^{N-2} L(p + (k - i)\alpha, p + (k + 1 - i)\alpha). \quad (2)$$

The reader can refer to Scharstein and Szeliski [1] for a complete taxonomy of dense stereovision algorithms. In many recent publications from this domain, authors use color segmentation in their methods [2–5]. However color segmentation and other primitive extraction methods are independent for each view. It is then impossible to ensure the consistency between the disparity maps. Moreover, it may be a cause of errors when applied to images with mainly homogeneous colors. Therefore these methods are incompatible with the objective presented previously, which imposes working in a local colorimetric context. We may not have any assumption about the content of the images and, thus, we cannot extract other features. So we aim to make up for this lack of information by taking advantage of redundancies in the $N$ images.

A lot of algorithms deal with occlusions in order to obtain better disparity maps, which preserve discontinuity at object's boundaries. The first step to deal with occlusions is to be able to detect them. Egnal and Wildes [6] compare five approaches. Some of them are based on the idea that color discontinuities correspond to depth discontinuities. These approaches are called photometry-based approaches. Alvarez et al. [7] use the gradient of the gray levels in order to locally adjust the smoothness constraint of their energy: the lower the value of the gradient, the stronger the smoothness constraint. On the other hand, geometry-based approaches use disparities in order to detect occlusion areas. The reader can refer to [6] for a complete comparison of such methods. We prefer geometry-based approaches to the photometry-based ones since they do not make any assumption on colors and allow disparity maps from the $N$ images to interact and to be linked. This link ensures the consistency of the disparities and makes up for the lack of information previously discussed. The most widely used geometry-based method is the Left Right Checking (LRC) approach [8–10]. The principle is that a pixel should match a pixel from another image with the same disparity; otherwise an occlusion occurs. In the case of two images (numbered 0 and 1), this can be expressed by

$$\begin{aligned} \epsilon_0(p) &= ||d_0(p) - d_1(p + d_0(p))||, \\ \epsilon_1(p) &= ||d_1(p) - d_0(p - d_1(p))||, \end{aligned} \quad (3)$$

where $\epsilon_i(p)$ is close to zero when there is no occlusion and high when pixel $p$ is occluded in image $i$. There have been several attempts to improve the robustness of this method [11, 12]. However, the original LRC is still the most popular approach [2, 5, 13, 14]. We propose an LRC-based approach, which differs in several points:

(i) it is extended to the multiview context,

(ii) it ensures a geometric consistency between depth maps.

After the detection step, the main difficulty is to handle occlusions in the matching algorithms. Woetzel and Koch [15] propose a correlation-based algorithm, which do not add up dissimilarity costs for the whole set of image pairs but for a subset of it. They replace (2) by

$$M_\alpha^i(p) = \sum_{(j,k)\in S} L(p + (j - i)\alpha, p + (k - i)\alpha), \quad (4)$$

where $S$ is the set of chosen image pairs. The authors propose two methods to choose the set $S$ of image pairs. The first one is to select the furthest left or the furthest right images. The second one is to select the pairs with the smallest $L$ costs. This method reduces the impact of occlusions on the results but introduces a lot of errors in the images which are the futhest away.

There are two categories of methods based on energy minimization performing the matching while taking occlusions into account.

The first category contains iterative methods [8, 10] based on Algorithm 1. In order to apply this principle, Proesmans et al. [8], who work with two images, use four maps, one disparity map plus one occlusion map per image. The occlusion maps are computed using the LRC approach. Strecha and Van Gool [10] extend this principle to the multiview context. First, $N$ disparity maps are computed from the $N$ views. Then for each view, $N - 1$ occlusion maps are computed as being the LRC evaluation with all other views.

In order to obtain better results, some methods start again at step 2 when step 3 is over and loop until the system converges. The problem of iterative methods is that

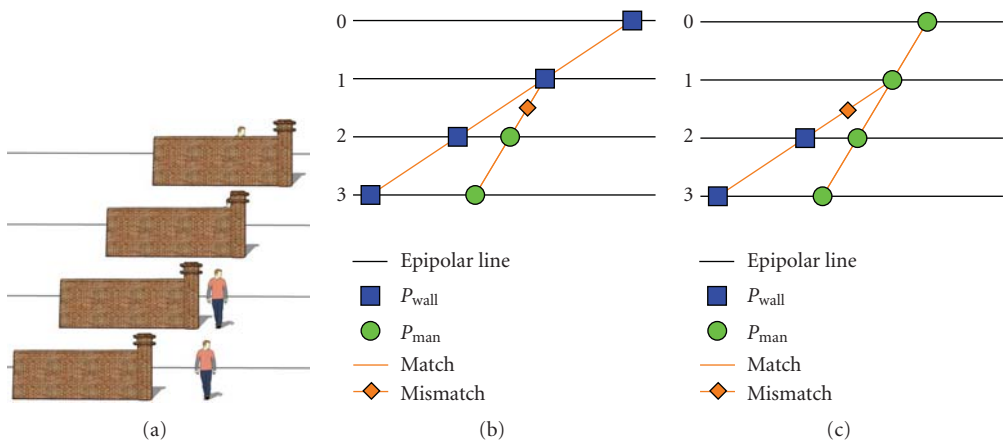



Figure 2: Examples of matching graphs.

(1) disparities estimation without taking into account occlusions
(2) estimation of occlusion areas
(3) disparities estimation with occlusion handling

Algorithm 1: Iterative methods principle.

disparities and occlusions estimations are independent, do not interact with each other and, thus, do not ensure a global geometric consistency.

The second category is then composed of methods to estimate occlusions and disparities simultaneously. In the context of two views, Alvarez et al. [9] introduce the following energy $E(d_0, d_1) = E_0(d_0, d_1) + E_1(d_0, d_1)$ with

$$\begin{aligned} E_0(d_0, d_1) &= E_0^d(d_0) + \chi E_0^s(d_0) + \beta E_0^o(d_0, d_1), \\ E_1(d_0, d_1) &= E_1^d(d_1) + \chi E_1^s(d_1) + \beta E_1^o(d_0, d_1), \end{aligned} \tag{5}$$

where $E_k^d$ evaluates dissimilarities between corresponding pixels, $E_k^s$ corresponds to the smoothness constraint, and $E_k^o$ is the sum of $\epsilon_k$ for every pixel of image $k$. $\chi$ and $\beta$ are weighting factors.

Note that even if pixels are detected as occluded, their dissimilarities are still taken into account in the dissimilarity term. That means that this term contains dissimilarities of mismatching pixels, which have nothing in common. This is a problem since that introduces noise into the energy. In order to solve that, Ince and Konrad [13] use an energy similar to (5) with the dissimilarity terms of the form

$$\begin{aligned} E_0^d(d_0) &= \sum_{p \in I_0} F(\epsilon_0(p)) \times L(p, p + d_0(p)), \\ E_1^d(d_1) &= \sum_{p \in I_1} F(\epsilon_1(p)) \times L(p, p - d_1(p)), \end{aligned} \tag{6}$$

where $F$ is a weighting function, which approaches zero when $\epsilon$ is high (i.e., occlusions are detected). Moreover, they use the smoothness term in order to extrapolate the disparities in occluded areas.

By the same token, we have proposed a multiview graph-cuts-based method in [14], which integrates occlusion penalties in its energy function without integrating dissimilarities of mismatching pixels.

In spite of the fact that these methods use smooth and discontinuity preserving functions, they still can contain inconsistencies that we will detail in Section 3.

## 3. A New Approach for Occlusion Detection

*3.1. Overview.* Let us imagine a standard scene with a man behind a wall. Four views of this scene are shot; Figure 2(a) shows corresponding epipolar lines from each image and superimposes them. This representation, that we call matching graph, is useful in order to define matches and occlusions. In Figure 2(b), pixel $p_{man} \in I_3$ (green circle) has disparity $d_3(p_{man})$. The corresponding pixel in $I_2$ is also a pixel of the man, so there is a match between these pixels. In image 1, the corresponding pixel of $p_{man}$ is part of the wall (blue square). It has a larger disparity (since the wall is nearer). Therefore there is an occlusion between these two pixels, as shown in Figure 2(b) with an orange diamond. The scene described in Figure 2(c) is an example of an impossible matching graph. This graph shows that the wall pixel in image 2 corresponds to a pixel in image 1, which is not a pixel of the wall but a representation of the man. It has a smaller disparity. This situation is impossible as it supposes that the wall is hidden by the man whose disparity implies that he is behind it. This is what we call a consistency error. The LRC does not handle this kind of inconsistencies.

We propose the following rules in order to define our approach of occlusions. Let us assume $p \in I_i$ and its corresponding pixel $q \in I_j$:

(i) if $d_i(p) = d_j(q)$, $p$ and $q$ match,

(ii) if $d_i(p) < d_j(q)$, $p$ is occluded,

(iii) if $d_i(p) > d_j(q)$, there is a consistency error.

In order to simplify writing, we call $\theta$ the occlusion image when an occlusion occurs at image number $\theta$, that is, when corresponding pixels from images $\theta$ and $\theta + 1$ do not match. In Figure 2(b), for instance, there is an occlusion between images 1 and 2, $\theta$ is then equal to 1.

*3.2. Energy Function.* In order to take the rules presented above into account, we use an energy function of the form

$$E(d) = E^m(d) + \sum_{i=0}^{N-1} E_i^s(d_i), \tag{7}$$

where $d$ is the set of all disparity functions $d_i$. $E_i^s$ is the smoothness term, which may be the same as the one used in (5). This smoothness constraint is applied to each disparity function. $E^m$ contains all the dissimilarity, occlusion, and consistency penalties.

In the case of Figure 2(b), $E^m$ will include the three dissimilarities between the four pixels of the wall, plus one dissimilarity between the two pixels of the man, plus one occlusion penalty between the two mismatching pixels. In Figure 2(c), $E^m$ is the sum of the three dissimilarities between the man pixels, the dissimilarity between the wall pixels, and the consistency penalty. Of course these examples take a very small number of pixels into account whereas $E^m$ considers all of them.

Finally, this term is given by

$$\begin{aligned} E^m(d) = &\sum_{i=0}^{N-2} \sum_{p \in I_i} E_{i+}^{\text{local}}(d, p, p + d_i(p)) \\ &+ \sum_{i=1}^{N-1} \sum_{p \in I_i} E_{i-}^{\text{local}}(d, p, p - d_i(p)), \end{aligned} \tag{8}$$

where $E^{\text{local}}$ is the local cost between $p \in I_i$ and its corresponding pixel either in the previous $(i-)$ or the next $(i+)$ image. Let us call this pixel $q$. We have

$$E_{i+}^{\text{local}}(d, p, q) = \begin{cases} L(p, q) & \text{if } d_i(p) = d_{i+1}(q), \\ K^{\text{occ}} & \text{if } d_i(p) < d_{i+1}(q), \\ K^{\text{con}} & \text{if } d_i(p) > d_{i+1}(q). \end{cases} \tag{9}$$

The term $E_{i-}^{\text{local}}(d, p, q)$ is the same except that $q$ is in image $i - 1$ instead of $i + 1$. $K^{\text{occ}}$ and $K^{\text{con}}$ are two constant values corresponding to the occlusion penalty and the consistency penalty, respectively. Due to our specific domain of application (augmented reality on autostereoscopic displays), the consistency constraint must be very strong and the value of $K^{\text{con}}$ is then very high to ensure that this case will never happen.

This energy function can be used in different methods as we will see in Section 4.

## 4. Application

In this section, we present two applications of our occlusion approach. The first one does not use any smoothness constraint and focuses on our approach of occlusions in order to emphasize its relevance on a correlation-based method. The second one is an application of the energy function as defined in the previous section on a graph-cuts-based method.

Both methods use the same constant $K^{\text{occ}}$. We found empirically that a value of 100 gives good results with our different sets. We give $K^{\text{con}}$ a value of 3000.

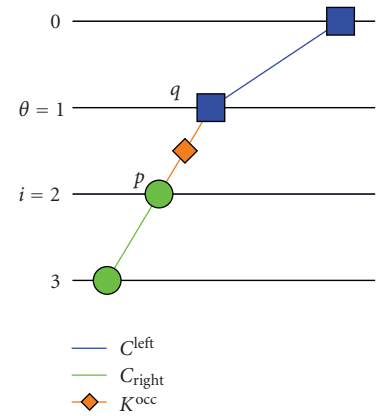



Figure 3: Example of the components of $C^{\text{local}}$ (10).

*4.1. Correlation-Based Method.* This method uses two distinct local costs. The first one supposes there is no occlusion and the second one supposes there is exactly one occlusion. These two costs are in competition by means of a Winner Takes All (WTA) algorithm.

The first cost could be any local cost as found in the literature. Our implementation uses cost $M_\alpha^i$ described in (2) where $L$ is the absolute difference of intensities summed over the three color components.

The second cost $C^{\text{local}}$ is a local subset of $E^m$ (8). Indeed the cost of a $p$ pixel includes the $E^{\text{local}}$ (9) energies for all pixels linked to $p$ in the $N$ images, assuming that there is only one occlusion and two disparities (on the left and the right of the occlusion). In order to ensure the consistency of the matching graph implicitly, only values meeting the consistency condition are tested. Therefore the constant $K^{\text{con}}$ never appears. $C^{\text{local}}$ finally contains one penalty $K^{\text{occ}}$ due to the occlusion, plus dissimilarities between matching pixels. The local cost for $p$ in image $i$ is

$$\begin{aligned} &C_i^{\text{local}}(p, \theta, \delta_{\theta-}, \delta_{\theta+}) \\ &\quad = C_i^{\text{left}}(q, \theta, \delta_{\theta-}) + C_i^{\text{right}}(q, \theta, \delta_{\theta+}) + K^{\text{occ}}, \end{aligned} \tag{10}$$

where $\theta$ is the occlusion image and $\delta_{\theta-}$ and $\delta_{\theta+}$ are two disparities, respectively, on the left and on the right of the occlusion. Figure 3 shows the three terms of $C^{\text{local}}$ with $i = 2$ and $\theta = 1$ with the same configuration as in Figure 2(b), that is, it shows the costs of each term. $q$ is the corresponding pixel of $p$ in image $\theta$: $q$ is equal to $p + (\theta - i)\delta_{\theta-}$ if $i$ is on the left of the occlusion ($i < \theta$), and equal to $p + (\theta - i)\delta_{\theta+}$ if it is on the right ($i > \theta$). $C^{\text{left}}$ and $C^{\text{right}}$ contain dissimilarity costs on

the left and on the right sides of the occlusion, respectively. They are given by

$$C_i^{\text{left}}(q,\theta,\delta_{\theta-}) = \sum_{k=0}^{\theta-1} L(q+(k-\theta)\delta_{\theta-}, q+(k+1-\theta)\delta_{\theta-}),$$
$$C_i^{\text{right}}(q,\theta,\delta_{\theta+}) = \sum_{k=\theta+1}^{N-2} L(q+(k-\theta)\delta_{\theta+}, q+(k+1-\theta)\delta_{\theta+}). \tag{11}$$

In order to ensure the consistency of the matching graph, only disparities meeting the following condition are tested:

$$\begin{aligned} \delta_{\theta-} &< \delta_{\theta+} \quad \text{if } i < \theta, \\ \delta_{\theta-} &> \delta_{\theta+} \quad \text{if } i > \theta. \end{aligned} \tag{12}$$

Finally, the selection is based on a WTA algorithm: if the minimum cost for a pixel is obtained using $M_\alpha^i$, then the disparity $\alpha$ is assigned to the pixel. If it is obtained using $C^{\text{local}}$ then the disparity assigned to the pixel is either $\delta_{\theta-}$ or $\delta_{\theta+}$, depending on whether $i$ is, respectively, on the left or on the right of the occlusion.

*4.2. Graph-Cuts-Based Method.* Our method is based on the energy function previously described in (7) and (8). We use the graph-cuts method in order to minimize our energy. Please refer to publications by Boykov et al. [16] for a complete presentation of the graph-cuts method and by Kolmogorov and Zabih [17] for an explanation of the graph construction. We use the *α-expansion* algorithm and, unlike others, we loop only once for each disparity. Moreover we always loop from the highest disparity to the lowest one, since we have found that this gives more accurate occlusion areas in the results. The graph we present in this section is based on this assumption, as we will see further. Our graph is composed of one node for each pixel from all the images. It also has a source ($s$) and a sink ($t$), which mean "keep the same disparity" and "change to disparity $\alpha$", respectively.

Now, we will see how to construct the graph corresponding to $E^{\text{local}}$ (9). In fact $p$ will not have the same corresponding pixel in the other image whether it changes its disparity or not. The graph corresponding to $E^{\text{local}}$ is then composed of three nodes $p$, $q_s$, and $q_t$, which are the corresponding pixels if $p$ is cut from the source or from the sink. However, it can be separated into two graphs using pairs $(p, q_s)$ and $(p, q_t)$. Figure 4 shows the general graph corresponding to $E^{\text{local}}$.

(i) *Pair* $(p, q_s)$
This pair makes sense only if $p$ is cut from the source. It means that cuts $C_{ts}^{pq_s}$ and $C_{tt}^{pq_s}$ are equal to 0. Cut $C_{ss}^{pq_s}$ is equal to either $L(p,q_s)$ or $K^{\text{occ}}$ depending on the previous disparities of $p$ and $q_s$. Since our main loop is going from the highest disparity to the lowest one, cut $C_{st}^{pq_s}$ means $q_s$ will have a lower disparity than $p$. It is then equal to $K^{\text{con}}$. As mentioned by Kolmogorov and Zabih [17], the following constraint has to be checked in order to ensure that the graph is achievable:

$$C_{st} + C_{ts} \geq C_{ss} + C_{tt}. \tag{13}$$

In our case $C_{ts}^{pq_s}$ and $C_{tt}^{pq_s}$ are equal to 0 and $C_{st}^{pq_s}$ is always greater than $C_{ss}^{pq_s}$, so that our graph is achievable.

(ii) *Pair* $(p, q_t)$
On the same principle as the previous pair, we have $C_{ss}^{pq_t}$ and $C_{st}^{pq_t}$ equal to 0. $C_{tt}^{pq_t}$ is equal to dissimilarity penalty $L(p,q_t)$ and $C_{ts}^{pq_t}$ is equal to the occlusion penalty. Note that the value of $K^{\text{occ}}$ has to be greater or equal to $L(p,q_t)$ in order to respect the constraint given by (13). To that end, we truncate $L(p,q_t)$ to the value of $K^{\text{occ}}$.

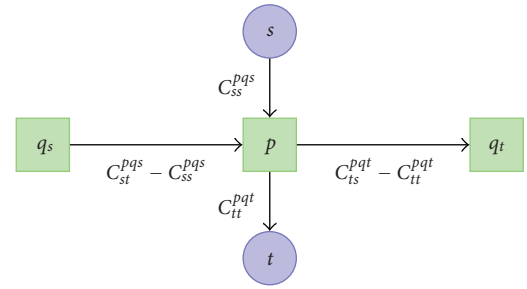



FIGURE 4: Graph corresponding to $E^{\text{local}}$ (9).

The smoothness of the result is ensured by term $E^s$ of the energy given in (7). We use the following definition:

$$E_i^s(d_i) = \sum_{(u,v)\in\Psi_i} \|d_i(u) - d_i(v)\|, \tag{14}$$

where $\Psi_i$ is the set of neighbour pixels pairs in image $i$. We use two implementations of the smoothness constraint. In the first one $\Psi_i$ contains only horizontal neighbours in order to obtain a 1D smoothness constraint. Such a constraint makes for an independent selection of epipolar lines and then a parallel implementation. The second one uses a 2D smoothness constraint and includes both horizontal and vertical neighbours.

## 5. Results

To compare our methods, first between them and secondly with other existing ones, we use two sets of 8 images. The first one is a set of images of a virtual scene, which allows us to compare results against ground truth. The second one is a set of photographies taken at Palais du Tau in Reims [18]. The dimensions of images in both sets are $512 \times 384$ pixels. Figure 5 shows one image of each set. The photography has homogeneous colors whereas the virtual scene has various colors. This allows us to test our methods in both cases.

We compare three pairs of methods. The first pair is composed of correlation-based methods. One uses the cost of (2) and the other is our own correlation-based method.

(a) (b)

Figure 5: A photography shot at Palais du Tau (a) and a virtual scene (b).

Table 1: Errors with ground truth on the virtual scene plus computation times on both Palais du Tau and virtual sets.

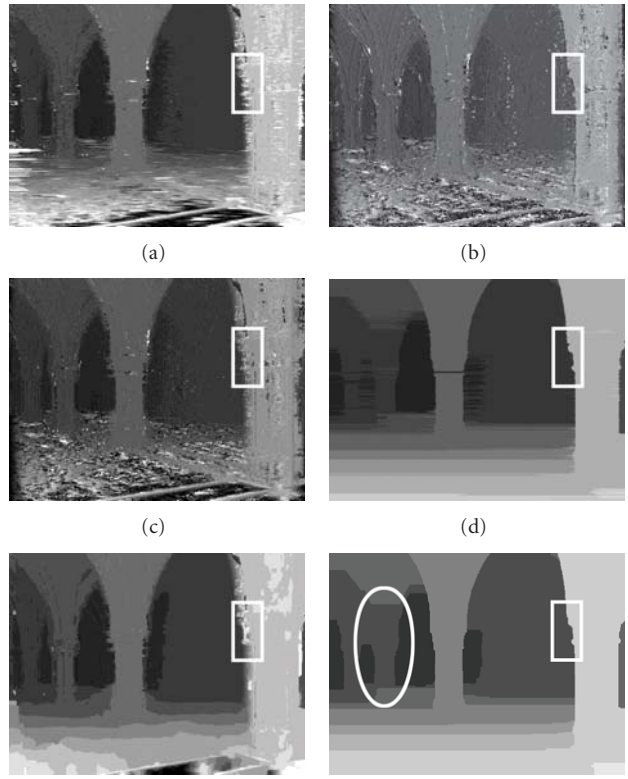
| Method | Occlusion | Error | Times (s) | |
|---|---|---|---|---|
| | | | Tau | Virtual |
| Correlation | without | 0.644 | 0.23 | 0.25 |
| | with | 0.630 | 1.25 | 1.24 |
| 1D smoothness | without | 0.718 | 6.88 | 6.02 |
| | with | 0.572 | 17.89 | 12.39 |
| 2D smoothness | without | 0.692 | 12.02 | 8.98 |
| | with | 0.543 | 38.13 | 18.36 |

The second and third pairs are graph-cuts based methods. One with a smoothness constraint along epipolar lines, and one with a 2D smoothness constraint. Each pair is composed of a method using our energy function and one using a standard energy of the form

$$E_0(d_0) = E_0^d(d_0) + \chi E_0^s(d_0). \tag{15}$$

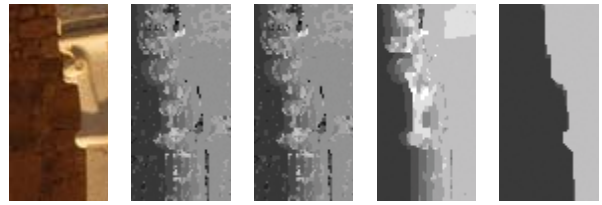
Using the ground truth of the virtual scene, we give the error rate corresponding to each method in Table 1. This error results from the absolute differences between real disparities and our disparities summed on all pixels and divided by the number of pixels. The result is then the average disparity error per pixel. Our local methods have the highest error rate, and our global method has the smallest error rate. For each category, the error rate of the method using our occlusion approach is always lower than the method which does not use it. We observe that the error rate of the correlation-based method without occlusion handling is lower than the one of the method using 1D smoothness constraint. We think that is due to the fact that our virtual images contain no noise and no specular light at all. That is the reason why the correlation-based method gives particularly good results in this case. Therefore, the comparison between correlation-based methods and energy minimization based methods is meaningless.

Table 1 also gives computation times on both sets of images. We used an Intel Core 2 Duo CPU E4700 and 2 Go of memory. Both correlation-based methods are implemented using CUDA on an NVIDIA Quadro FX 3700 graphic card. Times include the computation of the whole set of the $N$ disparity maps. Globally, methods using the occlusion approach are slower than the other methods. This is due to the fact that they have more possibilities to take into account. Indeed, the correlation-based method has more tests to carry out, and the graph-cuts based method has a more complicated graph structure, that is slower to solve.



(a) (b) (c) (d) (e) (f)

Figure 6: Disparity maps obtained using methods based on: correlation (a),(b), horizontal smooth constraint (c),(d), and 2D smooth constraint (e),(f). Images (a),(c),(e) and (b),(d),(f) are, respectively, computed without and with our occlusion approach.



(a) (b) (c) (d) (e)

Figure 7: Extracts of Figures 5, 6(a), 6(b), 6(e), and 6(f).

Figure 6 shows results obtained without and with occlusion handling ($K^{\text{occ}} = 100$). Figure 6(a) illustrates the standard method; whereas Figure 6(b) corresponds to the method we have presented. We can see in these images that our method has precisely detected occlusions on boundaries of the columns. However, areas without depth discontinuities like the background contain errors. We think that it is due to the noise sensitivity of our occlusion detection. The first three extracts given in Figures 7(a), 7(b), and 7(c) show the details of these results on the front column. We observe that our correlation-based method has accurately defined

discontinuities fitting the real boundaries of the column, but disparities in occluded areas contain errors (noise in Figure 7(c)), since the method has difficulties to find them in such areas. Finally, our approach is not well suited to the principle of correlation-based methods. In fact, in such methods the selection of disparities is independent for each pixel, and the consistency from one disparity map to another cannot be ensured.

On the other hand, graph-cuts based methods allow the symmetrical minimization of energy, ensuring a strong consistency. Figures 6(c) and 6(d) show results with the 1D smoothness constraint, and Figures 6(e) and 6(f) show results with the 2D smoothness constraint. Again, the method using our occlusion approach has very accurate depth discontinuities at object boundaries. In our 1D smoothness method (Figure 6(d)), the horizontal line effect is more visible than in the classical method (Figure 6(c)). This is due to the fact that our occlusion approach penalizes any disparity variation, because it is detected as an occlusion. Our method tends to add a strong smoothness constraint along epipolar lines. The 2D smoothness constraint allows compensating for this artifact. The images in Figures 7(d) and 7(e) show the front column obtained with these methods. We notice that without occlusion handling the method cannot find the disparities of some pixels, which are not visible in all images. On the other hand, it accurately detects the occlusion using our occlusion approach. However, we observe in Figure 6(f) that the column in the background (in a white ellipse) is not well defined. This is due to the principle of plane-sweeping algorithms and to the fact that this column is actually between two planes. The reader can refer to [14] for a presentation of our refinement step, which solves this problem.

## 6. Conclusion

We have introduced a new approach in order to handle occlusions of a scene in a multiview context. As a proof of the relevance of this new detection rule, we have presented two methods with the particularity of handling objects boundaries very accurately. Even if these methods can handle two-view stereovision, they are designed for the multiview context with any number of views. The results we obtain show that our occlusion approach succeeds in detecting objects boundaries to the detriment of computation times, and can still create disparities even for pixels that are not visible in all views. Moreover, used on symmetrical energy minimization-based methods, our approach ensures a geometric consistency, which is crucial for autostereoscopic displays. However, computation time is the main problem of our methods. That is the reason why our objective is to find a means to minimize energy faster. One idea is the GPU implementation of the graph cuts. Some work has already been done in this domain [19, 20] but we are not using it for the moment since it induces a lot of constraints on the graph structure and must be adapted to our energy. Another possiblity that we are working on is to reduce the number of nodes used in our graph, in order to simplify the maximum flow problem.

## Acknowledgments

The work presented in this paper was supported in part by the "Agence Nationale de la Recherche" as part of the CamRelief project. This project is a collaboration between the University of Reims Champagne-Ardenne and 3DTV Solutions. The authors wish to thank Didier Debons, Michel Frichet, and Florence Debons for their contribution to the project.

## References

[1] D. Scharstein and R. Szeliski, "A taxonomy and evaluation of dense two-frame stereo correspondence algorithms," *International Journal of Computer Vision*, vol. 47, no. 1–3, pp. 7–42, 2002.

[2] A. Klaus, M. Sormann, and K. Karner, "Segment-based stereo matching using belief propagation and a self-adapting dissimilarity measure," in *Proceedings of the 18th International Conference on Pattern Recognition (ICPR '06)*, vol. 3, pp. 15–18, Hong Kong, August 2006.

[3] Z.-F. Wang and Z.-G. Zheng, "A region based stereo matching algorithm using cooperative optimization," in *Proceedings of the 26th IEEE Conference on Computer Vision and Pattern Recognition (CVPR '08)*, pp. 1–8, Anchorage, Alaska, USA, June 2008.

[4] G. Zhang, J. Jia, T.-T. Wong, and H. Bao, "Recovering consistent video depth maps via bundle optimization," in *Proceedings of the 26th IEEE Conference on Computer Vision and Pattern Recognition (CVPR '08)*, pp. 1–8, Anchorage, Alaska, USA, June 2008.

[5] Q. Yang, L. Wang, R. Yang, H. Stewénius, and D. Nistér, "Stereo matching with color-weighted correlation, hierarchical belief propagation, and occlusion handling," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 31, no. 3, pp. 492–504, 2009.

[6] G. Egnal and R. P. Wildes, "Detecting binocular half-occlusions: empirical comparisons of five approaches," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 24, no. 8, pp. 1127–1133, 2002.

[7] L. Alvarez, R. Deriche, J. Sánchez, and J. Weickert, "Dense disparity map estimation respecting image discontinuities: a PDE and scale-space based approach," *Journal of Visual Communication and Image Representation*, vol. 13, no. 1-2, pp. 3–21, 2002.

[8] M. Proesmans, L. van Gool, E. Pauwels, and A. Oosterlinck, "Determination of optical flow and its discontinuities using non-linear diffusion," in *Proceedings of the 3rd European Conference on Computer Vision (ECCV '94)*, vol. 2, pp. 295–304, Springer, Secaucus, NJ, USA, 1994.

[9] L. Alvarez, R. Deriche, T. Papadopoulo, and J. Sánchez, "Symmetrical dense optical flow estimation with occlusions detection," in *Proceedings of the 7th European Conference on Computer Vision-Part I (ECCV '02)*, pp. 721–735, Springer, London, UK, 2002.

[10] C. Strecha and L. Van Gool, "PDE-based multi-view depth estimation," in *Proceedings of the 1st International Symposium on 3D Data Processing Visualization and Transmission (3DPVT '02)*, pp. 416–425, 2002.

[11] S. Ince and J. Konrad, "Geometry-based estimation of occlusions from video frame pairs," in *Proceedings of IEEE International Conference on Acoustics, Speech and Signal Processing*

*(ICASSP '05)*, vol. 2, pp. 933–936, Philadelphia, Pa, USA, March 2005.

[12] P.-M. Jodoin, C. Rosenberger, and M. Mignotte, "Detecting half-occlusion with a fast region-based fusion procedure," in *Proceedings of the British Machine Vision Conference*, pp. 417–426, 2006.

[13] S. Ince and J. Konrad, "Occlusion-aware optical flow estimation," *IEEE Transactions on Image Processing*, vol. 17, no. 8, pp. 1443–1451, 2008.

[14] C. Niquin, S. Prévost, and Y. Remion, "Accurate multi-view depth reconstruction with occlusions handling," in *Proceedings of the 3rd 3DTV-Conference on the True Vision-Capture, Transmission and Display of 3D Video (3DTV-CON '09)*, Potsdam, Germany, May 2009.

[15] J. Woetzel and R. Koch, "Real-time multi-stereo depth estimation on GPU with approximative discontinuity handling," in *Proceedings of the 1st European Conference on Visual Media Production (CVMP '04)*, pp. 245–254, London, UK, March 2004.

[16] Y. Boykov, O. Veksler, and R. Zabih, "Fast approximate energy minimization via graph cuts," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 23, no. 11, pp. 1222–1239, 2001.

[17] V. Kolmogorov and R. Zabih, "What energy functions can be minimized via graph cuts?" *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 26, no. 2, pp. 147–159, 2004.

[18] http://palais-tau.monuments-nationaux.fr.

[19] A. Bhusnurmath and C. J. Taylor, "Graph cuts via $\ell_1$ norm minimization," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 30, no. 10, pp. 1866–1871, 2008.

[20] V. Vineet and P. J. Narayanan, "CUBA cuts: fast graph cuts on the GPU," in *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops (CVPR '08)*, pp. 1–8, Anchorage, Alaska, USA, June 2008.

Hindawi Publishing Corporation
International Journal of Digital Multimedia Broadcasting
Volume 2010, Article ID 924091, 11 pages
doi:10.1155/2010/924091

*Research Article*

# A Compact Representation for 3D Animation Using Octrees and Affine Transformations

**Youyou Wang and Guilherme N. DeSouza**

*Department of Electrical and Computer Engineering, University of Missouri, Columbia, MO 65211, USA*

Correspondence should be addressed to Guilherme N. DeSouza, desouzag@missouri.edu

Received 2 May 2009; Revised 30 September 2009; Accepted 8 December 2009

Academic Editor: Georgios Triantafyllidis



This paper presents a new and compact 3D representation for nonrigid objects using the motion vectors between two consecutive frames. Our method relies on an Octree to recursively partition the object into smaller parts. Each part is then assigned a small number of motion parameters that can accurately represent that portion of the object. Finally, an adaptive thresholding, a singular value decomposition for dealing with singularities, and a quantization and arithmetic coding further enhance our proposed method by increasing the compression while maintaining very good signal-noise ratio. Compared to other methods that use tri-linear interpolation, Principle Component Analysis (PCA), or non-rigid partitioning (e.g., FAMC) our algorithm combines the best attributes in most of them. For example, it can be carried out on a frame-to-frame basis, rather than over long sequences, but it is also much easier to compute. In fact, we demonstrate a computation complexity of $\Theta(n^2)$ for our method, while some of these methods can reach complexities of $O(n^3)$ and worse. Finally, as the result section demonstrates, the proposed improvements do not sacrifice performance since our method has a better or at least very similar performance in terms of compression ratio and PSNR.

## 1. Introduction

As the technology for graphics processing advances, so does the details in the 3D models used for animation. So, despite these advances, when storing, transmitting, or rendering such models, the need for fast and compact representations is the same as it was a few years ago. In that sense, two categories of systems can be found in the literature: the time-independent methods and the time-dependent methods.

The first and most traditional category is the time-independent, and it was first introduced in [1]. In that case, a 3D object is represented using its geometric properties at that moment. That is, 3D points [2, 3], triangular meshes, surface normals, edge orientations [4–7], wavelet coefficients [8, 9], and other features of the object are analyzed within a single time instant, or frame. These methods have advantages in terms of the quality of the model, but they are of course not very compact. On the other hand, time-dependent methods exploit the temporal relationship of these same types of features in order to increase compression. In one of the first systems to be reported [10], but also in more recent ones [11–15], the basic idea is to encode the motion of the 3D object. That is, the difference between consecutive frames, rather than the properties of the object in the frames. In order to do that efficiently, the system must identify the parts of the object that are stationary from the parts that are moving, and describe only the latter.

This type of approach raises two major problems: (1) how to partition the two portions of the object—that is, moving and stationary portions; (2) how to describe the moving portions so they represent as perfectly as possible the non-rigid motion. Although much research has been done in this area [11–13, 16, 17], these approaches still suffer from the following: (1) inaccurate motion transformations [17], (2) the need for extra space to store the partitioning [12], (3) the need for prior information on the entire sequence [13–15], and so forth.

In this paper, we address the above problems by proposing a rigid partitioning of the 3D space combined with an affine transformation for motion capture. Some of the major

advantages of our method are its computational efficiency, the compactness of the motion, and the space representation.

In Section 2, we will talk about some of the related work in compression of animated sequences. Here, we distinguish animation from real 3D data in particular for the fact that animation can rely on feature correspondence between frames. Section 3 contains the details of our approach and in Section 4, we compare our method against other celebrated methods in the literature. We show the advantages of using our representation for compression, but we also point out where it can be improved.

## 2. Related Work

One of the first methods proposed for time-dependent 3D data compression can be found in [10]. From this paper, a new paradigm in time-dependent compression was established: represent successive frames by a small number of motion parameters and select a coding scheme to efficiently store the data. Even though many systems today offer completely different approaches to capture and parametrize this motion—from affine transformations [14, 15, 18], to principal components [13, 19, 20], and to wavelets that analyze the parametric coherence in the sequence [21, 22]—most time-dependent methods generally perform two basic steps: (1) partitioning of complex objects into smaller and simpler components; (2) description of the motion of these same components.

With regard to the partitioning method, we find systems using regular or rigid spatial partitioning, where vertices are divided according to their spatial location. One such example is the Octree [11, 16, 17, 23]. In this case, the space is recursively divided into 8 equal portions until some termination criteria stop the process.

On the other side of the coin, we find systems employing irregular partitioning. In [12], for example, the system employed the Iterative Closest Points algorithm (ICP) and assumed that the underlying motion between two consecutive frames followed a rigid transformation. The rigid transformation returned from the ICP was used to reconstruct the frame, while small and irregular portions of the object with small reconstruction error were clustered together to form a single rigid component. In [24], the clustering of the vertices was based on a method similar to a k-means, while the distance between clusters was defined as the Euclidean distance on the subspace defined by a principal component analysis (PCA). That means that the entire sequence had to be known beforehand in order to calculate the subspaces. The same can be said about other PCA-based methods [13, 19]—whether using irregular partitioning or not. Finally, in [20], several local coordinate frames were assigned to the object at the center of each cluster, and the vertices were assigned to clusters depending on their movements between consecutive frames. If the type of objects is restricted to, for example, the human body, the body parts can be clustered using their trajectories, as it was done in [25]. Actually, there is a third kind of systems where a spatial partitioning is not at all explicit. That is, in [26, 27], for example, vertices are grouped despite their spatial location, but rather based on their motion vectors.

After a partitioning is obtained, the next step is to find an efficient encoding for the motion. In that sense, some partitioning methods impose constraints on how the motion can be described. In other cases, however, the partitioning is generic enough that different motion descriptors can be used. In [11, 16, 17], all versions of the system employed a tri-linear interpolation, a regular partitioning (octree), and eight motion vectors attached to the corners of the cell. In other cases [12], the authors proposed an irregular partitioning with an affine transformation between clusters as the motion descriptor. Finally, as we mentioned earlier, PCA-based methods can achieve a good compression by storing only the principal components of the motion vectors, that is, a smaller dimension than the original one. That can be done both globally [13, 19] or locally [20, 24], but in either case, the entire sequence must be available for the calculation of the principal components. More recent approaches include the Principal Geodesic Analysis, a variant of the PCA method [28]; methods relying on prediction of the motion vectors, [26]; the replica predictor [27].

Another method relying on irregular partitioning is the recent Frame-Animated Mesh Compression (FAMC) [14, 15, 18], which was standardized within the MPEG as part of MPEG-4 AFX Amendment 2 [29]. In this case, the partitioning is irregular, but fixed. That is, the partitioning does not follow a rigid structure as in an octree, but it must be decided at the very first frame based on all frames in the sequence. While this allows for a more compact representation, it still requires prior knowledge of the entire sequence.

## 3. Proposed Approach

In many applications involving virtual reality and computer graphics, such as for human-robot interaction, real-time processing is a requirement that cannot be undermined. With that application in mind, we devised a fast and compact representation for 3D animation using rigid partitioning of the space and an affine transformation. As we will show in Section 4, our claim is that such combination can produce results comparable to or better than other methods in terms of compression, while still preserving a good signal-to-noise ratio.

*3.1. Octree Structure.* Our approach starts with the use of octrees for the partitioning of 3D objects. Octrees have been used in computer vision and computer graphics for many years [30]. This data structure has also been widely used in both time-independent methods, such as [2, 3], as well as time-dependent methods, such as in [11] and later improved in [16, 17, 23].

In our case, the partitioning using octree is similar to that in other time-dependent methods, however, the decision as to when partition and the termination criteria are different, making our method unique. That is, with octrees, the 3D space containing the object vertices is recursively divided into

8 subspaces, also known as the octants, cells, or cubes. In this paper, we will use the terms cube, cell, node, and octant interchangeably.

The partitioning starts with the application of an affine transformation and the calculation of an error measurement based on the motion vectors. If this error is too high, the cell is subdivided and the process repeats for each subcell. As for the termination criteria, we propose an adaptive thresholding of the reconstruction error followed by a singular value decomposition and quantization using arithmetic coding to further increase the compactness of the representation. All this process is simplified by rescaling (normalizing) all vertices to a size between [0, 1]—that is, the size of the root cube is always regarded as 1 unit.

*3.2. Algorithm.* Our algorithm consists of an encoding of the motion vector of the current frame with respect to the previous one. That is, the algorithm perform the following steps.

(1) First, it applies a tightly bounded cube around all vertices in the previous frame.

(2) Next, it calculates the affine transformation matrix between all vertices in the bounding cube and the corresponding vertices from the current frame.

(3) It checks for singularities of the affine and then it quantizes and dequantize the resulting affine matrix. This step is required in order to produce the reconstructed current frame and to calculate the error between the reconstructed and actual current frames.

(4) If the error in the previous step is too large, the algorithm partitions the bounding cube into eight smaller subcubes and the steps (5) and (7) above are repeated for each of the subcubes.

(5) Otherwise, it stores the quantized affine transformation as the motion vector for that cube.

The steps above are highlighted by the blue box in Figure 1(a).

Once a representation for the current frame is completed, the algorithm proceeds to the next frame. That is, it now uses the reconstructed current frame as the "previous" frame and the next frame as the "current" frame and steps are repeated until the last frame in the sequence is encoded. The idea is that only the positions of the vertices for the first frame are recorded and transmitted to the other side—in the case of 3D video streaming, for example, when frames are generated on one machine and rendered on another machine. After the first frame is transmitted, only motion vectors related to each cube of the octree are transmitted to the receiving end. In practice, in order to achieve better signal-to-noise ratios, intra frames could be inserted after an arbitrary number of frames to *reset* the error. However, in this paper we are interested in maximum compression only, and therefore, we will not offer any further discussion on how or when to insert intra frames.

The "dual" of the *encoding* algorithm described above is the *decoding* algorithm, and it is presented in Figure 1(b). Since this algorithm consists of the same (dual) parts of the steps of the encoder, we will leave to the reader to explore the details of the decoder.

*3.3. Computation of the Affine Transformation.* One of the main steps in our approach is the calculation of a single motion vector that will describe the movement of all the vertices in the cube with respect to two consecutive frames. Since the correspondence between vertices from two different frames is known, this motion vector can be approximated using an affine transformation $A$ whose reconstruction error can be expressed as

$$E = \sum_{i=1}^{N} \left\| A * \vec{p}_i - \vec{q}_i \right\|^2, \tag{1}$$

where $N$ is the total number of vertices in the cube, and $\vec{p}_i$ is a 4 by 1 homogeneous vector with the coordinate of vertex $i$ in the previous frame. Similarly, $\vec{q}_i$ is the homogeneous coordinates of the corresponding vertex in the current frame. In other words, the affine transformation $A$ is the actual motion vector between the vertices of a cube in the previous frame, and the corresponding vertices of the current frame.

Considering the entire structure of the octree, the total reconstruction error is the sum of all the errors at the leaf nodes of the tree. That is

$$\mathbf{E} = E_1 + E_2 + \cdots + E_M = \sum_{j=1}^{M} \left( \sum_{i=1}^{N_j} \left\| A_j * \overrightarrow{p_{d_{ji}}} - \overrightarrow{q_{d_{ji}}} \right\|^2 \right), \tag{2}$$

where $M$ is the number of leaf nodes, $N_j$ is the number of vertices in the $j$th leaf node, and $d_{ji}$ is the index of the $i$th vertex in that same leaf node.

In vector form, the homogeneous coordinates of the points in the leaf node $j$, at the previous frame $f-1$, are given by

$$F_{j(f-1)} = \begin{bmatrix} p_{1d_{j1}} & p_{1d_{j2}} & \cdots & p_{1d_{jN_j}} \\ p_{2d_{j1}} & p_{2d_{j2}} & \cdots & p_{2d_{jN_j}} \\ p_{3d_{j1}} & p_{3d_{j2}} & \cdots & p_{3d_{jN_j}} \\ 1 & 1 & \cdots & 1 \end{bmatrix}, \tag{3}$$

and the corresponding coordinates at the current frame $f$ are given by

$$F_{jf} = \begin{bmatrix} q_{1d_{j1}} & q_{1d_{j2}} & \cdots & q_{1d_{jN_j}} \\ q_{2d_{j1}} & q_{2d_{j2}} & \cdots & q_{2d_{jN_j}} \\ q_{3d_{j1}} & q_{3d_{j2}} & \cdots & q_{3d_{jN_j}} \\ 1 & 1 & \cdots & 1 \end{bmatrix}. \tag{4}$$

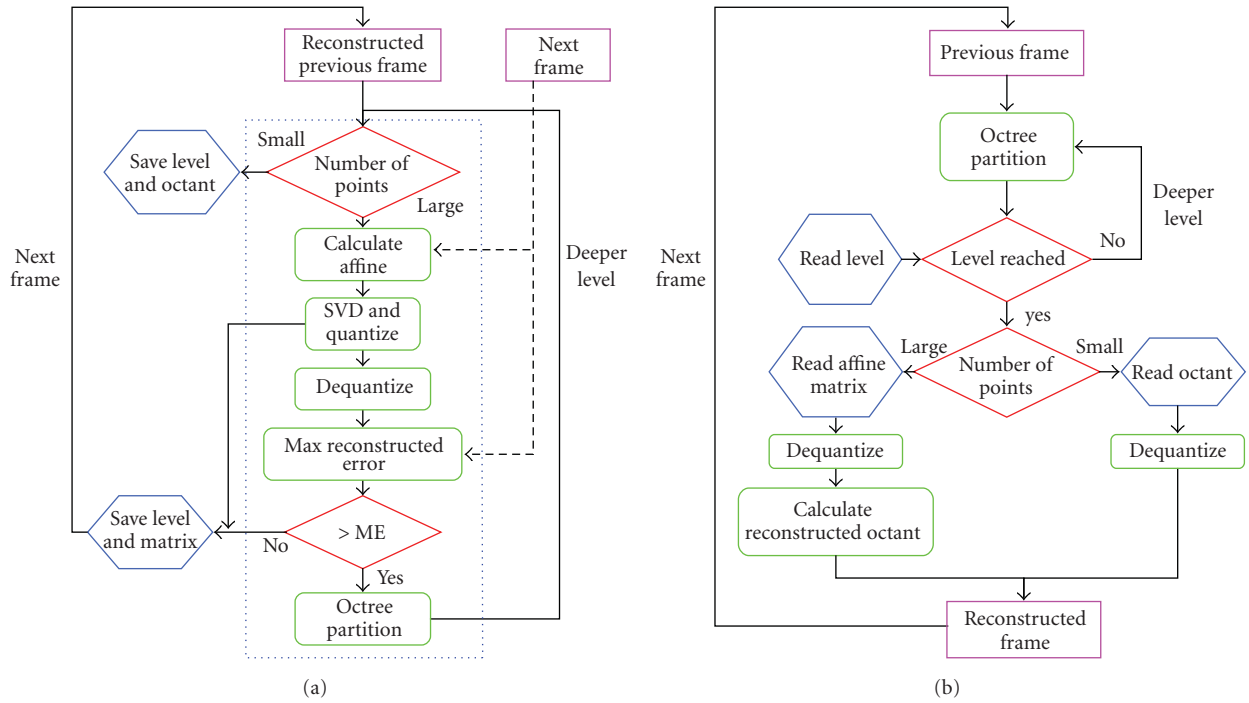



Figure 1: Flowchart of the algorithm: (a) Encoder, and (b) Decoder.

```
Algorithm(node)
   Affine(node)                    = 4 × n²
   Quantization(Affine)            = c_onst
   Reconstruction Error(node)      = 16 × n
   MaxError(vertices):             = 3 × n
   If Max Error less then Treshold = c_onst
      Stops
   Else
      Partition(node):             = 8 ∗ Algorithm(node/8)
   End
End
```

Algorithm 1: Complexity of the proposed algorithm.

The affine $A_j$ that minimizes the error $E_j$, that is, minimizes $A * F_{j(f-1)} = F_{jf}$ in the least square sense is given by a right pseudoinverse. That is

$$A_j F_{j(f-1)} = F_{jf}, \tag{5}$$

$$\begin{aligned} A_j F_{j(f-1)} F_{j(f-1)}^T &= F_{jf} F_{j(f-1)}^T, \\ A_j &= F_{jf} F_{j(f-1)}^T \cdot \left(F_{j(f-1)} F_{j(f-1)}^T\right)^{-1}. \end{aligned} \tag{6}$$

The matrix $A_j$ is a 4 by 4 matrix with [0 0 0 1] as the last row. Since each pair of corresponding points provides three constraints, to solve for the unknowns in the system of equations above $N$ must be $\geqslant 4$. Also, since the transformation between $F_{j(f-1)}$ and $F_{jf}$ is not a perfect transformation, the calculated $A_j$ leads to a reconstruction error $|A_j F_{j1} - F_{j2}| > 0$. If $N$ is smaller than 4, no affine is calculated and the position of the vertices in that cube is transmitted instead.

*3.4. Quantization and Singular Nodes.* Each element of the affine transformation matrix is stored using integers, which affects the precision, but increases the compactness of the representation. To compensate for this loss of precision, a frame $f$ is encoded with respect to the reconstructed frame, rather than the actual frame $f - 1$. By doing so, the quantization error in the latter frame is corrected by the motion estimation for the current one. Therefore, quantization errors only affect the frame, but do not propagate throughout the whole sequence.

The quantized affine transformation matrix $A'$ derived from the original affine transformation matrix $A$ by

$$A' = \left\lfloor 2^k \left( \frac{A - a_{\min}}{a_{\max} - a_{\min}} \right) \right\rfloor, \tag{7}$$

where $k$ is the quantization step. Also, in order to be able to compare our method with the method developed in [11], we set the same linear quantization method with a step of 16 bits. Ideally, $a_{\min}$ and $a_{\max}$ would be the minimum and maximum elements among all affine matrices. However, that would require the prior calculation of the motion vectors for the entire sequence. Instead, we use a predefined value for both $a_{\min}$ and $a_{\max}$. This arbitrary choice is possible because, as we explained earlier, we normalize the dimensions of the root cube to $i = 1 \cdots N_j$. That guarantees that the elements

Table 1: Properties of the benchmark sequences used for testing.

| | Vertices | Triangles | Frames | Size (bytes) |
|---|---|---|---|---|
| Dance | 7061 | 14118 | 190 | 16099080 |
| Chicken | 3030 | 5664 | 400 | 14544000 |
| Cow | 2904 | 5804 | 193 | 6725664 |
| Snake | 9179 | 18354 | 134 | 14759832 |
| Ball | 5552 | 11100 | 100 | 6662400 |

Table 2: Results of the Comparison for the "Chicken" sequence.

| | Ours | | | |
|---|---|---|---|---|
| | Matrices | Size | Ratio | PSNR |
| Chicken (1%) | 52694 | 709139 | 20.51 : 1 | 39.43 |
| Chicken (5%) | 15120 | 242625 | 59.94 : 1 | 32 |
| Chicken (10%) | 6822 | 119231 | 121.98 : 1 | 29 |
| Chicken (15%) | 4184 | 79548 | 182.83 : 1 | 27.74 |
| Chicken (25%) | 2025 | 44023 | 330.37 : 1 | 26 |

| | Paper[11] | | | Paper [17] | |
|---|---|---|---|---|---|
| | Size | Ratio | PSNR | Size | Ratio |
| Chicken (5%) | 6481000 | 22.5:1 | 28 | 490227 | 29 : 1 |
| Chicken (10%) | 318000 | 45.7:1 | 25.7 | 344985 | 58 : 1 |
| Chicken (15%) | 175000 | 83:1 | 24.6 | 205464 | 88 : 1 |
| Chicken (25%) | 76700 | 189:1 | 22.5 | N/A | N/A |

Table 3: Results of the Comparison for the "Dance" sequence.

| | Ours | | | | Paper [17] | |
|---|---|---|---|---|---|---|
| | Matrices | Size | Ratio | PSNR | Size | Ratio |
| Dance (1%) | 32103 | 489644 | 32.88 : 1 | 24.08 | 878270 | 18 : 1 |
| Dance (2%) | 20314 | 365378 | 37.18 : 1 | 21.53 | 490227 | 33 : 1 |
| Dance (3%) | 15217 | 283377 | 56.81 : 1 | 19.95 | 344985 | 47 : 1 |
| Dance (5%) | 10360 | 167687 | 96 : 1 | 18.06 | 205464 | 78 : 1 |

Table 4: Results of the Comparison for the "Cow" Sequence.

| | Ours | | | | Paper [17] | |
|---|---|---|---|---|---|---|
| | Matrices | Size | Ratio | PSNR | Size | Ratio |
| Cow (1%) | 43335 | 589656 | 11.4 : 1 | 26.35 | N/A | N/A |
| Cow (5%) | 15873 | 233673 | 28.78 : 1 | 19.68 | 424603 | 16 : 1 |
| Cow (10%) | 10310 | 163772 | 41.07 : 1 | 16.65 | 202942 | 32 : 1 |
| Cow (15%) | 6796 | 115296 | 58.33 : 1 | 15.27 | 140999 | 48 : 1 |

Table 5: Results for the "Snake" and "Ball" sequences.

| | Ours | | | |
|---|---|---|---|---|
| | Matrices | Size | Ratio | PSNR |
| Snake (1%) | 40590 | 616136 | 23.96 : 1 | 29.67 |
| Snake (2%) | 27066 | 410012 | 36 : 1 | 26.98 |
| Snake (3%) | 21588 | 327996 | 45 : 1 | 25.38 |
| Snake (5%) | 15167 | 231451 | 63.77 : 1 | 23.34 |
| Ball (2–5%) | 99 | 143 | 46590 : 1 | 43.83 |

Table 6: Results for the PCA algorithm using different bases.

| | 3 bases | | 5 bases | |
|---|---|---|---|---|
| | Ratio | PSNR | Ratio | PSNR |
| Dance (sd = 14) | 67.5 | 16.9 | 40.5 | 20.5 |
| Cow (sd = 6) | 163 | 12.6 | 98.1 | 14.1 |
| Chicken (sd = 16) | 139.3 | 28.2 | 83.6 | 31.1 |
| | 10 bases | | 30 bases | |
| | Ratio | PSNR | Ratio | PSNR |
| Dance (sd = 14) | 20.2 | 25.4 | 6.7 | 38.7 |
| Cow (sd = 6) | 49.2 | 16.64 | 16.5 | 23.7 |
| Chicken (sd = 16) | 42.2 | 36.4 | 14.2 | 47.4 |

of $A$ will only be large in the case of a singularity, for example, points are too close to each other. In that case, two things happen: (1) we apply a singular value decomposition (SVD) to solve for $A$ in (5); (2) we fix the reconstruction error to 5%. That is, when approximating the pseudoinverse by its SVD, we use only the eigenvalues corresponding to the first 95% of the principal components.

*3.5. Termination Criteria.* In Section 3.2, we explained how the algorithm stops at step (4). However, there are actually two criteria for such termination.

The first criterion to stop the partitioning of the octree comes from the reconstruction error. That is, the maximum reconstruction error allowed for any single vertex is defined by

$$\mathrm{ME} < \max_{i=1\cdots N_j} \left( \left| \overrightarrow{\widehat{q}_{d_{ji}}} - \overrightarrow{q_{d_{ji}}} \right| \right), \tag{8}$$

where $\overrightarrow{\widehat{q}_{d_{ji}}}$ and $\overrightarrow{q_{d_{ji}}}$ are the original and reconstructed vertices of the $j$th node.

In other words, if the reconstruction error of any single vertices exceeds ME, the node is partitioned into eight subcubes. Otherwise, the algorithm stops. In Section 4 we explain the choices of this threshold.

The second criterion to stop the partitioning is the number of vertices inside a cell. As we explained in Section 3.3, if that number is 4 or less, we store the coordinates of the vertices directly instead of the motion vectors (affine).

*3.6. Algorithm Complexity.* The complexity of our algorithm derives mainly from the pseudo inverse used during the calculation of the affine transformation. That is, for a $n \times 4$ matrix $F_{j(f-1)}$, the complexity of calculation the pseudo inverse is $O(4 \times n^2)$. Here, we will use $O(f(\cdot))$ as the *asymptotic upper bound* of $f(\cdot)$; $\Theta(f(\cdot))$ the *asymptotic tight bound* of $f(\cdot)$; $\Omega(f(\cdot))$ as the *asymptotic lower bound* of $f(\cdot)$.

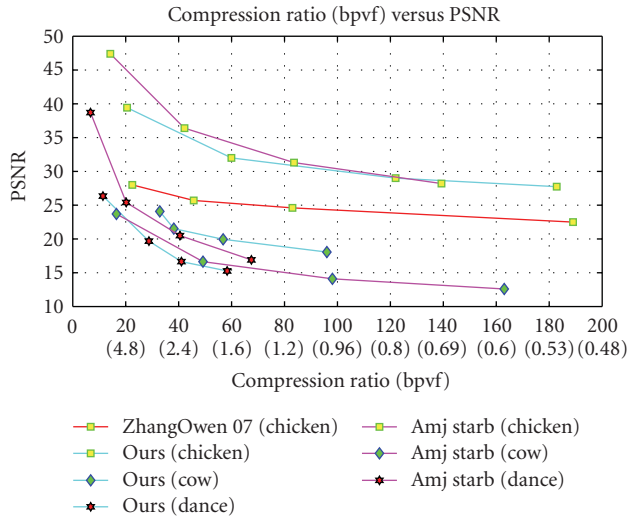



Figure 2: Comparison between our proposed method and ZhangOwen's and AmjStarb's algorithms.

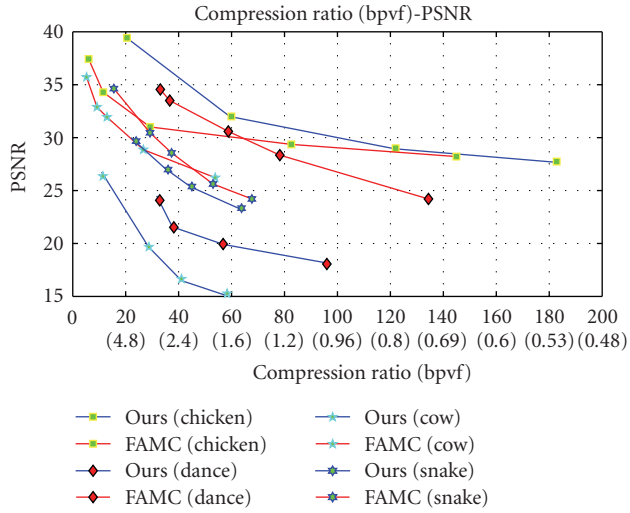



Figure 3: Comparison between the FAMC + DCT [14] and our proposed method.

Hence, when we consider each step of the proposed algorithm in Algorithm 1, we arrive the following recursive equation:

$$T(n) = c_1, \quad \text{for } n = 4,$$
$$T(n) = 4n^2 + c_2 + 19n + n + 8 * T\left(\frac{n}{8}\right), \quad \text{for } n > 4, \tag{9}$$

which can be further simplified as in:

$$T(n) = \Theta(1), \quad \text{for } n = 4$$
$$T(n) = 8 * T\left(\frac{n}{8}\right) + f(n), \quad \text{for } n > 4, \tag{10}$$

where $f(n) = 4n^2 + 20n + c_2$. This equation can be solved using the third case of the Master Theorem [31], where $f(n) = \Omega(n^{1+\epsilon})$, $\epsilon = 1$. That is, $f(n)$ is polynomially larger than $n^2$. Therefore, the solution for the recursion is $T(n) = \Theta(n^2)$.

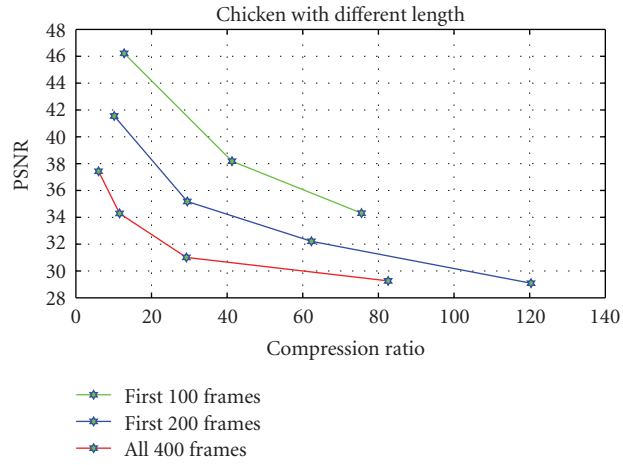



Figure 4: Decrease in compression ratio as a function of the number of frames used in the FAMC + DCT algorithm—for any given (fixed) PSNR.

*3.7. Internal Representation of the Data.* The final data consists of the following items. The first frame of the sequence with all $N$ uncompressed 3D coordinates of the vertices using single float-point precision: $N \times 3 \times 4$ bytes. The $m_f$ quantized affine transformation matrices corresponding to the $m_f$ leaf nodes found in the octree for each frame $f$ in the sequence of $F$ frames: $\sum_{f=1}^{F} m_f \times 3 \times 4 \times 2$. If a leaf node contains less than four vertices, the coordinates of these vertices are used instead. Finally, one byte for level of each leaf node in the octree. To further increase the compression ratio, we apply an arithmetic encoding to this data.

## 4. Results

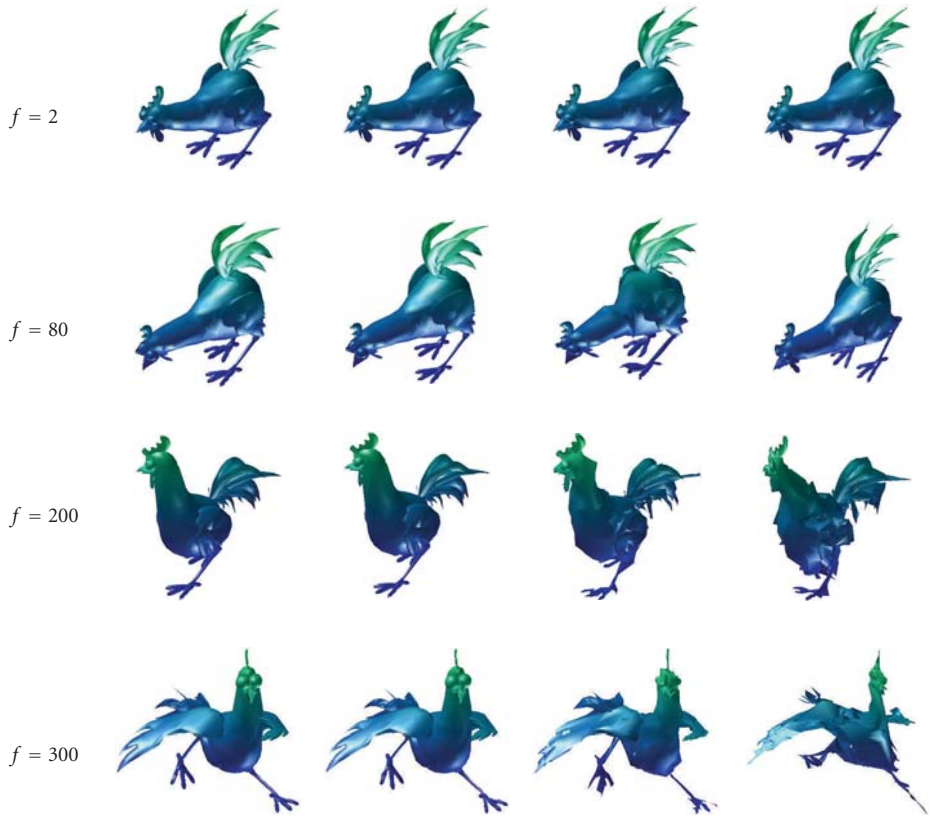
We compared our algorithm against three other methods in the literature. We will refer to these as the ZhangOwen's algorithm [11, 17]; the AmjStarb's algorithm [20]; the FAMC + DCT algorithm [14, 15]. As we already explained in Section 2, the first method uses an octree with a trilinear interpolation, while the second one uses a PCA-based approach, and the third uses an irregular partitioning also followed by an affine transformation.

*4.1. Quantitative Measurement.* In order to compare our method against these other approaches, we calculated the *Peak Signal-Noise Ratio*, or PSNR, as defined in [11]

$$\text{PSNR} = 10 \log_{10} \frac{d_{\max}}{\text{AVGMSE}},$$
$$\text{AVGMSE} = \frac{1}{M} \sum_{i=1}^{M} \text{MSE}_i, \tag{11}$$

(a) Original frame (b) Our method: PSNR = 32 dB (c) FAMC: PSNR = 34 dB (d) FAMC: PSNR = 30 dB

Figure 5: Illustration of the visual perception for differences of 2 dB in frame reconstruction.




(a) Original frame (b) ME = 0.01 (c) ME = 0.1 (d) ME = 0.25

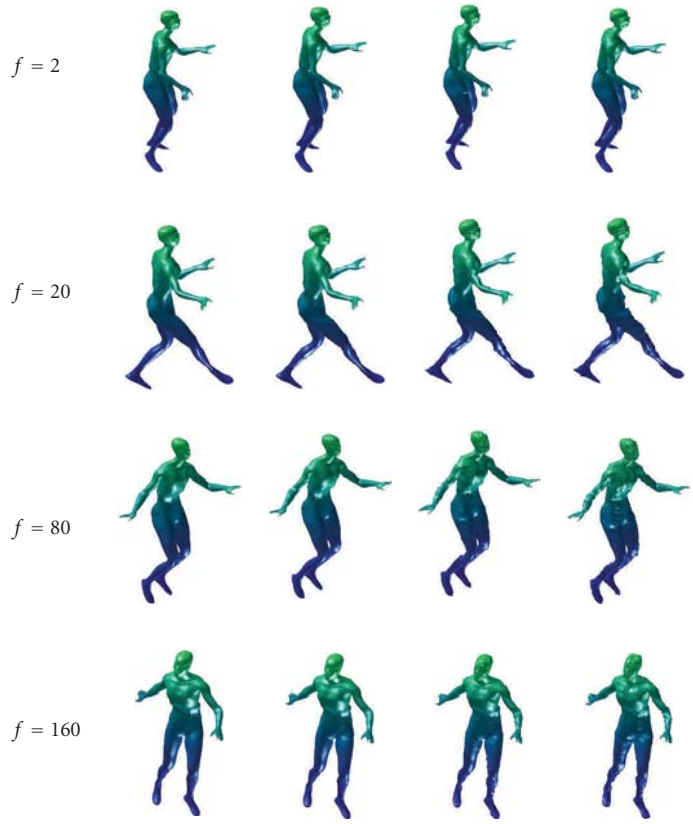
Figure 6: Reconstructed frames for the "Chicken" sequence.

where $d_{\max}$ is the size of the largest bounding box among the different frames. Also, MSE is the mean-square error of the distance between reconstructed and actual vertices, that is, $\text{MSE}_i = |\vec{\hat{q}}_i - \vec{q}_i|^2$.

*4.2. Benchmark Dataset.* We applied the four algorithms—ZhangOwen's algorithm, AmjStarb's, FAMC + DCT, and our method—to a total of five different benchmark sequences. Each of these sequences contains objects with different rigidity and number of vertices. Table 1 summarizes the properties of these sequences.

*4.3. Quantitative Results.* First, we compared our approach against ZhangOwen's and AmjStarb's algorithms in terms of Compression Ratio versus PSNR. Figure 2 shows the relationship between these two performance measures for three of the benchmark sequences. For detailed numerical data, the reader should refer to Tables 2, 3, 4, and 5. The points of the curves in Figure 2 for our method correspond to the different percentages in parenthesis in the tables, and they were obtained by controlling the choices of ME in (8). As for ZhangOwen's and AmjStarb's algorithms, the controlling parameters were, respectively, a similar maximum error threshold and the number of principal components (i.e., the number of bases)—as we explain later in this section.

As Figure 2 shows, our approach consistently outperforms ZhangOwen's algorithm, both in terms of compression ratio and PSNR. For AmjStarb's algorithm, our approach




(a) Original frame (b) ME = 0.01 (c) ME = 0.03 (d) ME = 0.05

Figure 7: Reconstructed frames for the "Dance" sequence.

performs quite similarly, however, we must point out once again that a PCA-based approach requires prior knowledge of the entire sequence in order to achieve good results, while our method can be carried out on a frame-to-frame basis.

As the tables above imply, the compression ratio is highly dependent on the animation sequence. For example, for the "Ball" and "Chicken" sequences, which present a simpler motion, the compression ratio is also higher. In especial, for the "Ball" sequence, the entire set of 100 frames required only 99 matrices. That means that one single affine matrix was sufficient for each of these frames—no subnode was required under the root node of the octree.

Also, the value used for ME does not completely determine the PSNR, although they are highly correlated. For example, the Chicken sequence with an ME = 5%, our method achieved a PSNR of 32, while for the Cow sequence an ME = 1% turned out to reach a PSNR of only 26.35. We assume that this is because of the complexity of the motion, which still is the dominating factor in the final result.

Since the author did not provide the calculation for PSNR in [17], the data for some of the sequences was not available. However, the author did not make any changes between [11] and [17] that could have affected the PSNR. Therefore, we can assume that the PSNR would be the same for both methods in [11, 17].

Finally, for the PCA-based algorithm in [20], it is important to mention that we implemented their approach using the same parameters reported in their paper. Table 6 summarizes the results for four different numbers of principal components or bases, that is, 3, 5, 10, and 30.

After the tests above, we also compared our method to a state-of-the-art algorithm: the FAMC + DCT [14, 15]. As we mentioned earlier, the FAMC is the standard method for the MPEG-4 AFX Amendment 2 [29] and it uses an irregular partitioning based on the topology of the mesh. That is, clusters of vertices are created by successively incorporating neighboring vertices that maintain the error in reconstruction from the affine transformation within a certain limit. This analysis of the error is performed over long stretches of the sequence, while a fixed partitioning, decided at the time of the first frame, is employed throughout the sequence. In Figure 3, we present a comparison between our method and the FAMC + DCT for the first four benchmark sequences in Table 1. As the figure shows, our method outperforms the FAMC + DCT for the Chicken sequence; it presents a performance similar to the FAMC + DCT for the Snake sequence; it is outperformed by the FAMC + DCT for the other two sequences (Dance and Cow). The reason for that behavior comes mainly from the rigid versus nonrigid characteristics of the partitioning in both methods.

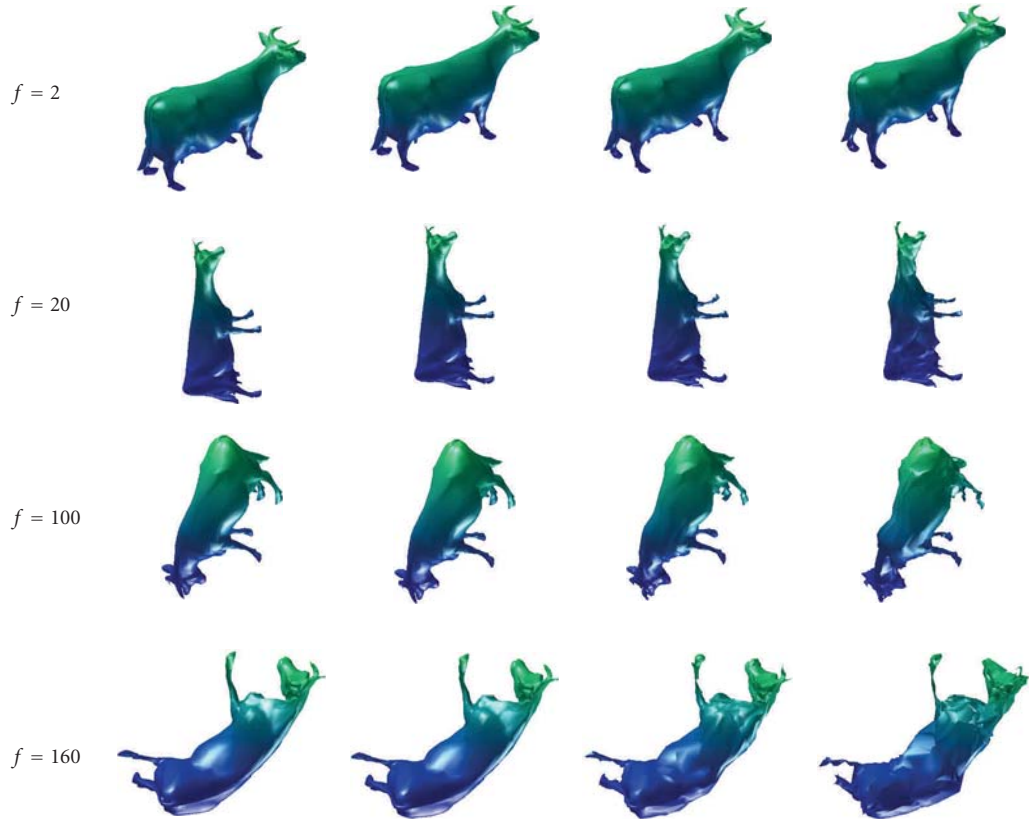



Figure 8: Reconstructed frames for the "Cow" sequence.

That is, while the octree can lead to unnecessary partitioning of neighboring vertices with the same motion vectors simply because of the proximity of another group of vertices with a different motion, in a non-rigid partitioning as in the FAMC, vertices are grouped independently of the existence of this second group of vertices. That difference between the two approaches usually leads to a more efficient partitioning of the vertices by the FAMC. However, that same efficiency is compromised by long sequences—as it was the case for the Chicken, which has the largest number of frames in our tests (400)—and by radical motions of the vertices—as it was the case for the Snake, whose vertices move much more freely than in the other sequences. In other words, since the size and the number of clusters in the FAMC depend on those two factors (the number of frames and the nonrigidity of the motion of the vertices), the larger they become, the more likely it is for the FAMC to require more clusters with fewer vertices in each cluster, and consequently to present a reduction in compression ratio. This claim is supported by Figure 4, where we depict the performance of the FAMC + DCT method for the Chicken sequence as a function of the number of frames used in the sequence. As the figure indicates, if we fix the PSNR, the compression ratio decreases as we add more frames to the sequence. It is important to notice however that the compression ratio used in those curves were obtained without taking into account the need to add a key frame at the end of each smaller sequence. This added reference frame is usually very large and even if a static compression is performed as suggested in [14], the cost of constantly including such key frames would lower even further the compression ratio. In order to perform the comparison in Figure 3, we chose to use all 400 frames in the Chicken sequence for the FAMC + DCT algorithm.

*4.4. Qualitative Results.* We also perform a qualitative analysis of our algorithm for different parameters. However, before we present these results, we would like to illustrate what means visually for two algorithms to present a difference of 2 dB in PSNR. In that regard, Figure 5(a) presents one of the original frames in the Chicken sequence and the corresponding reconstructed frames achieved by our method and by the FAMC with different PSNRs. Our algorithm, in Figure 5(b), obtained a PSNR of 32 dB, while the FAMC in Figures 5(c) and 5(d) presented PSNRs of 30 dB and 34 dB, respectively. As the reader will notice, a change of 2 dB between (b) and (c) does not seem as pronounced as the same 2 dB difference between (b) and (d). This result is again to illustrate that the same 2 dB can appear quite differently to humans, and that some times the absolute number alone

is not enough to determine which reconstructed frame looks "better."

Finally, Figures 6, 7, and 8 depict various reconstructed frames for each of the three sequences: Chicken, Dance and Cow; using different values of ME.

## 5. Conclusion and the Future Work

We proposed an affine-based motion representation with adaptive threshold and quantization that uses a rigid octree structure for partitioning. Our experimental results indicated that the proposed algorithm is, in terms of compression ratio, superior to other octree-based methods, while it achieved similar performance when compared to PCA-based methods or the FAMC + DCT algorithm. However, our method has a much smaller computation complexity and it does not require prior knowledge of the sequence. Instead, it can be carried out on a frame-to-frame basis.

We have demonstrated that the computation complexity of our algorithm for one frame is $\Theta(n^2)$, where $n$ is the number of vertices in the frame. Once again, this is a major advantage of our method when compared to other algorithms such as ZhangOwen's algorithm [11, 17] and the AmjStarb's algorithm [20]. Moreover, while state-of-the-art methods such as the FAMC + DCT can be improved by means of a PCA technique (as described in [18]) and outperform the compression ratio of our method in all tested sequences, the price tag to pay in that case is a complexity greater than $\Theta(n^3)$.

Both the PSNR and the compression ratios for our method were very high, and a choice of ME = 0.01 provided an excellent compromise between these two performance measurements.

One serious limitation of most time-dependent methods, including our method, is the requirement for correspondence between vertices in different frames. This prevents this method from being applied to real 3D data—cloud of points. In order to solve this problem, we propose to build a pseudocorrespondence between frames using the Iterative Closet Points algorithm [32, 33]. Another option would be the use of Robust Points Matching (RPM) [34–36] however, we anticipate that this technique would lead to a much more complex algorithm. A combination of ICP and RPM was proposed [37], which have a much better performance, but still the run time is a concern, since the RPM is $O(n^3)$ and ICP is $O(n \cdot \log)$. In the future, we intend to attack this problem of pseudocorrespondences.

## Acknowledgments

The authors would like to thank Andrew Glassner for giving them access to Chicken data; Aljoscha Smolic, Nikolce Stefanoski, and Kivanc Kose for sharing the Chicken data; Hector Briceno for sharing the Cow, Ball, Snake, and Dance data. They would also like to thank Jinghua Zhang for explanations provided regarding their papers.

## References

[1] M. Dearing, "Geometry compression," in *Proceedings of the 22nd Annual Conference on Computer Graphics (SIGGRAPH '95)*, pp. 13–20, Los Angeles, Calif, USA, August 1995.

[2] Y. Huang, J. Peng, and M. Gopi, "Octree-based progressive geometry coding of point clouds," in *Proceedings of Eurographics Symposium on Point-Based Graphics*, pp. 103–110, 2006.

[3] R. Schnabel and R. Klein, "Octree-based point cloud compression," in *Proceedings of Eurographics Symposium on Pointbased Graphics*, pp. 111–120, 2006.

[4] C. Touma and C. Gotsman, "Triangle mesh compression," in *Proceedings of Graphics Interface Conference*, pp. 26–34, Canadian Human-Computer Communications Society, Vancouver, Canada, June 1998.

[5] J. Rossignac, "3D compression made simple: edgebreaker with zip and warp on a corner-table," in *Proceedings of IEEE Conference on Shape Modelling and Applications*, pp. 278–283, 2001.

[6] A. Szymczak, "Optimized edgebreaker encoding for large and regular triangles meshes," in *Proceedings of Data Compression Conference (DCC '02)*, p. 472, Snao Bird, Utah, USA, 2002.

[7] T. Lewiner, H. Lopes, J. Rossignac, and A. W. Vieira, "Efficient Edgebreaker for surfaces of arbitrary topology," in *Proceedings of the 17th Brazilian Symposium of Computer Graphic and Image Processing and 2nd Ibero-American Symposium on Computer Graphics*, pp. 218–225, Curitiba, Brazil, October 2004.

[8] M. Weeks and M. Bayoumi, "3D discrete wavelet transform architectures," in *Proceedings of IEEE International Symposium on Circuits and Systems (ISCAS '98)*, vol. 4, pp. 57–60, Monterey, Calif, USA, May-June 1998.

[9] F. F. Rodler, "Wavelet-based 3D compreesion with fast random access for very large volume data," in *Proceedings of the 7th Pacific Conference on Computer Graphics and Applications*, p. 108, 1999.

[10] J. E. Lengyel, "Compression of time-dependent geometry," in *Proceedings of the Symposium on Interactive 3D Graphics*, pp. 89–95, Atlanta, Ga, USA, April 1999.

[11] J. Zhang and C. B. Owen, "Octree-based animated geometry compression," in *Proceedings of Data Compression Conference (DCC '04)*, pp. 508–517, Snowbird, Utah, USA, March 2004.

[12] S. Gupta, K. Sengupta, and A. A. Kassim, "Compression of dynamic 3D geometry data using iterative closest point algorithm," *Computer Vision and Image Understanding*, vol. 87, no. 1–3, pp. 116–130, 2002.

[13] P.-F. Lee, C.-K. Kao, J.-L. Tseng, B.-S. Jong, and T.-W. Lin, "3D animation compression using affine transformation matrix and principal component analysis," *IEICE Transactions on Information and Systems*, vol. E90-D, no. 7, pp. 1073–1084, 2007.

[14] K. Mamou, T. Zaharia, and F. Prêteux, "Famc: the MPEG-4 standard for animated mesh compression," in *Proceedings of International Conference on Image Processing (ICIP '08)*, pp. 2676–2679, San Diego, Calif, USA, October 2008.

[15] N. Stefanoski and J. Ostermann, "Spatially and temporally scalable compression of animated 3D meshes with MPEG-4/FAMC," in *Proceedings of the 15th International Conference on Image Processing (ICIP'08)*, pp. 2696–2699, San Diego, Calif, USA, October 2008.

[16] J. Zhang and J. Xu, "Optimizing octree motion representation for 3D animation," in *Proceedings of the 44th Annual Southeast Conference*, pp. 50–55, Melbourne, Fla, USA, March 2006.

[17] J. Zhang, J. Xu, and H. Yu, "Octree-based 3D animation compression with motion vector sharing," in *Proceedings of the 4th International Conference on Information Technology-New Generations (ITNG '07)*, pp. 202–207, Las Vegas, Nev, USA, April 2007.

[18] K. Mamou, T. Zaharia, F. Preteux, A. Kamoun, F. Payan, and M. Antonini, "Two optimizations of the MPEG-4 FAMC standard for enhanced compression of animated 3D meshes," in *Proceedings of the 15th IEEE International Conference on Image Processing (ICIP'08)*, pp. 1764–1767, San Diego, Calif, USA, October 2008.

[19] M. Alexa and W. Müller, "Representing animations by principal components," *Computer Graphics Forum*, vol. 19, no. 3, pp. 411–418, 2000.

[20] R. Amjoun and W. Straßer, "Efficient compression of 3D dynamic mesh sequences," *Journal of the WSCG*, vol. 15, no. 1–3, pp. 99–107, 2007.

[21] I. Guskov and A. Khodakovsky, "Wavelet compression of parametrically coherent mesh sequences," in *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pp. 136–146, Grenoble, France, 2004.

[22] F. Payan and M. Antonini, "Wavelet-based compression of 3D mesh sequences," in *Proceedings of the 2nd IEEE International Conference on Machine Intelligence (ACIDCA-ICMI '05)*, Tozeur, Tunisia, November 2005.

[23] K. Muller, A. Smolic, M. Kautzner, and T. Wiegand, "Rate-distrotion optimization in dynamic mesh compression," in *Proceedings of IEEE International Conference on Image Processing*, pp. 533–536, Atlanta, Ga, USA, October 2006.

[24] M. Sattler, R. Sarletter, and R. Klein, "Simple and efficient compression of animation sequences," in *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pp. 209–217, Los Angeles, Calif, USA, 2005.

[25] Q. Gu, J. Peng, and Z. Deng, "Compression of human motion capture data using motion pattern indexing," *Computer Graphics Forum*, vol. 28, no. 1, pp. 1–12, 2009.

[26] J.-H. Yang, C.-S. Kim, and S.-U. Lee, "Compression of 3-D triangle mesh sequences based on vertex-wise motion vector prediction," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 12, no. 12, pp. 1178–1184, 2002.

[27] L. Ibarria and J. Rossignac, "Dynapack: space-time compression of the 3D animations of triangle meshes with fiexed connectivity," in *Proceedings of the ACM SIGGRAPH/Eurographics Sysmposium on Computer Animation (SCA '03)*, pp. 126–135, 2003.

[28] M. Tournier, X. Wu, N. Courty, E. Arnaud, and L. Revéret, "Motion compression using principal geodesics analysis," in *Proceedings of ACM Siggraph/Eurographics Symposium on Computer Animation (SCA '08)*, July 2009.

[29] "Iso/iec 14496-16, mpeg-4 animation framework extension (afx) model," http://wwwwww.iso.org//iso/iso_catalogue/_tc/catalogue_detail.htm?csnumber=44097.

[30] C. L. Jackins and S. L. Tanimoto, "Oct-trees and their use in representing three-dimensional objects," *Computer Graphics and Image Processing*, vol. 14, no. 3, pp. 249–270, 1980.

[31] T. H. Cormen, R. L. Rivest, C. E. Leisersonc, and C. Stein, *Introduction to Algorithms*, McGraw-Hill, New York, NY, USA, 2003.

[32] Y. Chen and G. Medioni, "Object modeling by registration of multiple range images," in *Proceedings of IEEE International Conference on Robotics and Automation*, vol. 3, pp. 2724–2729, Sacramento, Calif, USA, April 1991.

[33] S. Rusinkiewicz and M. Levoy, "Efficient variants of the ICP algorithm," in *Proceedings of the 3rd International Conference on 3-D Digital Imaging and Modeling*, pp. 145–152, Quebec City, Canada, 2001.

[34] H. Chui, J. Rambo, J. Duncan, R. Schultz, and A. Rangarajan, " Registration of cortical anatomical structures via robust 3D point matching," in *Proceedings of the 16th International Conference Information Processing in Medical Imaging (IPMI '99)*, pp. 168–181, Visegrád, Hungary, June-July 2003.

[35] A. Rangarajan, H. Chui, and E. Mjolsness, "A relationship between spline-based deformable models and weighted graphs in non-rigid matching," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 1, pp. I897–I904, Kauai, Hawaii, USA, December 2001.

[36] H. Chui and A. Rangarajan, "A new point matching algorithm for non-rigid registration," *Computer Vision and Image Understanding*, vol. 89, no. 2-3, pp. 114–141, 2003.

[37] N. Okada and M. Hebert, "Fast 3D tracking of non-rigid objects," in *Proceedings of IEEE International Conference on Robotics and Automation*, vol. 3, pp. 3497–3503, Taipei, Taiwan, September 2003.

Hindawi Publishing Corporation
International Journal of Digital Multimedia Broadcasting
Volume 2010, Article ID 425641, 16 pages
doi:10.1155/2010/425641

*Research Article*

# A Scalable Multiple Description Scheme for 3D Video Coding Based on the Interlayer Prediction Structure

**Lorenzo Favalli and Marco Folli**

*Department of Electronics, University of Pavia, Via Ferrata 1, 27100 Pavia, Italy*

Correspondence should be addressed to Lorenzo Favalli, lorenzo.favalli@unipv.it

Received 2 May 2009; Revised 8 September 2009; Accepted 11 December 2009

Academic Editor: Pietro Zanuttigh



The most recent literature indicates multiple description coding (MDC) as a promising coding approach to handle the problem of video transmission over unreliable networks with different quality and bandwidth constraints. Furthermore, following recent commercial availability of autostereoscopic 3D displays that allow 3D visual data to be viewed without the use of special headgear or glasses, it is anticipated that the applications of 3D video will increase rapidly in the near future. Moving from the concept of spatial MDC, in this paper we introduce some efficient algorithms to obtain 3D substreams that also exploit some form of scalability. These algorithms are then applied to both coded stereo sequences and to depth image-based rendering (DIBR). In these algorithms, we first generate four 3D subsequences by subsampling, and then two of these subsequences are jointly used to form each of the two descriptions. For each description, one of the original subsequences is predicted from the other one via some scalable algorithms, focusing on the inter layer prediction scheme. The proposed algorithms can be implemented as pre- and postprocessing of the standard H.264/SVC coder that remains fully compatible with any standard coder. The experimental results presented show that these algorithms provide excellent results.

## 1. Introduction

The research on stereoscopic video has received high interest over the past decade in order to provide viewers with more realistic vision than traditional 2D video. Stereoscopic video coding has been traditionally performed using two different views which are then merged to create a full set of different possible views. Other than using this left and right views, a more recent technique, known as depth image-based rendering (DIBR) [1], represents 3D video based on a monoscopic video and associated per-pixel depth information, simply called *color image* (or *video*) and *depth map,* respectively. While the color consists of three components, $Y$, $U$, and $V$ as in the traditional video applications, the depth map video uses only one component to store the depth information of objects within the scene related to the camera position. The advantage of such a scheme is that it can capture the stereoscopic sequences more easily compared to the traditional left and right views techniques.

Even though 3D video can provide a more impressive experience than conventional 2D video, in the past, many factors such as huge bandwidth requirements and the discomfort due to special devices needed to observe 3D effect prevented widespread diffusion of 3D video in commercial services. Following the path of "traditional" video, it is easily foreseeable that, thanks to the improvements in coding techniques [2], also 3D applications will soon be available over both the Internet and wireless networks.

Reliable video transmission over unreliable or inefficient networks such as the Internet or wireless networks poses many challenges related to bandwidth variations and packet losses due to congestion on one side and to fading, interference, and mobility on the other one [3]. Traditionally, to cope with network and device heterogeneity, scalability techniques have been proposed.

A scalable video sequence is composed of a so-called *base*-layer and of one (or more) *enhancement*-layer(s): compared to a single-layer sequence, the base-layer is self-contained and fully decodable to a signal of lower quality and/or

lower resolution in terms of pixel or time. Enhancement layers, on the contrary, cannot be decoded if the base layer is lost or damaged and can only be used to improve the overall quality. Scalable coders not only allow a stream to be sent over channels with different bandwidth constraints or to devices having different capabilities, but also allow for different error protection schemes or even adaptive transmission techniques to be applied. Scalability is successfully introduced in the coding algorithms since the MPEG2 standard [4] up to the fine grain scalable option (FGS) in MPEG4 [5, 6] and H.264 [7].

A different approach in search for a solution to the problem of heterogeneous and unreliable networks is represented by multiple description coding (MDC). An MDC algorithm creates several substreams, all individually decodable, each at a lower quality than the original: receiving all the descriptions, ideally allows the full recovery of the single stream coded video [8, 9]. This approach is very attractive since it is possible to exploit the inherent protection provided by path diversity among the different descriptions [10].

Since scalability and multiple descriptions target the solution of different problems (bandwidth variations for scalability and robustness for multiple description coding), it is useful to exploit a combination of the two complementary methods in order to obtain a more efficient video coding algorithm. Previous works have already addressed the topic, mixing scalability and MDC thus creating scalable multiple descriptions (MDSC) algorithms. Approaches include exploitation of temporal segregation [11], a hybrid of spatial and temporal prediction [12], and wavelet coding [13, 14].

The starting point in this paper is to develop efficient mixes of scalability and multiple description and be compatible with a standard H.264/SVC coder. To this aim, we developed a method using simple pre and post processing schemes to generate substreams that can be used within the H.264/SVC coder. In the preprocessing part we down sample the original color and depth maps by rows and columns generating four subsequences that can be independently coded. To reduce redundancy, we propose to predict two of them by using some of the tools that guarantee scalability in the H.264/SVC coder. This method, called Interlayer Prediction Spatial Multiple Description Scalable Coding (ILPS-MDSC), takes advantage of the interlayer prediction method [7] introduced in H.264/SVC.

The first sections of the paper are devoted to a description of previous related work in 3D and multiple descriptions coding (Sections 2 and 3, resp.) and to the introduction of the scalable coding tools implemented in H.264/SVC (Section 4). The algorithm is described in Section 5, while the description of its implementation on top of the H.264/SVC coder and simulation results are provided in Section 6.

## 2. 3D Coding Algorithms

In this section we briefly recall the two basic techniques used to generate multiple views that can be used to generate 3D videos.

*2.1. Stereoscopic (Left/Right) Video Coding.* Coding of stereoscopic images has been the subject of research through several years. It basically mimics the human binocular view and consists of coding scenes captured by two slightly distant cameras. In stereoscopic viewing, two sets of images are used, one offered to the left eye and the other to the right eye of the viewer. Each of the views forms a normal two-dimensional sequence. Furthermore, the two views are taken under specific constraints, such that the human brain can fuse these two slightly different views resulting in the sensation of depth in the scene being viewed. Since this means doubling the information to be transmitted, the main focus in the research has been to exploit correlation between the two views. Since both cameras capture the same scene, the basic idea is to exploit the interview redundancy for compression. These redundancies can be of two types: interview similarity is across adjacent camera views, and temporal similarity between temporally successive images of each video. A coder that can efficiently explicit this kind of similarity is usually called *multiview coder* [15]. Unfortunately, although the two images are very similar, a *disparity* problem emerges as the system geometry may not be perfect. This disparity must be compensated and is generally described as an horizontal displacement between points collected by the two different sensors.

*2.2. The DIBR Algorithm.* The main advantage of DIBR technique compared to traditional representation of 3D video with left-right views is that it provides high-quality 3D video with smaller bandwidth. This is because the depth map can be coded more efficiently than the two streams of monoscopic views if correlations and properties of the depth map are properly identified. Conceptually, DIBR utilizes a depth frame to generate two virtual views from the same reference (original) view, one for the left eye and the other one for the right eye [16]. This process can be described by the following 2-steps procedure. Firstly, the original image points are re-projected into the 3D domain, utilizing the respective depth values. Thereafter, given the position of the left and right camera in the 3D domain, the views are obtained by the projection of the 3D intermediate points to the image plane of each camera. This procedure is usually referred to as "3D image warping" in the computer graphics literature. In this process the original image points at locations $(x, y)$ are transferred to new locations $(x_L, y)$ and $(x_R, y)$ for left and right view, respectively according with the following formulas

$$x_L = x + \frac{\alpha_x \cdot t_c}{2}\left(\frac{1}{Z} - \frac{1}{Z_C}\right),$$
$$x_R = x - \frac{\alpha_x \cdot t_c}{2}\left(\frac{1}{Z} - \frac{1}{Z_C}\right), \quad (1)$$

where $\alpha_x$ is the focal length of the reference camera expressed in multiples of the pixel width and $t_C$ is the distance between the left and right virtual cameras. $Z_C$ is the convergence distance located at the zero parallax setting (ZPS) plane and $Z$ denotes the depth value of each pixel in the reference view.

Notes that the $y$ component is constant since the virtual cameras used to capture the virtual views (left-right) are assumed to be located at the same horizontal plane. The quality of virtual views depends on the quality of received color and depth map.

## 3. Multiple Descriptions Algorithms

Generally speaking, multiple description coding dates back to the mid 1970s when the first theoretical works appeared [17] soon followed by applications to voice coding [18]. Application to video was later introduced as described in [8]. The concept is simple and is based on splitting in some way the information to be transmitted which is then sent over independent channels. Being able to actually generate two (or more) distinct and independent information flows would optimally exploit transmission diversity. In the following we introduce some algorithms for MDC developed in the 2D video coding framework and some applications to 3D video.

*3.1. Multiple Descriptions for 2D Video Transmission.* A simple and efficient MDC scheme can be based on the temporal splitting of the odd and even frames of a video sequence into separate, individually decodable descriptions that can be decoded using standard receivers. Such a scheme, called Multiple Description Motion Compensation (MDMC), is described in [19] by designing temporal predictors that exploit not only the temporal correlation within a description, but also across the descriptions. Another simple method for MDC derives the descriptions as spatially subsampled copies of the original video sequence. In [20] a polyphase sub sampler along rows and columns is used to generate an arbitrary number of downsampled descriptions that are then independently coded. This scheme is then called Polyphase Spatial Subsampling multiple description coding (PSS-MDC).

The main problem of these techniques, as long as they are solely multiple description methods, is that they aim at increasing the robustness by exploiting link diversity and do not address other important transmission challenges, such as bandwidth variations or device heterogeneity, which require a scalable approach. On the other side, a traditional scalable approach does not guarantee the same robustness provided by MDC.

The complementarity of the two approaches has been exploited to implement an efficient solution that was called Multiple Description Scalable Coding (MDSC). A simple example of MDSC is the scalable extension of MDMC, proposed by [11]. A combination of motion compensation and spatial subsampling is described in [12].

A new type of MDSC in which the multiple description is not obtained only via spatial or temporal algorithms but also introducing quality metrics affecting signal-to-noise ratio (SNR) has been proposed by several authors using wavelet-based coding, in order to reduce temporal and spatial redundancy [13]. Another approach using the DWT is proposed in [14]. It is possible to combine MCTF and DWT in the so-called 3D (or 2D+$t$) discrete wavelet transform which may be used to generate a PSS-MD with an arbitrary number or descriptions by first subsampling the original sequence by rows and columns and then coding each of them [21].

Based on the scalable extension of the H.264 coder (H.264/SVC [22]), in [23] the authors proposed a further improvement of the MD-MCTF scheme, which separates each highpass frame generated by MCTF in two frames, the motion frame and the texture frame. Each of these frames is then handled separately, and the motion information is divided between the descriptions using a quincunx lattice, while the texture information is divided by sending the odd frames in one of the descriptions and the even frames in another with the exception of the intra coded macroblocks which are inserted in both descriptions.

*3.2. Multiple Descriptions for 3D Video Coding.* Depending on the stereoscopic coding approach used, different algorithms have been proposed to extend MDC to 3D video sequences.

There are primarily three approaches that can be used for coding of stereoscopic video. The first one, called *Simulcast stereoscopic coding,* involves independent coding of left and right views: either of the two views may be decoded and displayed as a compatible signal on 2D video receivers. In the second, called *Compatible stereoscopic coding*, one view, say the left view, is coded independently while the other one, in this case the right view, is then coded with respect to the independently coded view. Thus, only the independently coded view may be decoded and displayed as the compatible signal. Finally, the last method, called *Joint stereoscopic coding*, consists in coding both the left and right views together. Most of the common approaches tend to mix different views into one description. In these cases, one view is fully coded, while the others are usually time and/or spatially subsampled version of the originals [24]. This way of doing ensures that some information from which we reconstruct a three-dimensional vision is always available.

Considering the case of DIBR coding, descriptions may be formed using any of the MDC algorithms described for the case of 2D video while depth data are superimposed to any description and eventually represent the *enhancement layer* in an MDSC-like approach [25]. An example of this scheme is the one proposed in [25], where the temporal MDC of the base layer is produced by using the multistate video coding approach, which separates the even and odd frames of a sequence into two MDC streams. Hence, in the scalable MDC simulation, the even and odd frames are separated before the encoding process for both texture and depth and are used to generate the two descriptions: one contains the even frames of both color view and depth map, while the other description contains the odd frames. In both descriptions, frames from the color view are coded in the base layer and those from the depth map are coded in the enhancement layer.

## 4. Description of Scalable Coding Tools

H.264/AVC [22] is one of the latest international video coding standards. It uses state-of-the-art coding techniques and provides enhanced coding efficiency for a wide range of applications, including video telephony, video conferencing and video streaming. The scalable extension of H.264/AVC, named H.264/SVC, uses a layered approach to provide spatial, temporal, and SNR scalability such that only some specific combinations of spatial, temporal, and SNR data can be extracted and decoded from a global scalable video stream. The basic idea in designing the scalable extension of H.264/AVC is to extend the hybrid coding approach of this coder toward motion compensated temporal filtering. However, the algorithms that perform the prediction and update steps are similar to motion compensation techniques in the generalized B frame used in H.264/AVC, so that backward compatibility of the base layer of the scalable extension is guaranteed. Further detail of H.264/SVC can be found in [26].

In addition to the basic coding tools of H.264/AVC, SVC provides some so-called interlayer prediction methods, which allow the exploitation of the statistical dependencies between different layers to improve the coding efficiency of the enhancement layers. In the previous coder, the only supported interlayer prediction methods employ the reconstructed samples of the lower layer signal. The prediction signal is formed by motion-compensated prediction inside the enhancement layer, by upsampling the reconstructed lower layer signal, or by averaging such an upsampled signal with a temporal prediction signal. Although the reconstructed lower layer samples represent the complete lower layer information, they are not necessarily the most suitable data that can be used for interlayer prediction.

Usually, the interlayer predictor has to compete with the temporal predictor, and especially for sequences with slow motion and high spatial detail, the temporal prediction signal typically represents a better approximation of the original signal than the upsampled lower layer reconstruction. In order to improve the coding efficiency for spatial scalable coding, two additional interlayer prediction concepts have been added in SVC: prediction of macroblock modes and associated motion parameters and prediction of the residual signal. All interlayer prediction tools can be chosen on a macroblock or submacroblock basis allowing an encoder to select the coding mode that gives the highest coding efficiency and they are described below.

*InterLayer Intra Prediction*. for SVC enhancement layers, an additional macroblock coding mode is provided, in which the macroblock prediction signal is completely inferred from colocated blocks in the reference layer without transmitting any additional side information. When the colocated reference layer blocks are intra-coded, the prediction signal is built by the upsampled reconstructed intra signal of the reference layer: a prediction method also referred to as interlayer intra-prediction

*InterLayer Macroblock Mode and Motion Prediction*. when at least one of the colocated reference layer blocks is not intra-coded, the enhancement layer macroblock is inter-picture predicted as in single-layer H.264/AVC coding, but the macroblock partitioning—specifying the decomposition into smaller blocks with different motion parameters, and the associated motion parameters—is completely derived from the colocated blocks in the reference layer. This concept is also referred to as interlayer motion prediction. For the conventional intercoded macroblock types of H.264/AVC, the scaled motion vector of the reference layer blocks can also be used as replacement for usual spatial motion vector predictor.

*InterLayer Residual Prediction*. a further interlayer prediction tool referred to as interlayer residual prediction, targets a reduction of the bit rate required to transmit the residual signal of intercoded macroblocks. With the usage of residual prediction, the upsampled residual of the colocated reference layer blocks is subtracted from the enhancement layer residual (difference between the original and the inter-picture prediction signal) and only the resulting difference, which often has a smaller energy then the original residual signal, is encoded using transform coding as specified in H.264/AVC.

Other scalability techniques can be exploited by the H.264/SVC coder. In each layer, an independent hierarchical motion-compensated prediction structure with layer-specific motion parameters is employed. This hierarchical structure provides a temporal scalable representation of a sequence of input pictures that is also suitable for efficiently incorporating spatial and quality scalability. The reconstruction quality of a layer can be improved by additional coding of the so-called progressive refinement slices. These slices represent refinements of the texture data (intra- and residual data). Since the coding symbols of progressive refinement slices are ordered similarly to a coarse-to-fine description, these slices can be arbitrarily truncated in order to support fine granular scalability (FGS) or flexible bit-rate adaptation. Finally, Medium Grain Scalability (MGS) is introduced. This scalability allows to choose a partitioning of a block of $4 \times 4$ DCT coefficients as specified by the so called *MGSVectors*. A quality identifier ($Q_{\mathrm{id}}$) which represents a priority inside the bitstream is then assigned to each packet of a slice. This type of scalability may lead to a number of layers that varies from coarse grain scalability to fine grain scalability. All these types of scalability are not specifically addressed in the present work.

## 5. Proposed Scheme

We have developed a scheme to generate multiple descriptions, based on the prediction algorithms of the scalable extension of H.264/AVC. In order to preserve the standard coder, a pre and post processing scheme is implemented. In the preprocessing part, we downsample the original sequence by rows and columns thus generating four different subframes, similarly to what is done in PSS-MD. These subframes could in principle be used to generate four independent descriptions. Since they are in general highly correlated, we can exploit this by reducing the number of descriptions to two so that each description will be formed

by two of the subframes arranged so that one subframe will be predicted form the other. By doing this, we still obtain two independent descriptions and, for each description, we remove a large part of the redundancy due to spatial correlation.

Scalability within each description is easily obtained exploiting the options provided by the scalable extension of the H.264 standard introduced in Section 4. Specifically, a coarse grain scalable (CGS) stream (i.e., a stream formed by a base layer and a single enhancement layer) is obtained by forcing the coder to accept one subframe as the base layer and the other as the enhancement layer. This process is depicted in Figure 3 and repeated for both the descriptions to be generated. Prediction is performed by the coder itself using its internal *interlayer* prediction *algorithms*. No prediction is performed across the descriptions.

In case of bandwidth variations or devices requiring a lower-grade signal, scalability will be simply obtained by discarding the enhancement layer(s). Note that a more radical form of scalability can be devised by allowing the receiver to decode only the base layer of one description, which means in practice that only 1/4th of the information will be delivered. In other words the following scenarios can be seen.

(1) Both descriptions are completely received. In this case all the information is recovered,

(2) One of the enhancement layers is dropped (regardless which one). In this case the receiver only looses 1/4th of the information as a description will be fully received together with the base layer of the other one,

(3) Both enhancement layers are lost. Now we have only half of the information available corresponding to the base layers of both descriptions,

(4) One description is completely lost as well as the enhancement layer of the other. Still something can be displayed to the user although corresponding to only one of the original image pixels.

In the postprocessing part, the original sequence is obtained by merging the descriptions. In case of lost description or discarded enhancement layers, the missing pixels are reconstructed by interpolation from the received ones.

Two different schemes are applied to group the subsequences obtained after the polyphase spatial subsampling. In the first one, called *by rows*, we group the subsequences in which the pixels form even or odd rows of the original sequence. In the other one, called *quincunx*, we group the subsequence so that the pixels form a quincunx lattice of the original sequence.

In other words, referring to Figures 1 and 2, if we number the four subsequences as in the raster scan from the top left corner, in the *by rows* scheme, we form the first description with subsequences one and two, and the other one with subsequences three and four. Instead, in the *quincunx* scheme, we group the subsequences one and three to form the first description, and the subsequences two and four for the other one.

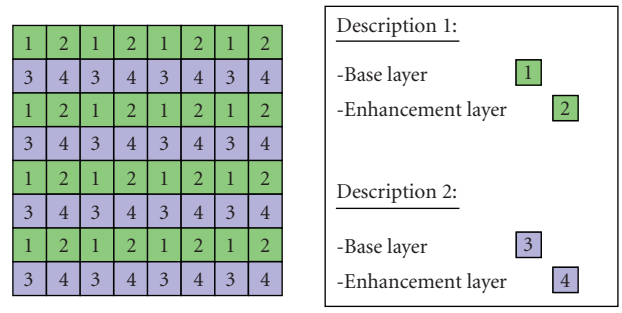



Figure 1: Subsampling patterns and descriptions composition for the case of by rows sampling.

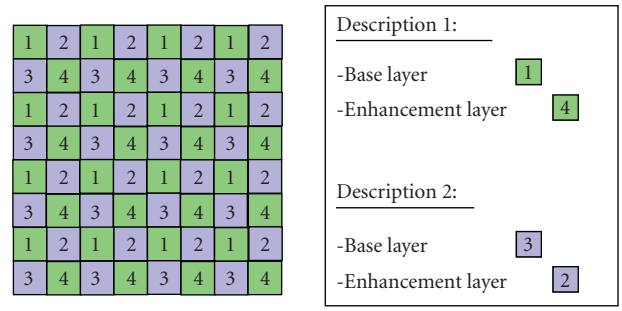



Figure 2: Subsampling patterns and descriptions composition for the case of quincunx sampling.

In case of lost information (either a whole description or the subframe corresponding to an enhancement layer), we use two different interpolation methods according to the different transmission scheme applied. In case we receive only one *by rows* description; then we recover the missing information taking the mean of the two nearest pixels. Otherwise, if we receive only one *quincunx* description, we recover the missing information simply taking the mean of the four nearest pixels.

Given this basic description of the algorithm, there are two different way to apply it to 3D video sequences. The first scheme, that takes into account the *Simulcast stereoscopic coding*, consists in coding independently the two different views (both Left/Right or Color/Depth), by applying the interlayer prediction algorithm to two subsequences of the same view, similar to what was done for the 2D sequences in previous works [27]. We call this method *Interlayer Prediction Structure (ILPS)*. In the second one, we try to create a *Compatible stereoscopic coding*, by subsampling the views in four subsequences, as done in PSS, and using the interlayer prediction to predict all the subsequences of one view (resp., right or depth), from the same subsequence of the other view (left or color). We call this scheme 3D-ILPS.

3D-ILPS is somehow similar to what is done in [24] where the authors mix a spatially scaled copy of one view with the full version of the other to form a description. By doing this, the authors do not introduce scalability in the strict sense in each description but exploit prediction to remove redundancy between the two views in each

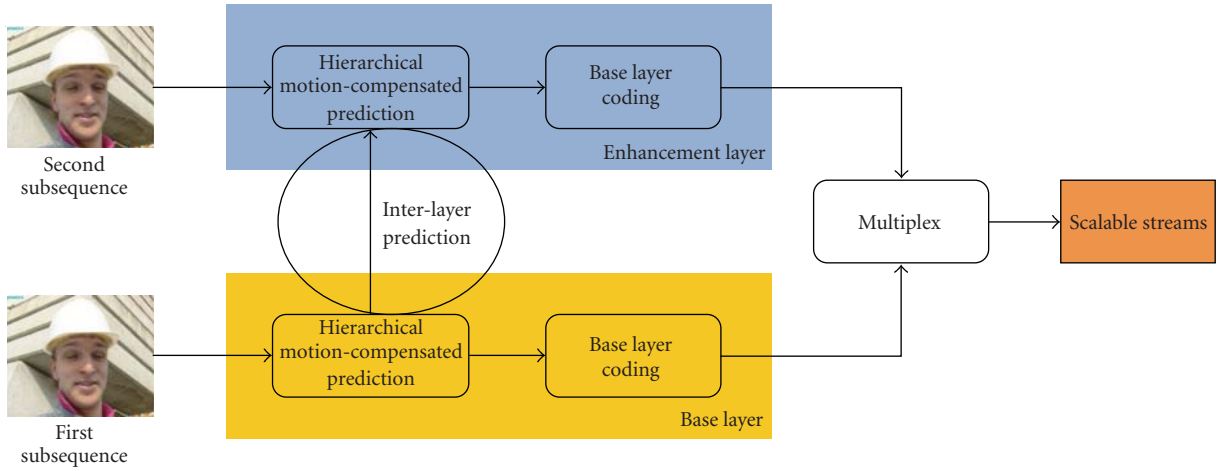



Figure 3: ILPS-MDSC with the interlayer prediction highlighted.

description. The redundancy introduced by this algorithm is then driven by the subsampling factor and the *QP* parameter applied to the predicted sequence.

The two proposed algorithms differ significantly and their characteristics can be exploited in different networking environments. In ILPS, each description carries a subsampled version of one of the two views, either left-right or color-depth, in a way that introduces a SNR scalability. In this method, the redundancy introduced is driven only by the header needed to correctly transmit all the descriptions. For those characteristics, we can say that the ILPS method can be very useful in case of transmission over unreliable channels in which all the receivers can decode a three-dimensional sequence, but have heterogeneous devices that require some sort of scalability. In the 3D-ILPS algorithm, instead, we propose a slightly different way to form the description, while keeping the same subsampling method as ILPS. By forming the description with, respectively, a subsampled version of the left/color and right/depth sequence, we introduce a new dimensional scalability. With this algorithm, it is possible to easily extract a 2D view from the stereoscopic sequence by discarding the enhancement layer. Considering this, we can apply the 3D-ILPS algorithm to network in which coexist users that can decode and view the 3D-videos and user that can only view monoscopic sequences.

## 6. Results

The software used in our experiments is H.264/SVC rel. 8.1, but the algorithm can be easily extended to virtually every version of the coder. Results are reported using the YUV 4 : 2 : 0 sequences *Breakdance* and *Ballet* (100 frames each), in order to determine the performance of the method under several different conditions. All sequences are at CIF resolution, 30 fps, single reference frame, GOP size 8. An 1 frame is only inserted at the beginning, prediction is performed at 1/4 pixel accuracy over $16 \times 16$, $16 \times 8$, $8 \times 16$, 8x8 blocks with SAD metric. Finally, CABAC is applied. Both sequences are natively at a resolution of $1024 \times 768$ and have been converted to CIF format by means of a non-normative downsampling, described in JVT-R006.

The results are shown over a rate span from 100 kbit/s to 1900 kbit/s in steps of 200 kbit/s, with or without random losses, to evaluate the pure performances of the coder with respect to other coding methods. In particular we plot the quality-rate curves for the color and left views, and we show some tables of the performances of the depth view, from a rate span of 100 kbit/s to 900 kbit/s, since after that rate the differences of PSNR are very minimal. Regarding the right view, no results are reported, since it can be easily extended from the ones showed for the left view.

Comparisons are mainly carried out using peak signal-to-noise ratio (PSNR) evaluation. Although it is known that it is unable to capture some 3D specific artifacts, PSNR is anyway widely used in literature to asses the quality of a 3D sequence from the quality of its components [25, 28, 29] and it has been shown that, though far from being perfect, it shows a good relationship with visual evaluation [30]. We also present some limited subjective quality measurement with the aid of the Mean Opinion Score (MOS), by playing the two rendered views (for the left and right eye) independently to the viewers.

In order to evaluate the robustness of the proposed algorithm with respect to the simple PSS-MD algorithm and a Single Description Coding (SDC), we have considered a random packet losses of about 10% of the overall transmitted packet. In case we receive only one description, we considered a packet loss of 5%, since in this cases we lost about the 55% of the information. In order to compare the performance of the single description coding with the case when only one description is received, we have simulated in such sequences a packet loss of about 50%, equal to the loss of information obtained when only one description is received.

A set of simulations has also been run considering a multiview video coder (MVC), specifically the H.264/MVC video coder version 5.0, where we predict the right sequence

Table 1: Performance of depth view, Ballet, without packet loss.

| | One subsequence received | | | | Two subsequence received | | | |
|---|---|---|---|---|---|---|---|---|
| Rate | SDC | PSS | ILPS | 3D-ILPS | SDC | PSS | ILPS | 3D-ILPS |
| 100 | 39,46 | 33,22 | 33,22 | 33,50 | 39,46 | 34,28 | 35,77 | 37,17 |
| 300 | 44,58 | 34,47 | 34,47 | 34,45 | 44,58 | 36,49 | 38,88 | 39,57 |
| 500 | 47,19 | 34,74 | 34,74 | 34,73 | 47,19 | 37,02 | 39,85 | 40,32 |
| 700 | 48,95 | 34,86 | 34,86 | 34,85 | 48,95 | 37,28 | 40,32 | 40,67 |
| 900 | 50,38 | 34,92 | 34,92 | 34,92 | 50,38 | 37,41 | 40,61 | 40,89 |
| | Three subsequence received | | | | Four subsequence received | | | |
| Rate | SDC | PSS | ILPS | 3D-ILPS | SDC | PSS | ILPS | 3D-ILPS |
| 100 | 39,46 | 35,16 | 35,58 | 35,81 | 39,46 | 35,78 | 36,61 | 36,70 |
| 300 | 44,58 | 38,75 | 38,96 | 38,85 | 44,58 | 41,55 | 42,15 | 41,86 |
| 500 | 47,19 | 39,81 | 39,95 | 39,87 | 47,19 | 44,20 | 44,84 | 44,48 |
| 700 | 48,95 | 40,37 | 40,47 | 40,37 | 48,95 | 46,15 | 46,82 | 46,31 |
| 900 | 50,38 | 40,69 | 40,77 | 40,70 | 50,38 | 47,69 | 48,35 | 47,90 |

Table 2: Performance of depth view, Ballet, 10% packet loss.

| | One subsequence received | | | | Two subsequence received | | | |
|---|---|---|---|---|---|---|---|---|
| Rate | SDC | PSS | ILPS | 3D-ILPS | SDC | PSS | ILPS | 3D-ILPS |
| 100 | 34,64 | 32,62 | 32,38 | 33,24 | 34,64 | 33,48 | 34,90 | 36,50 |
| 300 | 36,97 | 33,92 | 33,76 | 33,92 | 36,97 | 35,59 | 37,54 | 38,41 |
| 500 | 39,13 | 34,30 | 33,99 | 34,11 | 39,13 | 35,90 | 37,93 | 39,02 |
| 700 | 39,53 | 34,19 | 34,36 | 34,43 | 39,53 | 36,22 | 38,52 | 39,67 |
| 900 | 40,75 | 34,21 | 34,18 | 34,54 | 40,75 | 36,09 | 38,85 | 39,55 |
| | Three subsequence received | | | | Four subsequence received | | | |
| Rate | SDC | PSS | ILPS | 3D-ILPS | SDC | PSS | ILPS | 3D-ILPS |
| 100 | 34,64 | 37,07 | 34,42 | 35,40 | 37,69 | 34,07 | 35,49 | 35,89 |
| 300 | 36,97 | 35,65 | 37,18 | 38,14 | 40,61 | 37,62 | 39,36 | 41,22 |
| 500 | 39,13 | 38,16 | 38,29 | 39,24 | 43,78 | 38,42 | 42,54 | 42,24 |
| 700 | 39,53 | 38,88 | 39,14 | 39,18 | 44,70 | 38,52 | 43,83 | 44,63 |
| 900 | 40,75 | 38,84 | 38,96 | 39,64 | 44,39 | 39,15 | 45,18 | 46,14 |

from the left one. Unfortunately, this kind of coder does not support efficiently packet losses, so it was not possible to evaluate the performances of this algorithm in an error prone environment.

One final notation remark is that in all the tables, FD means that both base layer and enhancement layer are received for the specific description, BL refers to the case when only the base layer is received, and SVC and MVC are used to indicate single view and multiview video coding, respectively.

*6.1. Algorithm Performance.* We first present results for the cases when a subsequence is either completely received or it is completely lost. In case of loss of a subsequence, in order for the matching description to be decodable, we assume it corresponds to the subsequence used as the enhancement layer of the scalable description. Figures 4 and 5 show the performance when only one subsequence, corresponding to the base layer of one description, is received. As expected, all the proposed algorithms achieve the same performances since it is not possible, in this situation, to get benefits from our algorithms. In Figures 6 and 7 we consider the case when two subsequences, corresponding to the two base layers, are received. In this case, the performances of the *quincunx* and the *rows* schemes are very dependent from the specific sequence, so that it is not possible to select the best overall scheme. Instead, we can see that the ILPS algorithm seems to give better performances than the 3D-ILPS only in the *rows* scheme. Figures 8 and 9 show the performances when three subsequences are received. In these cases, all the considered algorithms seem to give similar performances, with only the 3D-ILPS algorithm that gives slightly worse performances. Finally, Figures 10 and 11 show the performances when all the subsequences are received. It is easy to see that, for the *Breakdance* sequence, the performances of the proposed method are comparable with the SDC and MDC, while, for the *Ballet* sequence, they are slightly worse.

Table 3: Performance of depth view, Breakdance, without packet loss.

| | One subsequence received | | | | Two subsequence received | | | |
|---|---|---|---|---|---|---|---|---|
| Rate | SDC | PSS | ILPS | 3D-ILPS | SDC | PSS | ILPS | 3D-ILPS |
| 100 | 38,58 | 33,41 | 33,41 | 33,66 | 38,58 | 33,72 | 36,24 | 38,29 |
| 300 | 44,41 | 35,00 | 35,00 | 34,99 | 44,41 | 35,52 | 40,19 | 42,28 |
| 500 | 47,26 | 35,40 | 35,40 | 35,40 | 47,26 | 36,03 | 41,68 | 43,97 |
| 700 | 49,20 | 35,56 | 35,56 | 35,55 | 49,20 | 36,26 | 42,46 | 44,94 |
| 900 | 50,70 | 35,63 | 35,63 | 35,63 | 50,70 | 36,36 | 42,91 | 45,51 |
| | Three subsequence received | | | | Four subsequence received | | | |
| Rate | SDC | PSS | ILPS | 3D-ILPS | SDC | PSS | ILPS | 3D-ILPS |
| 100 | 38,58 | 35,92 | 36,44 | 36,42 | 38,58 | 35,86 | 36,65 | 36,42 |
| 300 | 44,41 | 40,73 | 41,17 | 41,58 | 44,41 | 41,26 | 42,06 | 41,58 |
| 500 | 47,26 | 42,88 | 43,26 | 44,35 | 47,26 | 44,15 | 44,98 | 44,35 |
| 700 | 49,20 | 44,19 | 44,48 | 46,40 | 49,20 | 46,26 | 47,05 | 46,40 |
| 900 | 50,70 | 45,01 | 45,25 | 47,97 | 50,70 | 47,85 | 48,66 | 47,97 |

Regarding the depth view, Tables 1 and 3 report the PSNR value for the proposed algorithm, without packet losses. It is easy to see that the proposed algorithm gives better results than the PSS that and are not so far from the SDC. It is worth noting that all simulations are run with the coder generating the same total amount of bits for all the techniques. This means that the the MDC algorithms suffer from the fact that some redundancy is intrinsically introduced for two main reasons. First of all, since no prediction is performed across the descriptions, some correlation is anyway present and this reduces coding efficiency. Second, for each description to be independently (de)coded, we need headers to be duplicated. As a consequence, in the simulation conditions proposed, SDC will always perform better than MDC when all the information is received. The fact that ILPS and 3D-ILPS provide improvements with respect to PSS can be seen as a better coding efficiency.

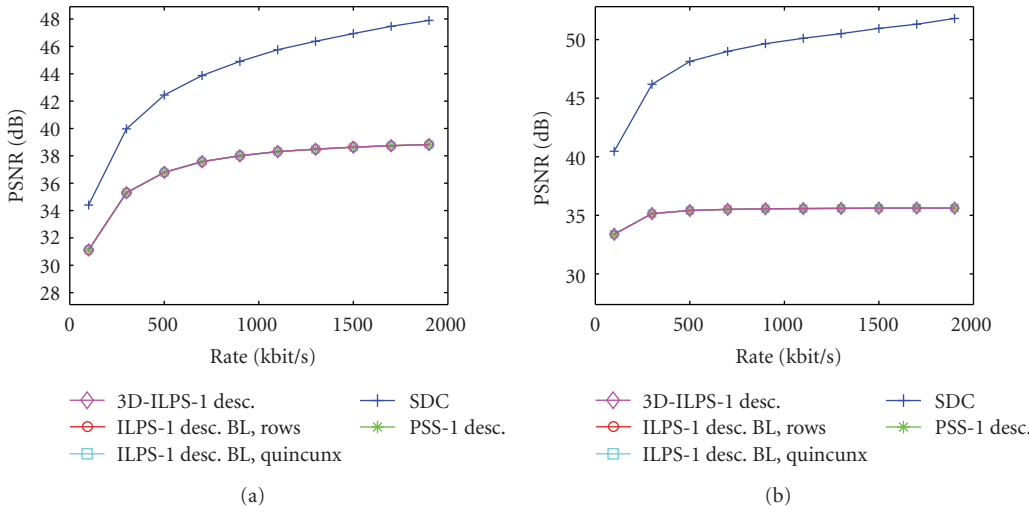



Figure 4: Performance of color view, one subsequence received. (a) Breakdance. (b) Ballet.

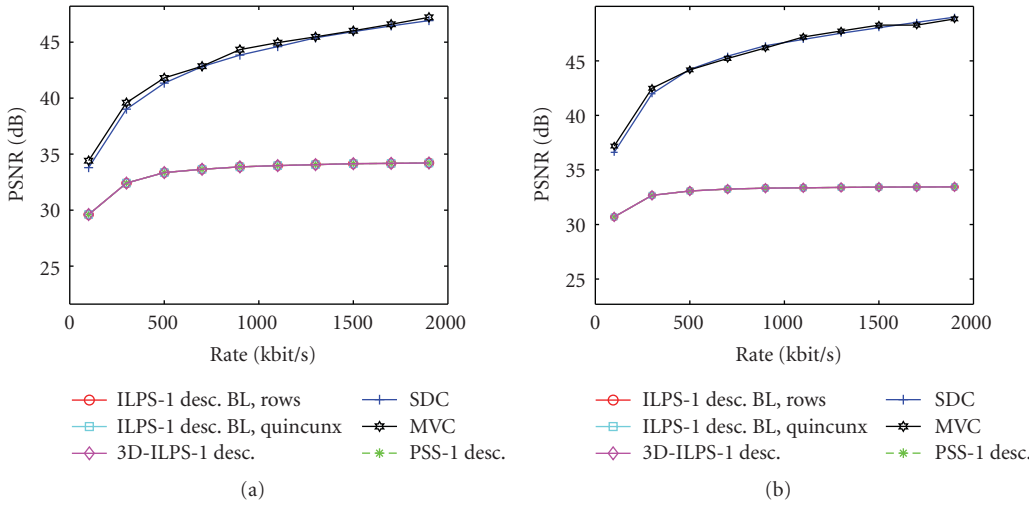



Figure 5: Performance of left view, one subsequence received. (a) Breakdance. (b) Ballet.

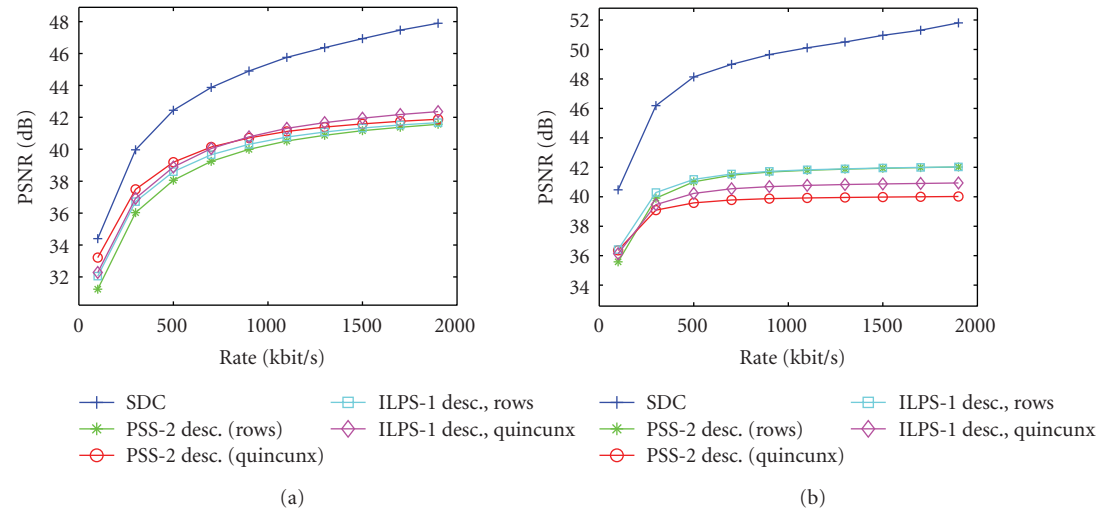



Figure 6: Performance of color view, two subsequences received. (a) Breakdance. (b) Ballet.

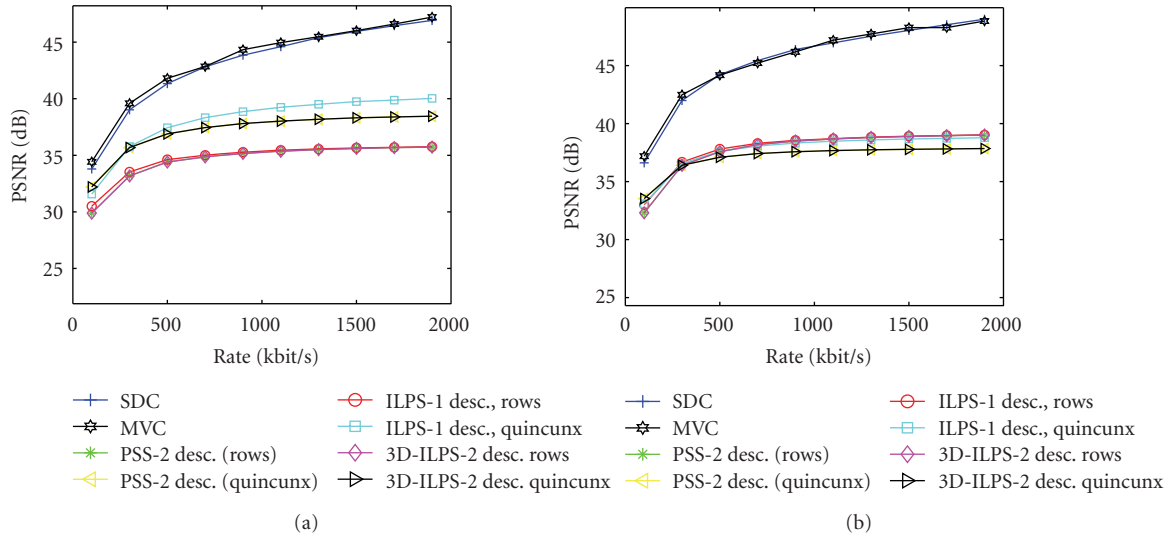



Figure 7: Performance of left view, two subsequences received. (a) Breakdance. (b) Ballet.

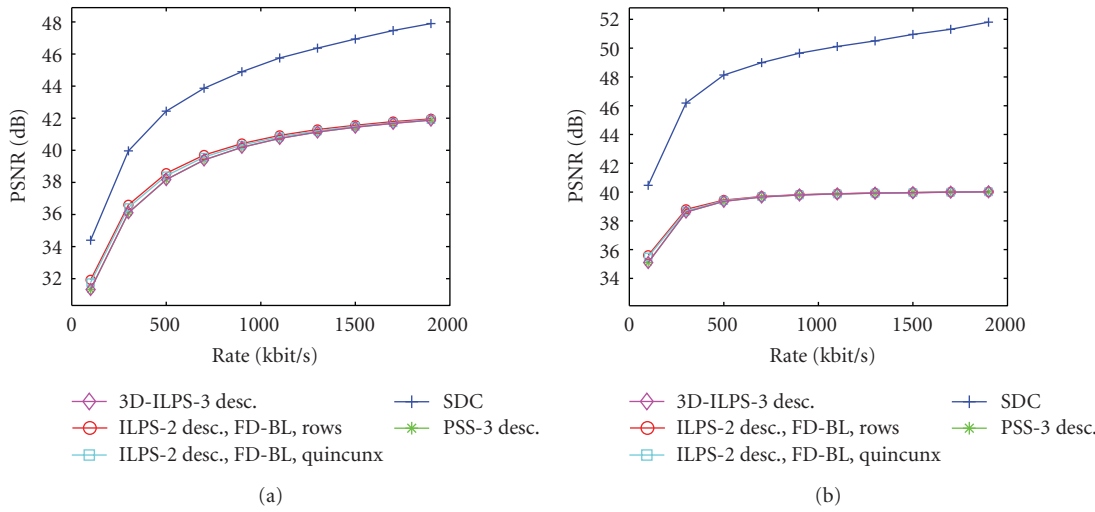



Figure 8: Performance of color view, three subsequences received. (a) Breakdance. (b) Ballet.

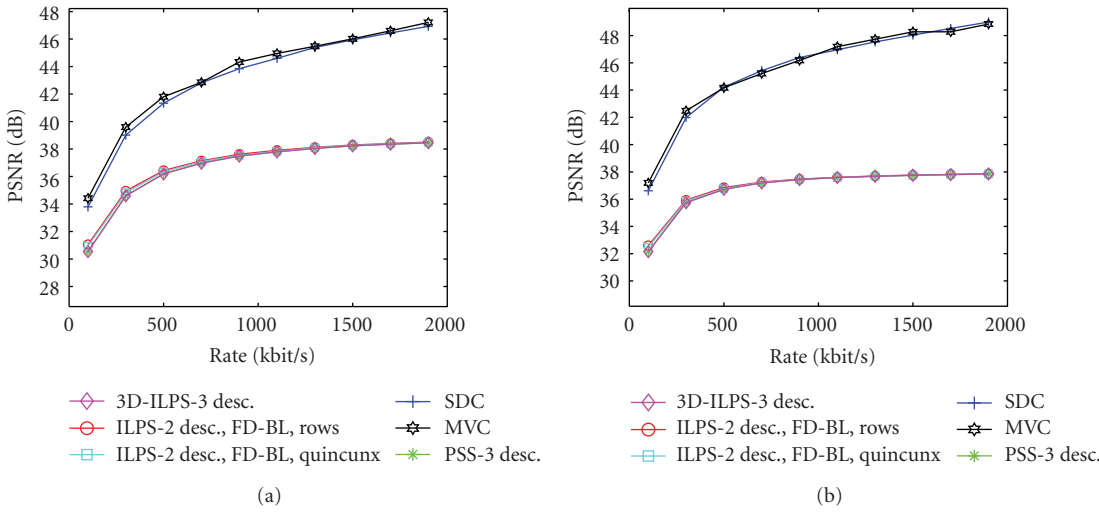



Figure 9: Performance of left view, three subsequences received. (a) Breakdance. (b) Ballet.

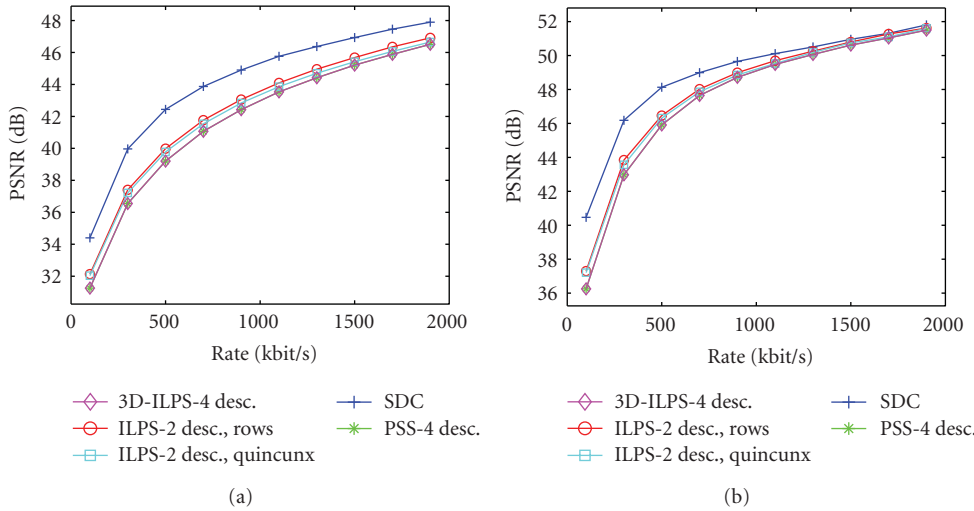



Figure 10: Performance of color view, all subsequences received. (a) Breakdance. (b) Ballet.

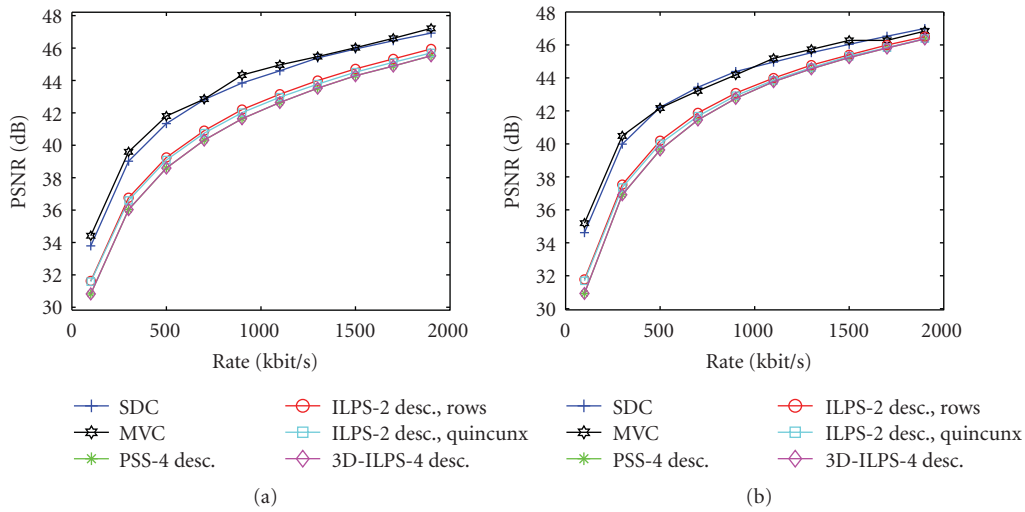



Figure 11: Performance of left view, four subsequences received. (a) Breakdance. (b) Ballet.

*6.2. Performance with Random Losses.* We now consider the case when random losses are superimposed to the previous cases. Random packet losses are generated as outcomes of a uniform distribution with loss probability $P = 10\%$. More detailed loss mechanisms should be considered in specific environments also taking into account the application scenario (real time versus streaming) and specific error control/retransmission strategies at medium access control layer. Furthermore, no specific error concealment strategies have been implemented so that it can be reasonably assumed that more realistic loss distributions would affect all different proposed methods in similar ways. Random losses on the contrary are less aggressive with single description streams.

Several simulations have been run for each of the possible cases using frame replacement as only error concealment technique. Despite of this, in some cases the decoder could not proceed: the reported results are given averaging over 10 different complete realizations. As expected, to obtain this number, SDC usually required a number of runs between two and three times larger than that of the MDC algorithms. This ratio may also be seen as a robustness gain provided by the path diversity approach. Note that no bandwidth constraints were given so that packet losses were only due to the random process.

Again, results are presented for increasing number of subsequences received and for color and left coding.

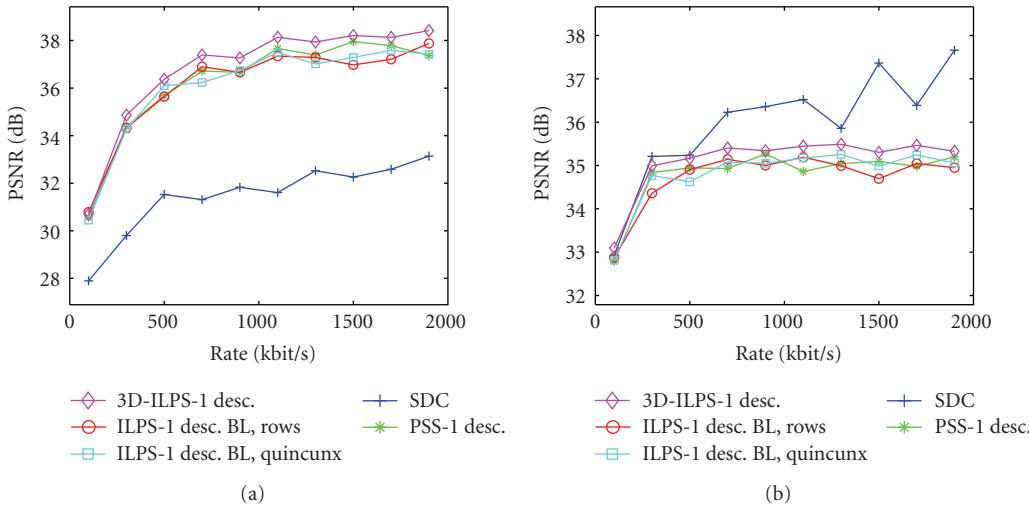



Figure 12: Performance of color view, one subsequence received, 10% packet loss. (a) Breakdance. (b) Ballet.

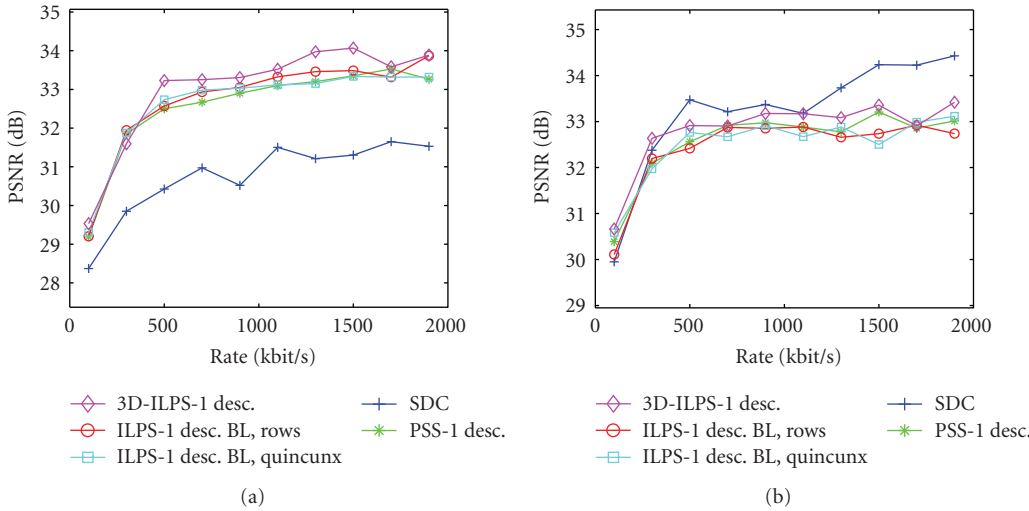



Figure 13: Performance of left view, one subsequence received, 10% packet loss. (a) Breakdance. (b) Ballet.

Figures 12 and 13 show the performance when only one subsequence is received, with random packet losses. In this case, the 3D-ILPS algorithm seems to give the better robustness to the packet losses. In Figures 14 and 15 it is possible to see the performance when two subsequences are received with random packet losses. As said before, the performances of the *quincunx* and the *rows* schemes are very dependent from the specific sequence, but we can easily view that the 3D-ILPS scheme again gives the best robustness to packet losses. Figures 16 and 17 show the performances when three subsequences are received. In these cases, all the considered algorithms seems to give similar performances, with only the 3D-ILPS algorithm that gives slightly lesser performances. Figures 18 and 19(b) sequences are received. In these cases, we can note a similar performance as the previous cases. Considering the depth view, Tables 2 and 4 report the robustness of the algorithms to a random packet loss of about 10%. These results indicate that both algorithms perform better than the PSS and the SDC coding. In particular, the 3D-ILPS algorithm seems to give the best performances among the considered schemes.

These results confirm the superior performances of the ILPS algorithm with respect to PSS, both in terms of coding efficiency and error robustness. This is not surprise since the 3D coding depicted scenario is an extension of the 2D coding scheme evaluated in the previous work [27].

The new algorithm proposed, the 3D-ILPS, further improves the performance of the ILPS scheme since it has been specifically studied for the 3D environment. Predicting the depth view from the same color view with the aid of the DIBR algorithm, this new algorithm has a better coding efficiency. Furthermore it also exhibits a strong robustness to transmission errors, since we can obtain four independent

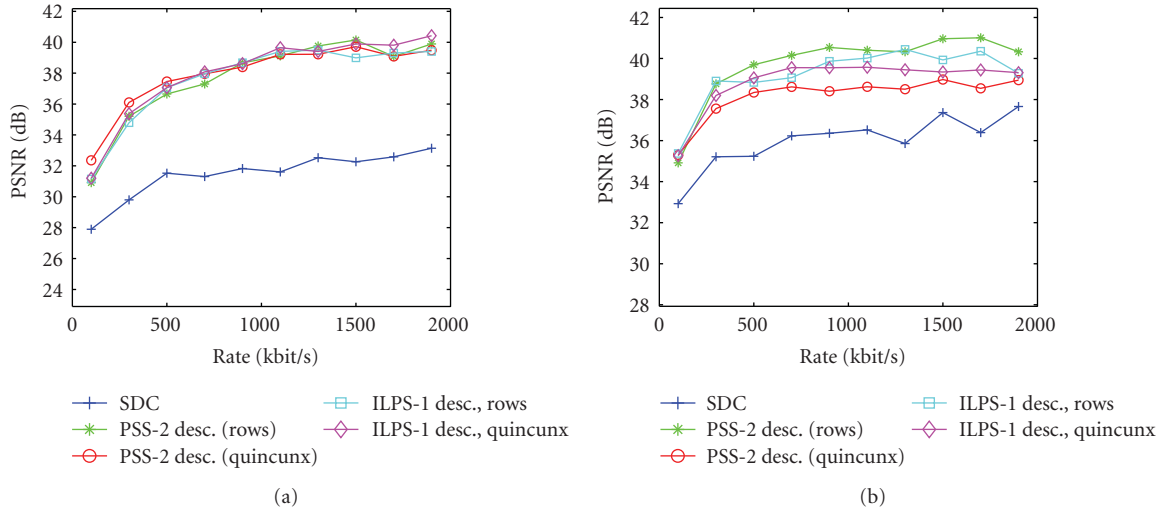



Figure 14: Performance of color view, two subsequences received, 10% packet loss. (a) Breakdance. (b) Ballet.

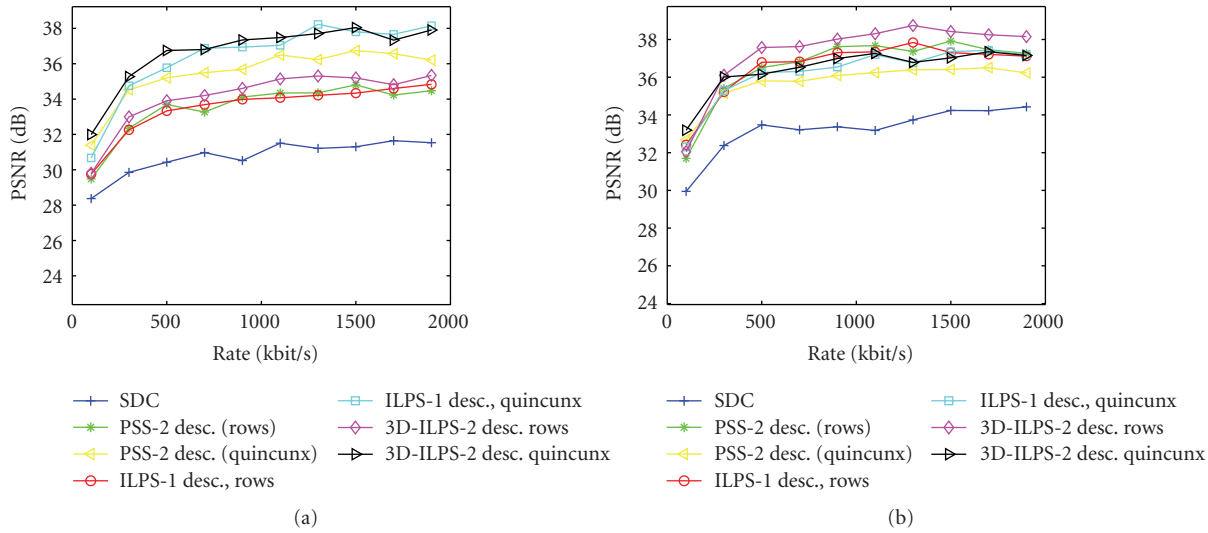



Figure 15: Performance of left view, two subsequences received, 10% packet loss. (a) Breakdance. (b) Ballet.

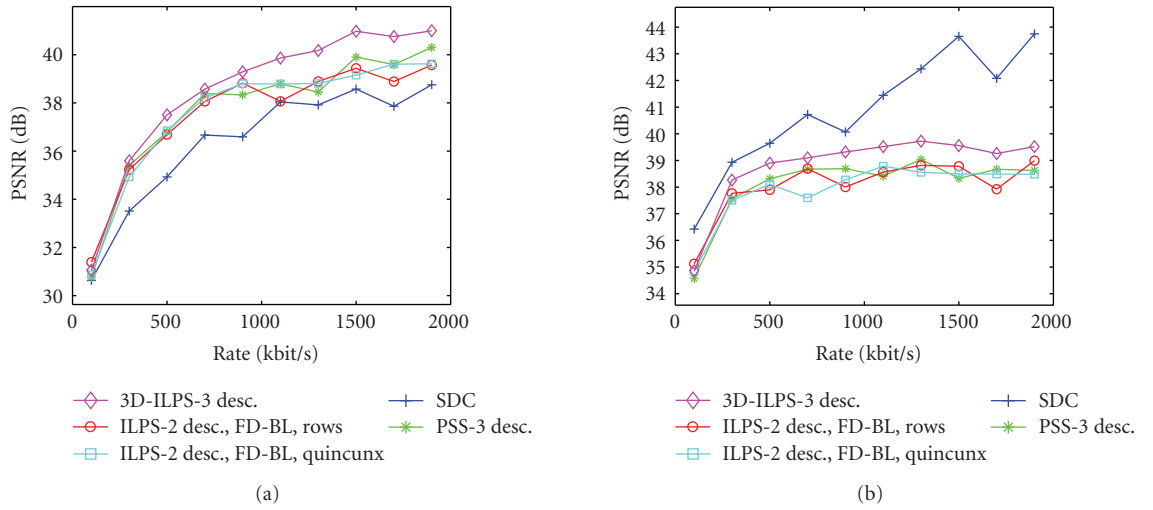



Figure 16: Performance of color view, three subsequences received, 10% packet loss. (a) Breakdance. (b) Ballet.

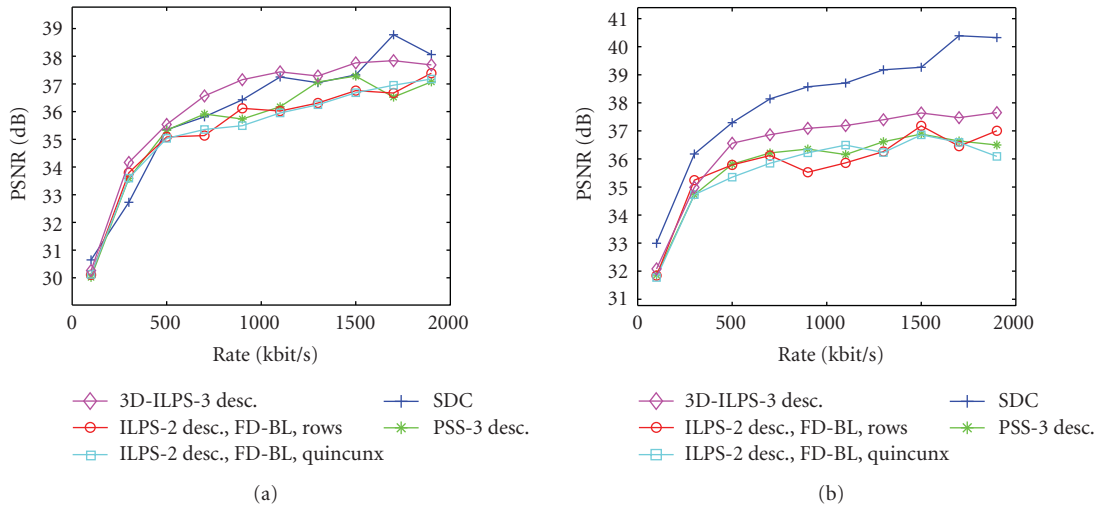



Figure 17: Performance of left view, three subsequences received, 10% packet loss. (a) Breakdance. (b) Ballet.

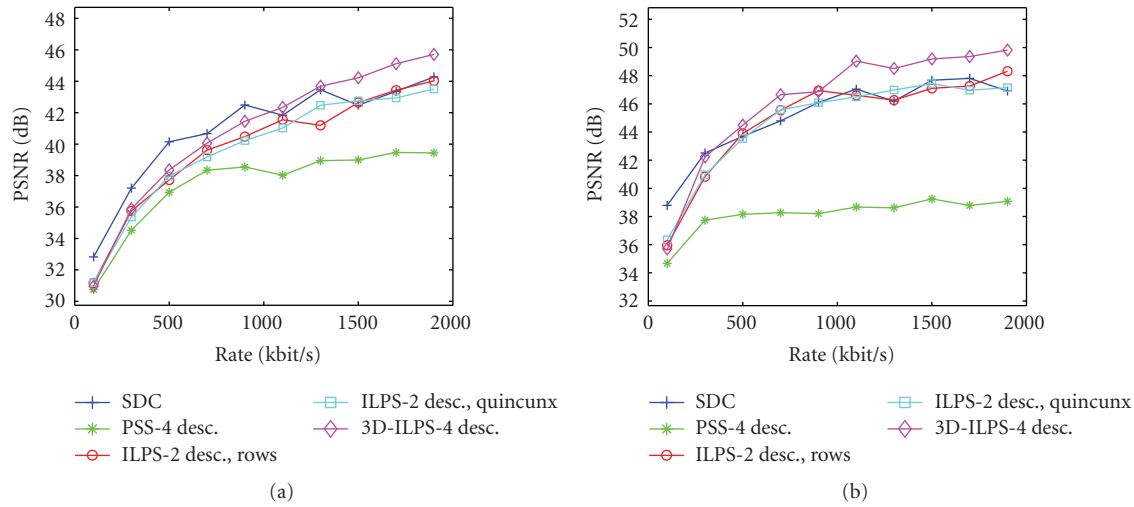



Figure 18: Performance of color view, all subsequences received, 10% packet loss. (a) Breakdance. (b) Ballet.

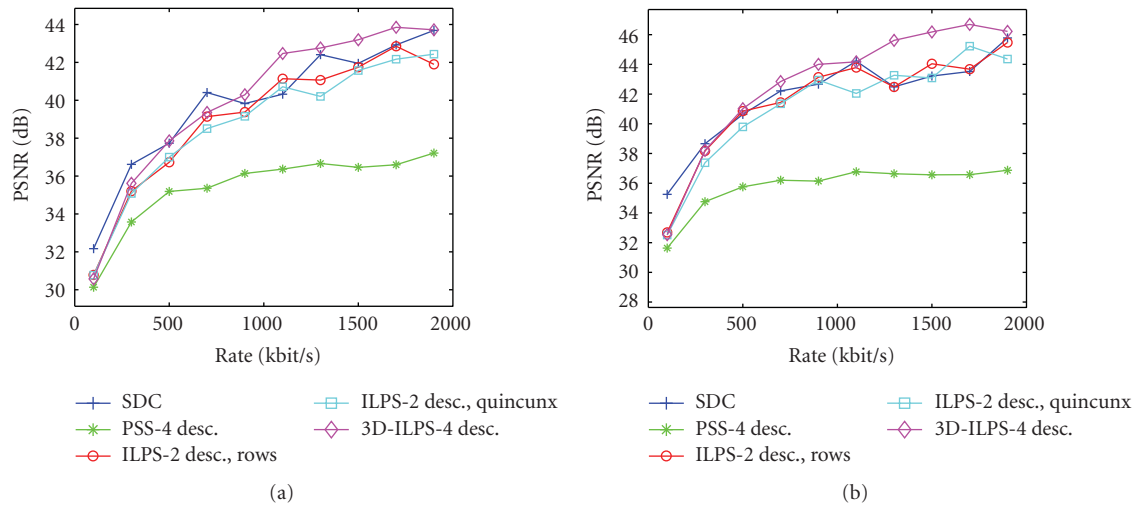



Figure 19: Performance of left view, four subsequences received, 10% packet loss. (a) Breakdance. (b) Ballet.

Figure 20: Image from Breakdance sequence. Left to right: color view, depth map, right view, left view. to bottom: SVC, PSS, ILPS, 3D-ILPS. All information received.

Figure 21: Image from Breakdance sequence. Left to right: color view, depth map, right view, left view. Top to bottom: SVC, PSS, ILPS, 3D-ILPS. All information received.

descriptions from the original sequence that can be sent over different channels or with different error protection schemes.

*6.3. Subjective Tests.* The "objective" results provided by PSNR are complemented by a set of subjective results. We first report, in Figure 20, images from the Breakdance sequence coded at an overall rate of 900 Kbit/sec for the proposed methods. It is supposed that no losses have occurred and it can be seen that the quality of all the algorithms cannot be discriminated (and this is true also at regular size). Figure 21 reports, on the contrary, the same cases when only the base layer of the two descriptions is received.

Visual experiments have been repeated to obtain subjective measures based on the mean opinion score (MOS) that can be very useful in order to produce some sort of metric for the Quality of Experience (QoE) [31]. Although the MOS evaluation could not be performed over a large number of viewers, we have been able to select a relatively small number of people, from our and neighboring laboratories, divided in two groups of about 10 persons each: with and without experience in video coding. We report only the results for the *Color* view, as the results for the other views are very

Table 4: Performance of depth view, Breakdance, 10% packet loss.

| | One subsequence received | | | | Two subsequence received | | | |
|---|---|---|---|---|---|---|---|---|
| Rate | SDC | PSS | ILPS | 3D-ILPS | SDC | PSS | ILPS | 3D-ILPS |
| 100 | 30,92 | 33,01 | 32,96 | 33,26 | 36,86 | 33,21 | 35,45 | 37,190 |
| 300 | 32,93 | 34,40 | 34,18 | 34,52 | 40,76 | 35,03 | 38,46 | 40,45 |
| 500 | 33,34 | 34,63 | 34,63 | 34,56 | 43,71 | 35,21 | 39,91 | 41,82 |
| 700 | 35,09 | 34,58 | 34,79 | 34,75 | 44,63 | 35,36 | 41,19 | 42,43 |
| 900 | 34,42 | 34,72 | 34,90 | 34,96 | 45,74 | 35,73 | 40,93 | 42,77 |
| | Three subsequence received | | | | Four subsequence received | | | |
| Rate | SDC | PSS | ILPS | 3D-ILPS | SDC | PSS | ILPS | 3D-ILPS |
| 100 | 34,25 | 35,17 | 35,34 | 35,80 | 36,86 | 34,89 | 35,02 | 35,58 |
| 300 | 37,78 | 38,84 | 39,36 | 39,79 | 40,76 | 39,14 | 39,61 | 40,17 |
| 500 | 39,32 | 41,84 | 40,92 | 41,61 | 43,71 | 40,80 | 41,47 | 42,38 |
| 700 | 39,28 | 41,68 | 41,88 | 43,02 | 44,63 | 42,57 | 43,68 | 43,43 |
| 900 | 40,63 | 43,49 | 43,61 | 43,37 | 45,74 | 42,33 | 45,96 | 45,73 |

Table 5: Mean opinion Score without packet loss, Breakdance/Ballet, color view.

| | One description received | | | | |
|---|---|---|---|---|---|
| | PSS-MD rows | PSS-MD quinc. | ILPS rows | ILPS quinc. | 3D-ILPS |
| Viewer A | 3/3.5 | 3.5/3 | 3.5/4 | 4/3.5 | 4/4.5 |
| Viewer B | 3.5/3 | 3.5/3.5 | 4/4.5 | 3.5/3 | 4.5/4.5 |
| | Two descriptions received | | | | |
| | SDC | MVC | PSS-MD | ILPS rows | ILPS quinc. | 3D-ILPS |
| Viewer A | 4.5/5 | 5/5 | 4/4.5 | 4.5/5 | 4.5/5 | 5/5 |
| Viewer B | 5/5 | 5/5 | 4.5/4.5 | 4.5/5 | 4/4.5 | 5/5 |

Table 6: Mean Opinion Score with 10% packet loss, Breakdance/Ballet, color view.

| | One description received | | | | |
|---|---|---|---|---|---|
| | PSS-MD rows | PSS-MD quinc. | ILPS rows | ILPS quinc. | 3D-ILPS |
| Viewer A | 2.5/2 | 2/2 | 3/2.5 | 2.5/2 | 3.5/3.5 |
| Viewer B | 3/2.5 | 2/2.5 | 2.5/3 | 3/3.5 | 3/4 |
| | Two descriptions received | | | | |
| | SDC | PSS-MD | ILPS rows | ILPS quinc. | 3D-ILPS |
| Viewer A | 4/3.5 | 3/3.5 | 4/3.5 | 3.5/4 | 4/4.5 |
| Viewer B | 3.5/4 | 3.5/3.5 | 3.5/3.5 | 3.5/3.5 | 4.5/5 |

similar to the ones reported. Results are in Tables 5 and 6. It is possible to see that the proposed method seems to achieve the same or even slightly better performances than PSS-MD, and at least the same performance than the SDC and MVC, in particular with packet losses. It is important to highlight that the 3D-ILPS method seems to give the best overall subjective quality among the proposed methods.

## 7. Conclusions

In this paper we have introduced an novel algorithm to generate 3D multiple descriptions in a H.264/SVC coder, with the use of *depth* and *color* views, as well *left* and *right* views, and we have shown its performance. Provided results show its effectiveness in terms of both coding efficiency, robustness, and flexibility. Work is in progress to improve these algorithms and to introduce them in more realistic network scenarios. In addition, future works will be related to introduce Medium Grain Scalability in each layer in order to be more flexible in transmission environments with varying effective bandwidth. Work is also planned to study more specific quality measures and models to determine rate distortion functions suitable to further adapt the proposed schemes and to be used in a more extensive QoE framework.

## Acknowledgments

This work is partially supported by the Italian Ministry of Research (MUR) under Grants FIRB-RBIN043TKY "Software and Communication Platforms for High-Performance Collaborative Grid."

## References

[1] C. Fehn, "A 3D-TV system based on video plus depth information," in *Proceedings of the 37th Asilomar Conference on Signals, Systems and Computers (ACSSC '03)*, vol. 2, pp. 1529–1533, Pacific Grove, Calif, USA, November 2003.

[2] B. Kamolrat, W. A. C. Fernando, M. Mrak, and A. Kondoz, "Joint source and channel coding for 3D video with depth image-based rendering," *IEEE Transactions on Consumer Electronics*, vol. 54, no. 2, pp. 887–894, 2008.

[3] M. van der Schaar and P. A. Chou, Eds., *Multimedia over IP and Wireless Networks*, Elsevier, New York, NY, USA, 2007.

[4] ISO/IEC/JTC/SC29/WG11—ISO/IEC 13818.

[5] W. Li, "Overview of fine granularity scalability in MPEG-4 video standard," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 11, no. 3, pp. 301–317, 2001.

[6] H. Radha, M. van der Schaar, and Y. Chen, "The MPEG-4 fine-grained scalable video coding method for multimedia streamingover IP," *IEEE Transactions on Multimedia*, vol. 3, pp. 53–68, 2001.

[7] H. Schwarz, D. Marpe, and T. Wiegand, "Basic concepts for supporting spatial and SNR scalability in the scalable H.264/MPEG4-AVC extension," in *Proceedings of the International Conference on Systems, Signals and Image Processing (IWSSIP '05)*, Chalkida, Greece, September 2005.

[8] V. K. Goyal, "Multiple description coding: compression meets the network," *IEEE Signal Processing Magazine*, vol. 18, no. 5, pp. 74–93, 2001.

[9] Y. Wang, A. R. Reibman, and S. N. Lin, "Multiple description coding for video delivery," *Proceedings of the IEEE*, vol. 93, no. 1, pp. 57–69, 2005.

[10] E. Setton, P. Baccichet, and B. Girod, "Peer-to-peer live multicast: a video perspective," *Proceedings of the IEEE*, vol. 96, no. 1, pp. 25–38, 2008.

[11] M. Liu and C. Zhu, "Multiple description video coding using hierarchical B pictures," in *Proceedings of the IEEE International Conference on Multimedia and Expo (ICME '07)*, pp. 1367–1370, Beijing, China, July 2007.

[12] N. Franchi, M. Fumagalli, R. Lancini, and S. Tubaro, "Multiple description video coding for scalable and robust transmission over IP," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 15, no. 3, pp. 321–334, 2005.

[13] M. van der Schaar and D. S. Turaga, "Multiple description scalable coding using wavelet-based motion compensated temporal filtering," in *Proceedings of the International Conference on Image Processing (ICIP '03)*, vol. 3, pp. 489–492, Barcelona, Spain, September 2003.

[14] H. Bai and Y. Zhao, "Multiple description video coding based on lattice vector quantization," in *Proceedings of the 1st International Conference on Innovative Computing, Information and Control (ICICIC '06)*, vol. 2, pp. 241–244, Beijing, China, August 2006.

[15] M. Flierl and B. Girod, "Multi-view video compression—exploiting inter-image similarities," *IEEE Signal Processing Magazine*, vol. 24, no. 7, 2007.

[16] C. Fehn, R. Rarre, and S. Pastoor, "Interactive 3D-TV-concepts and key technologies," *Proceedings of the IEEE*, vol. 94, no. 3, pp. 524–538, 2006.

[17] J. K. Wolf, A. D. Wyner, and J. Ziv, "Source coding for multiple descriptions," *The Bell System Technical Journal*, vol. 59, no. 8, pp. 1417–1426, 1980.

[18] N. S. Jayant, "Subsampling of a DPCM speech channel to provide two 'self-contained' half-rate channels," *The Bell System Technical Journal*, vol. 60, no. 4, pp. 501–509, 1981.

[19] Y. Wang and S. Lin, "Error-resilient video coding using multiple description motion compensation," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 12, no. 6, pp. 438–452, 2002.

[20] M. Caramma, M. Fumagalli, and R. Lancini, "Polyphase downsampling multiple description coding for IP transmission," in *Visual Communications and Image Processing*, vol. 4310 of *Proceedings of SPIE*, pp. 545–552, San Jose, Calif, USA, January 2001.

[21] M. Yu, Z. Wenqin, G. Jiang, and Z. Yin, "An approach to 3D scalable multiple description video coding with content delivery networks," in *Proceedings of the IEEE International Workshop on VLSI Design and Video Technology (IWVDVT '05)*, pp. 191–194, Suzhou, China, May 2005.

[22] H. Schwarz, T. Hinz, H. Kirchhoffer, D. Marpe, and T. Wiegand, "Technical description of the HHI proposal for SVC CE1," *ISO/IEC JTC1/SC29/WG11*, Doc. m11244, Palma de Mallorca, Spain, October 2004.

[23] H. Mansour, P. Nasiopoulos, and V. Leung, "An efficient multiple description coding scheme for the scalable extension of H.264/AVC (SVC)," in *Proceedings of the 6th IEEE International Symposium on Signal Processing and Information Technology (ISSPIT '07)*, pp. 519–523, Vancouver, Canada, August 2007.

[24] A. Norkin, A. Aksay, C. Bilen, G. Bozdagi Akar, A. Gotchev, and J. Astola, "Schemes for multiple description coding of stereoscopic video," in *Proceedings of the International Workshop on Multimedia Content Representation, Classification and Security (MRCS '06)*, vol. 4105, pp. 730–737, Istanbul, Turkey, September 2006.

[25] H. A. Karim, C. T. E. R. Hewage, S. Worrall, and A. M. Kondoz, "Scalable multiple description video coding for stereoscopic 3D," *IEEE Transactions on Consumer Electronics*, vol. 54, no. 2, pp. 745–752, 2008.

[26] R. Schäfer, H. Schwarz, D. Marpe, T. Schierl, and T. Wiegand, "MCTF and scalability extension of H.264/AVC and its application to video transmission, storage, and surveillance," in *Visual Communications and Image Processing*, vol. 5960 of *Proceedings of SPIE*, no. 1, pp. 343–354, Beijing, China, July 2005.

[27] M. Folli, L. Favalli, and M. Lanati, "Parameter optimization for a scalable multiple description coding scheme based on spatial subsampling," in *Proceedings of the International Mobile Multimedia Communications Conference (MobiMedia '08)*, Oulu, Finland, July 2008.

[28] N. Ozbek and A. M. Tekalp, "Scalable multi-view video coding for interactive 3DTV," in *Proceedings of the IEEE International Conference on Multimedia and Expo (ICME '06)*, pp. 213–216, Toronto, Canada, July 2006.

[29] C. T. E. R. Hewage, H. A. Karim, S. Worrall, S. Dogan, and A. M. Kondoz, "Comparison of stereo video coding support in MPEG-4 MAC, H.264/AVC and H.264/SVC," in *Proceedings of the 4th IET International Conference on Visual Information Engineering (VIE '07)*, London, UK, July 2007.

[30] S. L. P. Yasakethu, C. T. E. R. Hewage, W. A. C. Fernando, and A. M. Kondoz, "Quality analysis for 3D video using 2D video quality models," *IEEE Transactions on Consumer Electronics*, vol. 54, no. 4, pp. 1969–1976, 2008.

[31] A. van Moorsel, "Metrics for the internet age: quality of experience and quality of business," in *Proceedings of the 5th Performability Workshop*, Erlangen, Germany, September 2001.

Hindawi Publishing Corporation
International Journal of Digital Multimedia Broadcasting
Volume 2010, Article ID 759323, 14 pages
doi:10.1155/2010/759323

*Review Article*

# Digital Holographic Capture and Optoelectronic Reconstruction for 3D Displays

**Damien P. Kelly,[1] David S. Monaghan,[1] Nitesh Pandey,[1] Tomasz Kozacki,[2] Aneta Michałkiewicz,[2] Grzegorz Finke,[2] Bryan M. Hennelly,[1] and Malgorzata Kujawinska[2]**

[1] *Department of Computer Science, National University of Ireland, Maynooth, Co. Kildare, Ireland*
[2] *Institute of Micromechanics and Photonics, Warsaw University of Technology, 8 A. Boboli St., 02525 Warsaw, Poland*

Correspondence should be addressed to Bryan M. Hennelly, bryanh@cs.nuim.ie

Received 27 April 2009; Revised 29 September 2009; Accepted 8 December 2009

Academic Editor: Georgios Triantafyllidis



The application of digital holography as a viable solution to 3D capture and display technology is examined. A review of the current state of the field is presented in which some of the major challenges involved in a digital holographic solution are highlighted. These challenges include (i) the removal of the DC and conjugate image terms, which are features of the holographic recording process, (ii) the reduction of speckle noise, a characteristic of a coherent imaging process, (iii) increasing the angular range of perspective of digital holograms (iv) and replaying captured and/or processed digital holograms using spatial light modulators. Each of these challenges are examined theoretically and several solutions are put forward. Experimental results are presented that demonstrate the validity of the theoretical solutions.

## 1. Introduction

3D display systems generate a great deal of public interest. The idea that any event, ranging from the trivial to the momentous, could somehow be fully recorded and the 3D scene replayed at a later time for an audience in another location is highly desirable. However, current technologies are far from this futuristic conception of 3D technology. Nevertheless recent improvements in 3D display and capture technologies have led the way for plausible pseudo-3D scenes to be recorded allowing the user to avail of a realistic 3D experience. Consider for example the development of cinema, film, TV and computer gaming over the past 30 years. Entertainment industries are constantly pushing for better 3D experiences. Whether new technologies or approaches provide a suitably realistic 3D experiences, one must consider the human perception dimension to the problem, for a more complete account we refer the reader to Chapter 17 of [1]. The consistent and continuous improvement in the quality of the sound and special effects in films is noticeably apparent. Perhaps this is even more dramatically underlined in computer gaming when one compares the improvement in graphics over the past three decades. Future improvements in 3D technology are being prepared by major players in the industry; Both Dreamworks Animation and Pixar have stated their intention to release all future films in 3D; there is an increase in the number of 3D cinemas being constructed; public screenings of 3D football matches, and so forth, are being shown in an attempt to increase public awareness of 3D technology [2]. So the question arises, how will 3D cinema translate to 3D TV?

It is a problem of content, in part, however as more people experience 3D cinema the demand for a similar 3D experience at home increases, which in turn drives the demand for 3D television content. There is a precedent for this, the changeover from black and white TV to colour TV. Several major technology companies such as Phillips and Holografika, currently have proto-types of 3D televisions that produce a convincing 3D experience. Although the current 3D technology may sometimes appear revolutionary, the fundamental science behind it is as old as film itself [3]. The central idea underpinning the 3D experience is that of stereoscopic vision [3, 4]: an observer viewing a scene sees two similar but slightly displaced versions of that scene with

the left and right eye. This information is processed by the brain and the viewer perceives a 3D scene. For a detailed introduction to some of these approaches to 3D display devices we refer the reader to [1].

Looking further into the future, another technology could potentially offer more realistic 3D capture and display: Digital Holography (DH) [5–12]. This imaging process allows the full wavefield information; amplitude and phase, to be recorded. With this information it is possible, using devices known as Spatial Light Modulators (SLM), to reconstruct the optical wavefield in another place and at another time [11–13]. Recreating the full wavefield is the only method by which an observer would be exposed to the same scene that had been recorded. While there are obstacles to achieving this goal, digital holographic capture display technology does work as shall be demonstrated by theoretical and experimental results presented in this paper.

Some of the major challenges are:

(i) the removal of the DC and conjugate image terms, which are features of the holographic recording process,

(ii) the reduction of speckle noise, a characteristic of a coherent imaging process,

(iii) increasing the angle of perspective of digital holograms, and

(iv) replaying captured and/or processed digital holograms using spatial light modulators.

In the following sections we will discuss each of these obstacles along with several different approaches that are currently being investigated to minimize their impact on image quality. While many of these issues have been addressed over the years in many different contexts, here we use a combination of these approaches in an attempt to improve holographic display techniques.

Each of the obstacles noted above must be addressed in order to capture and replay a single holographic frame. For real-time imaging however it will be necessary to capture, transmit and replay a series of frames. Real-time imaging will therefore bring challenges in additon to those already outlined. For convincing real time imaging it will be necessary to be able to replay approximately 50 frames (cinemas currently have 25 frames per second) per second. There are several potential bottlenecks to achieving this frame rate: (a) the time it takes to capture and process a frame, (b) sending data over transmission lines and finally (c) the refresh rate of the SLM devices must have a sufficiently quick to display ~50 frames per second. We note that as technology improves these timing and transmission difficulties may well recede.

In this research presented here the complex wavefield is captured in the National University of Ireland, Maynooth (NUIM), processed and transmitted to the Warsaw Technical University (WUT) where the hologram is loaded onto an SLM and reconstructed optoelectronically.

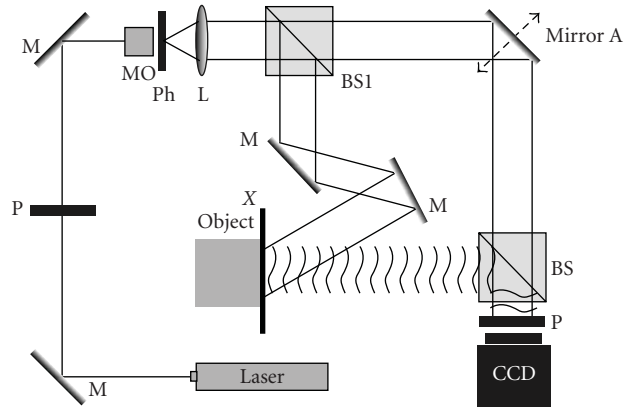



Figure 1: Schematic depicting a typical inline DH setup. M: Mirror, P: Polarizer, BS: Beam Splitter, Ph: Pinhole, and MO: Microscope Objective.

## 2. DC Terms and the Conjugate Image Term

In this section we examine some fundamental properties of a holographic imaging system. In Figure 1 we present a schematic of a typical DH setup. Laser light is spatially filtered using a microscope objective and a pinhole. The diverging spherical wave that emerges from the pinhole is collimated using a lens to form an approximately flat plane wave that is split into two beams, the reference beam and object beam, by the beam splitter BS1, see Figure 1. After reflection from several mirrors the object beam illuminates the object of interest. We refer to the field scattered from the object at Plane $X$ as $u(X)$. For the remainder of the manuscript we use the space variable $x$ to refer to the coordinate space in the capture plane and use $X$ variable to refer to the coordinate space in the reconstruction or object plane. The object wavefield $u(X)$ now propagates to the camera face where an interference pattern between the object and the reference wavefields is formed and the resulting intensity is recorded. Using the expressions $u_z(x)$ and $u_r(x)$ to refer to the object and reference wavefields, respectively, in the camera plane we can write the resulting intensity distribution as [14–17]

$$I = |u_r(x) + u_z(x)|^2, \tag{1a}$$

$$I = I_r + I_z + u_z(x)u_r^*(x) + u_z^*(x)u_r(x), \tag{1b}$$

$$I = I_r + I_z + |u_z(x)||u_r(x)|\cos[\phi(x) - \alpha], \tag{1c}$$

where

$$u_r(x) = \exp(j\alpha), \tag{2a}$$

and where

$$u_z(x) = |u_z(x)|\exp[j\phi(x)],$$
$$u_z(x) = \sqrt{\frac{1}{j\lambda z}}\int_{-\infty}^{\infty} u(X)\exp\left[\frac{j\pi}{\lambda z}(x-X)^2\right]dX. \tag{2b}$$

We note that $I_z = |u_z(x)|^2$ and $I_r = |u_r(x)|^2$. In (2b), $\lambda$ refers to the wavelength of the laser light and $z$, the distance between Plane $X$ and the camera face, see Figure 1. We also note that the relationship between $u_z(x)$ and $u(X)$ defined in (2b) is the well known Fresnel transform [17]. Examining (1b) we can see that the recorded interferogram $I(x)$ contains four terms, the DC terms: $I_z$ and $I_r$, the real image term: $u_z(x)u_r^*(x)$, and finally the conjugate image term, $u_z^*(x)u_r(x)$. We will assume that our reference wave is flat and therefore is constant for all values of $x$. This allows us to ignore the effect of the reference wave for the conjugate and real image terms as it now represents a constant phase. If we apply an inverse Fresnel transform on (1b), our image in the reconstruction plane will contain contributions from the conjugate image, the real image and the DC terms. Ideally we would like to isolate the real image term from the other terms thereby improving the quality of the reconstruction. Since the Fresnel transform is a linear operation, it is easier to understand the contribution of the different terms by considering them individually. By simulating free space propagation numerically in a computer we may back-propagate $u_z(x)$ to the object plane to recover our desired field $u(X)$. Applying an inverse Fresnel transform to the conjugate image term produces another field in the object plane which overlaps with $u(X)$, distorting the image. The DC terms, $I_r$ and $I_z$ will also play a negative role, distorting the reconstructed field in Plane $X$. We note however that if $u_r(x)$ is a flat unit amplitude plane wave, then $I_r(x)$ is constant across the capture plane and therefore is mapped by a Fourier transform to a Dirac delta function centred at zero frequency [17, 18]. This deleterious contribution may thus be filtered relatively easily using numerical techniques. Alternatively it is possible to record separately both DC terms; $I_r$ and $I_z$, and to numerically subtract these from the interferogram $I$ using a computer. In Figure 2 we present a captured and reconstructed DH of a *Starwars* character. The setup is an inline setup very similar to that depicted in Figure 1. In the experiment we use an AVT Dolphin camera with 1392 by 1048 pixels each of pitch 6.45 $\mu$m to record the interferogram while the parameters $\lambda$ and $z$ are given by $\lambda = 785$ nm, $z = 108$ mm, respectively. Performing a numerical inverse Fresnel transform on the non-processed interferogram generates the reconstructed hologram in Figure 2 [19, 20]. The DC terms and the conjugate image image noise are clearly evident. In Figure 3 we present the same reconstruction where the $I_{\text{ref}}$ DC term has been removed using a filtering technique. We reduce the power in our object arm to a low level and ignore the contribution of the $I_z$ term. From experimental experience we have found that this does not overly degrade the quality of the reconstructed hologram. Note the subsequent improvement in the reconstruction quality. Once the DC terms have been removed (and assuming a flat reference wave) we may rewrite (1c) as

$$I = u_z(x) + u_z^*(x) \tag{3}$$

On inspection of Figure 3, however it is clear that the conjugate image is still present and that it contributes to a significant deterioration of the reconstructed hologram. Therefore we now turn our attention to several techniques for removing this troublesome term.

Figure 2: Reconstructed DH of a Starwars schart. Note the presence of the both the DC terms and the conjugate image term.

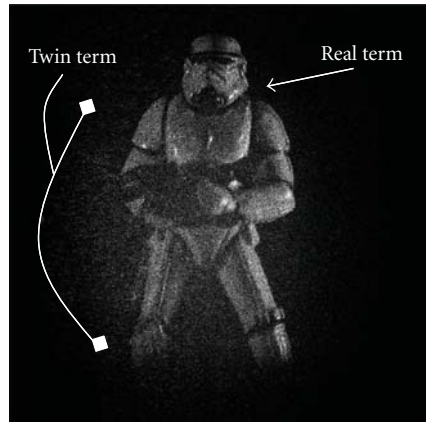
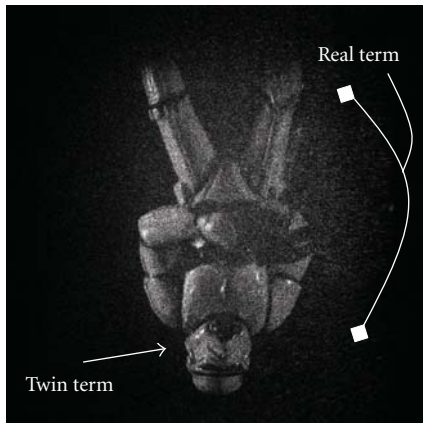



Figure 3: Same as Figure 2 however the DC terms have been removed using numerical processing techniques.

*2.1. Numerical Removal of the Conjugate Image Term.* Here we describe a technique for removing the conjugate image term numerically and refer the reader to [21] for a more complete description of the process. The basic idea behind this numerical conjugate image removal technique is to bring the conjugate image term into focus (where its features are sharp) and then to use edge detection techniques to remove the conjugate image term. We can achieve this by performing a forward numerical Fresnel transform on the interferogram. Performing a forward Fresnel transform on our previously captured hologram produces the result presented in Figure 4(a). Note that now the conjugate image term is in-focus. One can see by comparing Figures 2 and 4(a) that the conjugate and real image terms are inverted with respect to each other. In this particular example we wish to demonstrate the robustness of this approach. By




(a)

(b)

Figure 4: (a) This reconstruction is processed using the same hologram as that used to produce Figures 2 and 3. In this instance we forward propagate the hologram by a distance $+z$. In this instance the conjugate image is in focus and the real image term is out of focus. (b) This is the filter we use to remove the in-focus conjugate image.

simply setting to zero most of the in-focus term, see Figure 4(b), we may examine how much of the conjugate image has been removed. Clearly we have also removed information pertaining to the real image term; however we have attempted to discriminate more strongly against the in-focus conjugate image term. This approach can be further improved using sophisticated edge detection techniques. We now perform an inverse Fresnel transform on this processed hologram, a distance $2z$ to get back to the real image plane. In Figure 5 we present the result. While we note that the conjugate image term has not been completely removed, it has been nevertheless been significantly reduced. There are many different numerical techniques for removing the conjugate image term; see, for example, [22]. The main advantage of a numerical procedure for removing both the DC and conjugate image terms is that only one interferogram is then needed to capture our field $u(X)$. In the next section we will look at another technique for removing the conjugate

Figure 5: This reconstruction is the same as Figure 2, however both the conjugate image term and the DC terms have been removed using numerical processes described in the text.

image term that is generally more effective than the technique just described. This approach is known as phase shifting interferometric technique; however its main disadvantage is that several holograms need to be captured which may limit its application in real-time imaging [23–25].

*2.2. Phase Shifting Interferometer to Remove the Conjugate Image Term.* In this section we examine another technique for removing the conjugate image term. This approach, Phase Shifting Interferometry (PSI), is a technique in which the phase of one of the interfering beams in the setup is precisely changed and a set of phase shifted interferograms are recorded. These interferograms are then processed to extract the real object wavefield. In holography this means that the zero-order and the conjugate image noise can be removed. The advent of accurate piezo electric actuator technology has enabled this technique to become very useful in optical interferometry. Phase shifting applied to digital holography was applied to opaque objects by Yamaguchi and Zhang [23]. In the lab in NUIM, we have implemented controlled phase shifting to remove the zero-order and conjugate image reconstructions. A mirror (see Mirror A in Figure 1) is attached to a piezo-actuator (Piezosystem Jena, PZ38CAP) and placed in the object beam arm [23]. This mirror can be moved in a controlled and repeatable manner through very small displacements $\sim$1 nm. This allows us to introduce a controlled phase shift between the object and reference arms of the digital holographic setup. To ensure that we introduce the intended phase shift it is necessary to calibrate the piezo-actuated mirror. To perform this calibration we replace the object in Figure 1 with a plane mirror and observe the interference pattern in the camera plane. With ideal lenses and aligned optics we would expect to observe a uniform intensity pattern across the camera face. As the piezo-actuated mirror is subject to a displacement we expect the uniform intensity to vary from a maximum (when the object and reference plane waves exhibit total constructive interference) to minimum when the two beams exhibit destructive interference. We note however that with

misaligned optical elements and imperfect lenses we do not achieve this type of interference pattern. To perform our experiment we examined a region of the interferogram, summing the intensity of all the pixels therein. We then observed the variation in the intensity in this region as the piezo-actuated mirror was stepped in small increments by applying a known voltage. In Figure 6 we present the results. With this information we are now able to use PSI techniques to remove the DC and conjugate image terms. We proceed using the 4-step technique discussed by Yamaguchi et al. and capture a series of four digital holograms where the phase difference between each successive capture is $\pi/2$. From [25] we can now express the phase of the real image term as

$$\phi(x) = \frac{I_{3\pi/2} - I_{\pi/2}}{I_0 - I_\pi}, \tag{4}$$

where $I_\alpha$ refers to a captured hologram where the object and reference fields have been shifted $\alpha$ radians with respect to each other. We note that $|u_z(x)| = \sqrt{I_z}$ may be recovered by blocking the reference arm and recording the intensity of the object field. Using this procedure we can thus separate the real image term from the other contributions of the holographic imaging process. The result of reconstructing the data produced by applying (4) can be seen in Figure 7. It is interesting to compare the results presented in Figures 5 and 7. These seem to indicate that PSI-based approach yields superior results; however this should be further qualified. Capturing PSI type holograms has several significant disadvantages that may limit its application for real time imaging: (i) several captures are required, (ii) it is very important that the scene remains motionless during the entire capture process. In practice this latter constraint can be difficult to achieve and by its very nature interferometry is particularly sensitive to any type of motion. We are currently trying to improve numerical techniques for conjugate image term removal and are using PSI holograms as benchmark to determine their effectiveness.

We would to briefly comment on what is called off-axis reference holography. In this instance an off-axis plane wave is used as a reference wave. This has the effect of spatially separating the real and conjugate image so that the two terms may be isolated and separated from each other using numerical techniques. This technique has application for real-time imaging application, [26, 27]; however it imposes restrictive constraints on the recording device. Recording using an off-axis architecture may reduce the spatial resolution of the resulting hologram by up to four fold in contrast with an inline approach. For this reason we concentrate primarily on inline recording setups in this manuscript.

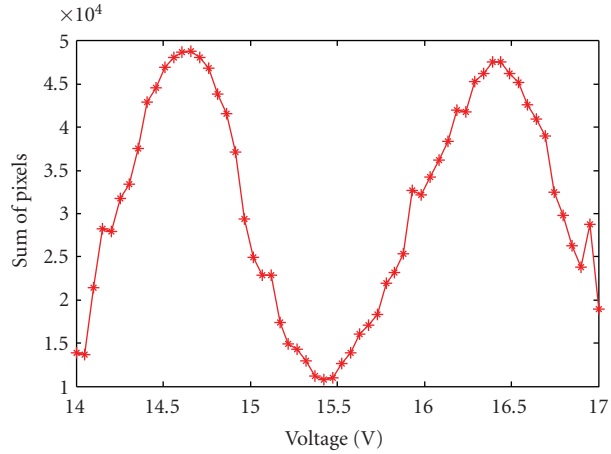



Figure 6: Calibration curve for PiezoJena actuator (Piezosystem Jena, PZ38CAP) driven using controller: NV401CLE. The intensity varies from a maximum at ~14.6 V, to a minimum at ~15.5 V, corresponding to a $\pi/2$ phase shift.

Figure 7: This reconstruction is the same as Figure 5, where both the conjugate image term and the DC terms have been removed. Unlike in the numerical procedure used in Figure 5, we have applied a PSI technique to separate out the real image term.

## 3. Speckle Noise Reduction

When light that is fairly coherent (both temporally and spatially) is reflected from an optically rough surface, the resulting intensity pattern has a random spatial intensity variation. This is known as speckle pattern. Although one of the first observations of the speckle phenomenon was made by Exner in 1878 and discussed widely at the time in the literature, it was the invention of the laser that led researchers to "re-discover" speckle in the 1960s. As lasers became more widely available, the speckle effect was soon put to good use in nondestructive, noncontact metrology applications. Nevertheless speckle is a significant hindrance in imaging applications and acts to severely degrade the performance of coherent optical systems. For imaging applications, particularly those that involve display or projection systems, the effect is striking, uncomfortable and irritating due in large part to the high contrast (unity contrast) associated with a speckle distributions. Thus speckle is both a fundamental property of a coherent imaging process and a significant hindrance for imaging applications. Over the years many different techniques for reducing speckle have been developed, unfortunately however these techniques

tend to reduce the resolution of the imaging system [28–30]. For example we have examined using mean filtering to reduce the appearance of speckles in reconstructed digital holograms. This approach reduces the speckle contrast at the expense of effectively low-pass filtering our image. There is therefore a corresponding reduction in higher spatial frequency content. Alternatively if several speckle fields are summed on an intensity basis the speckle contrast can be reduced significantly, see for example the discussion in Chapter 5 of [29]. It is of fundamental importance to realise that speckle fields added on a complex amplitude basis will not reduce the speckle contrast [30], they must be added on an intensity basis. Calculating the expected reduction in speckle contrast by summing different speckle fields on an intensity basis requires an involved theoretical investigation into the statistical temporal and spatial correlations between each of the different speckle intensity patterns. Furthermore, if the device detecting these intensity fields spatially and temporally integrates the instantaneous intensity, then the statistical properties of integrated speckle fields must also be considered. Again we refer the reader to [29] for more detail. However a good rule of thumb as to the expected reduction in speckle contrast is given by assuming that the speckle intensity patterns are statistically independent. With this assumption the speckle contrast, $C$ is given by

$$C = \frac{1}{\sqrt{N}}, \tag{5}$$

where $N$ is the number of statistically independent speckle intensity patterns.

In this section, we are going to examine two different approaches to speckle reduction. We will first examine how to reduce speckle using convenient numerical techniques before examining how to reduce speckle by capturing multiple holograms of the same scene.

*3.1. A Numerical Approach to Speckle Reduction.* Here we investigate a numerical technique for reducing the speckle contrast in a reconstructed digital hologram. We do this by summing together multiple intensity images, each of which contains a statistically independent speckle pattern lying on top of an unchanging object intensity distribution. Therefore from one captured hologram we need to somehow generate a relatively constant object intensity distribution and several statistically independent speckle fields. To do this we adopt a digital version [31] of an old technique proposed in the 1970's by Dainty and Welford [32], where the Fourier transform of the hologram is repeatedly spatially filtered and the resulting intensities are added together. Each spatial filter corresponds to a different rectangular band in the frequency domain. By moving the spatial filter in the Fourier domain we allow different parts of the hologram's Fourier spectrum to contribute to the holographic reconstruction. We note that by removing spatial frequencies we will reduce the resolution of our reconstructed hologram, however we will also reduce the speckle contrast in the reconstructed image. In Figure 8 we present our results. Figure 8(a) is the reconstructed hologram resulting from one intensity from only a single bandpass filter. This bandpass filter corresponds to a $348 \times 348$ pixel region in the frequency domain. In Figures 8(b)–8(f) we show the resulting image when 1, 3, 5, 7, 10 and 14 band pass filtering operations have been performed and the resulting intensities have been averaged. It is clear from examining Figure 8 that while the speckle contrast has been reduced it comes with the price of reducing the spatial resolution of the reconstructed image.

It is noticeable that in Figure 8(a) we clearly see dark horizontal and vertical fringes that overly the image of the object. These unwanted terms are due to the square bandpass filtering operation that created this image. Multiplication with a displaced rectangular function (one quarter the bandwidth of the fully reconstructed image) in the frequency domain amounts to convolution with a Sinc function [18] in the image plane. Furthermore this Sinc function is multiplied by a linear phase factor that relates to the displacement of the rectangular function. Convolution of the fully reconstructed image with this tilted Sinc function brings about the dark horizontal and vertical fringes that are shown in Figure 8(a). As we add together the intensities of different bandpassed versions of the image these fringes are averaged away since they are different for each bandpass. In Figure 8(b), after three filtered image intensities have been added together it is observed that the horizontal fringes have been visibly reduced but the vertical one remains in the image. This is due to the fact that the three bandpass filters were located vertically with respect to one another in the frequency domain. In Figure 8(c) after five intensities are added we see some reduction in the vertical fringe. This is due to the fact that the five band pass filters were composed of a column of four vertically shifted rectangular functions of size $348 \times 348$ pixels. For the fifth bandpass filter we move horizontally to the next column. This latter filter brings about the reduction in the vertical filter. As we move across all the columns and rows all the fringes are reduced.

*3.2. A Multiple Capture Approach to Speckle Reduction.* We note that the addition of the intensity of multiple hologram reconstructions has been previously demonstrated in the literature [33–37]. In [33] the illuminating source was rotated between captures in order to generate a series of reconstructions of the same static object with statistically independent speckle noise terms. The superposition of the intensity profiles reinforces the object image while the overlying speckle noise is averaged away. In [34] a similar method is used, this time the wavelength of the source is varied between captures. In [35] the angle of incidence upon the object of the illuminating source is varied between captures and in [36] a random diffuser, placed just after the source illuminating the object, is moved between captures. In [37] the authors set out to improve a Fourier Digital holography metrology system. The unwrapped phase (showing deformation topology) is averaged for multiple captures where there is a slight change in the CCD position between captures.

In our experiment, we introduce a piece of frosted glass (diffuser) into the object arm of our digital holographic setup. Standard PSI techniques are used to remove the DC and conjugate image terms. We are able to illuminate our

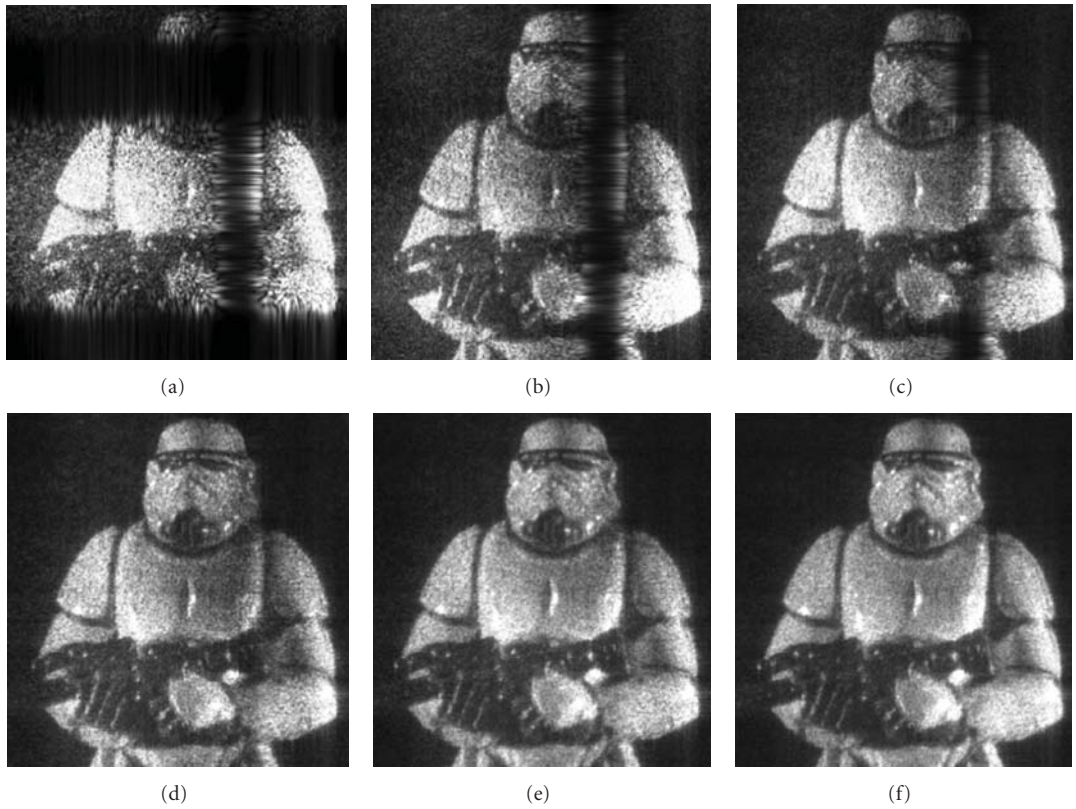



Figure 8: Removal of speckle noise using a numerical technique discussed in the text. We generate 14 different reconstructions from 1 hologram by filtering in the Fourier domain with a square aperture window of size 128 by 128 pixels. (a) One image, (b) Sum of 3 intensities, (c) Sum of 5 intensities, (d) Sum of 7 intensities, (e) Sum of 10 intensities, (f) Sum of 14 intensities.

object with a random diffuse field which is then translated approximately 5 mm between captures. This is similar to the approach that is presented in [36]. However we note that in [36] the off axis architecture was employed to record the holograms and in the results here an in-line PSI method was used. We believe that this may be the first time that a moving diffuser has been employed to reduce the speckle from PSI reconstructed holograms. One of the captured PSI holograms was reconstructed and is presented in Figure 9(a). The reconstruction is contaminated by speckle noise. We now move the piece of frosted glass and illuminate our object with a statistically different random diffuse field. Another PSI hologram is captured, reconstructed and added to Figure 9(a) on an intensity basis. Note that while speckle contrast has been reduced the resolution of the reconstructed hologram has been unaffected unlike in Figures 8(a)–8(f). A series of holograms were captured in a similar manner and the results are displayed in Figures 9(a)–9(f).

## 4. Increasing Perspective of DH Using a Synthetic Aperture Technique

Recently there has been growing interest in the use of Synthetic Aperture (SA) methods in order to increase the effective aperture of the recording CCD [38–44]. This in turn may allow one to generate a reconstructed image with a higher resolution [39], although we do not demonstrate this effect in this paper, and also to generate reconstructions showing a much greater range of perspective in the third dimension. SA digital holography may offer the greatest potential to record digital holograms that show sufficient 3D perspective for commercial display. In general all of the methods appearing in the literature aim to capture multiple holograms, each corresponding to a different section of the object wavefield. These can then be somehow merged together in order to obtain a larger hologram. The larger hologram will allow for a greater range of perspective to be seen as we reconstruct different windows of our SA hologram.

In [42], a method was outlined in which multiple holograms are recorded where the object is rotated between captures. The angle of rotation is small so that (i) the hologram appearing on the CCD does not decorrelate and (ii) some area of the wavefield appearing on the CCD is common to both captures. This allows correlation techniques to be used to determine accurately the change in angle between captures. Stitching is then performed in the Fourier domain to create a larger hologram. We are currently developing a variant of this method in which we place a mirror between the illuminated object and the CCD. It is the mirror that is rotated in our set-up meaning that we do not have to worry about speckle decorrelation that

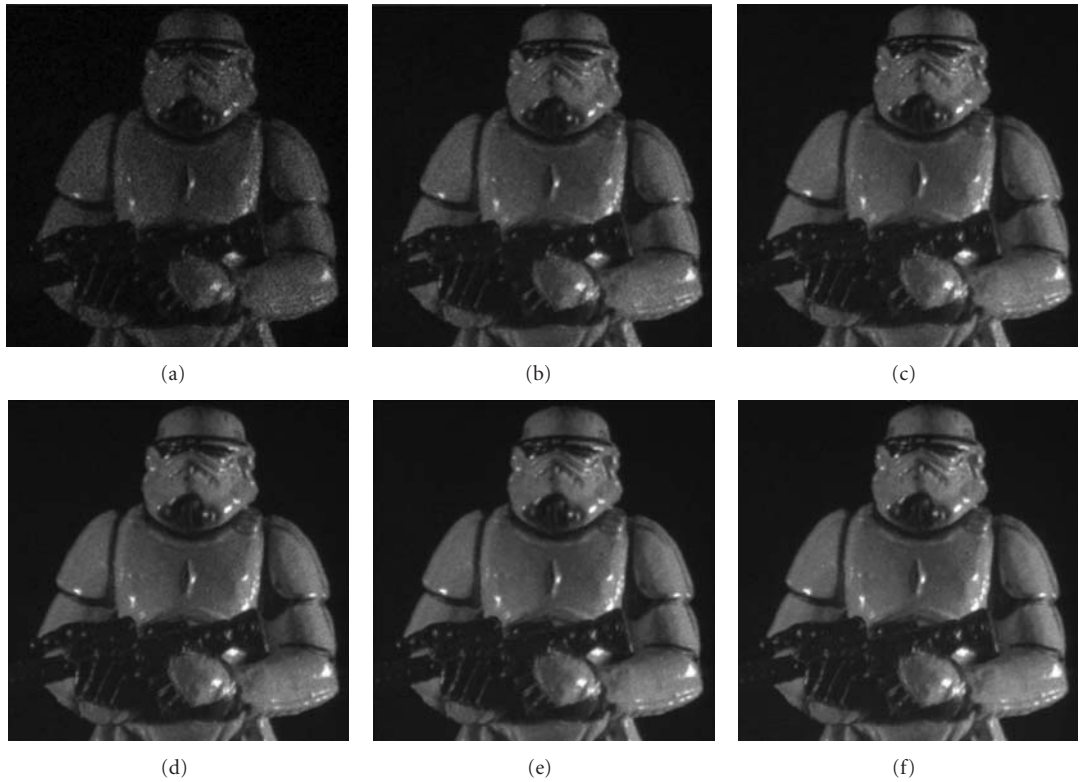



Figure 9: Removal of speckle noise by adding multiple reconstructed digital hologram captures together on an intensity basis. For each capture a different diffuse and random field is used to illuminate our object. (a) One image, (b) Sum of 3 intensities, (c) Sum of 5 intensities, (d) Sum of 7 intensities, (e) Sum of 10 intensities, (f) Sum of 14 intensities.

arises due to rotation of the object. However, the mirror creates the need for a correlation and stitching in the spatial domain. Therefore, we stitch our holograms together in four dimensions, in space ($x$ and $y$) and spatial frequency ($f_x$ and $f_y$). The shift between the two discrete hologram matrices is resolved to subpixel accuracy. We do this by applying a phase shift (corresponding to less than the pixel size, e.g., 1/5th of the pixel size) to one of the matrices in the Fourier domain. Since a phase shift in the Fourier domain corresponds to a shift in space in the space domain the matrix is said to be subpixel shifted in the space domain. This is equivalent to interpolating the discrete object wavefield "in-between" the pixels. We perform the correlations on each such subpixel shifted matrix and identify the best shifted matrix which gives the highest correlation peak value. This is further done for different values of the subpixel shift, for example, (1/5th of pixel, 1/8th of pixel, etc.). The best shift $d$ is chosen and thus resolved to subpixel accuracy. The steps taken for stitching are described in the schematic in Figure 10. We refer the reader to [42] for a thorough analysis of the method.

In Figures 11 and 12 we show some preliminary results. In Figure 11 we show a digital hologram of a resolution chart and the reconstruction. In Figure 12, we capture five holograms of the same static object, while rotating a mirror in the set-up. Each new hologram allows us to add approximately 300 new pixels to the hologram. In future work we expect to use this method to record 3D macroscopic objects with a range of perspective of approximately 10–15 degrees.

## 5. Optical Reconstruction of Digital Holograms Using with a Liquid Crystal Spatial Light Modulator

As we have seen in the previous sections in order to inspect the visual information encoded in a digital hologram we can apply numerical reconstruction algorithms. While this approach is suitable for many different applications, in particular metrology or microscopy, it is not appropriate for 3D display systems. If we wish to view the hologram in real 3D space, then the information contained in the digital hologram must be replayed using optoelectronic techniques. Briefly, some of the information contained in the captured hologram is loaded onto an SLM device. A plane wave is then reflected off the surface of the device, and an approximation to the original object wavefield is created. In our display system we use a reflective liquid crystal Spatial Light Modulator (SLM): HoloEye (model: HEO 1080 P) with $1920 \times 1080$ pixels, each with a pitch of $\Delta_{\text{rec}} = 8\,\mu\text{m}$ to approximately reconstruct the original object wavefield optically. A single SLM allows us to display either a phase (phase mode) or an amplitude (amplitude

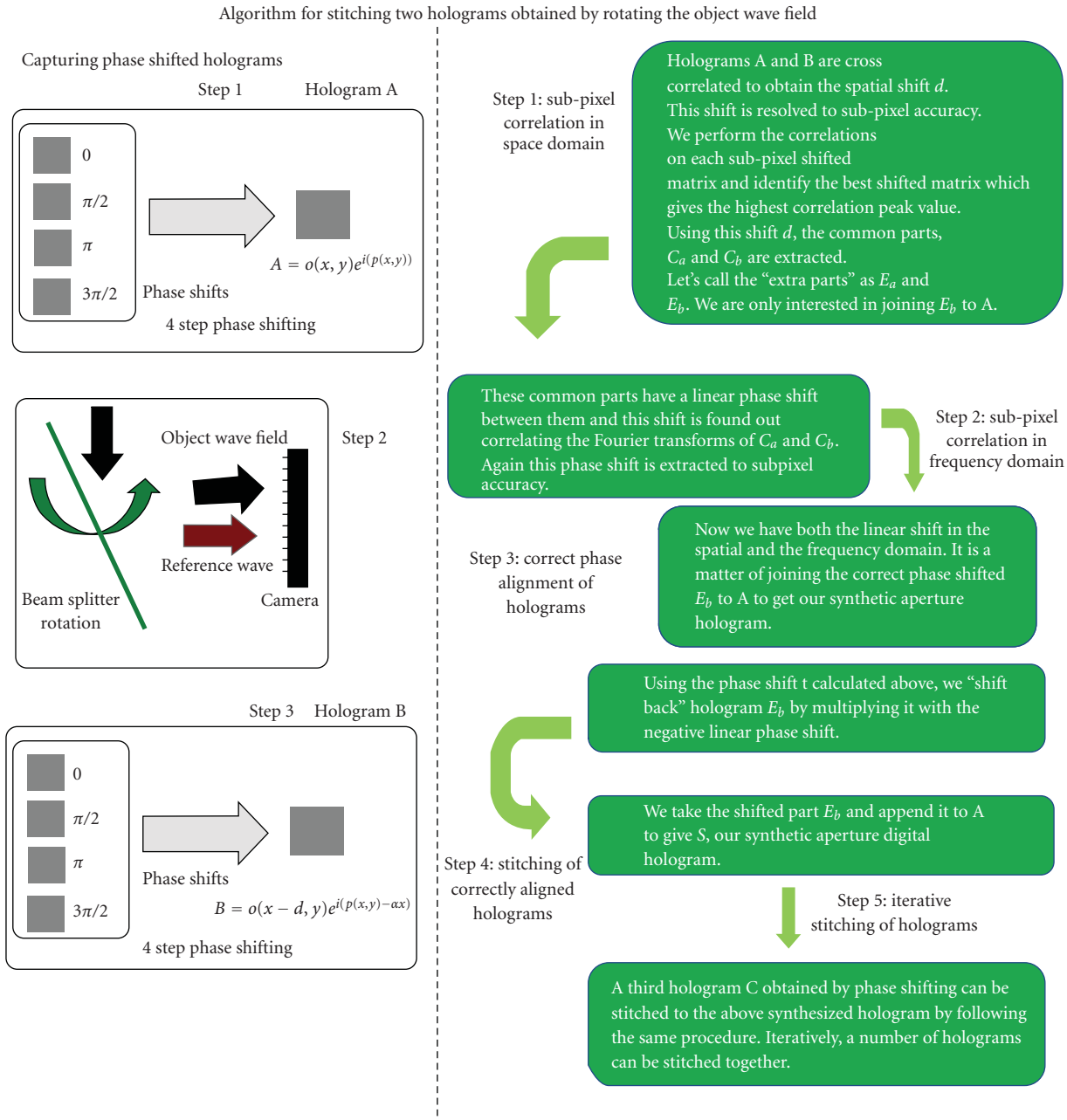



Figure 10: Schematic showing the various steps taken in the stitching process. The beam splitter in Step 2 on the left-hand side corresponds to BS in Figure 1. It is rotated in order to rotate the object wave field.

mode) distribution. Displaying both amplitude and phase is not straight-forward, as two modulators working in different modes (amplitude or phase) are required. These two SLM's would then need to be carefully aligned and furthermore would also be very sensitive to errors introduced by relative mechanical motions or vibrations between these two devices. Also, whether we use amplitude or phase mode depends on the data we load onto our SLM. We note for example that an actual interferogram (our unprocessed digital holograms that are captured in NUIM) contain only real positive numbers and therefore can be replayed using our SLM in amplitude mode. As we shall see however the resulting optoelectronic reconstruction is contaminated by both the DC terms and the conjugate image term. If we use numerical techniques to process our hologram, thereby obtaining the complex field information associated with our object distribution, we obtain both amplitude and phase data. For the alignment issues identified above we must now opt to load either the phase or amplitude data onto our SLM. From experimental experience we have found that more convincing optical reconstructions are obtained when the phase data is loaded onto our SLM and the device is operated in phase mode only, see also [17, Section 9.9.3]. Before we turn our attention to discussing some of our experimental

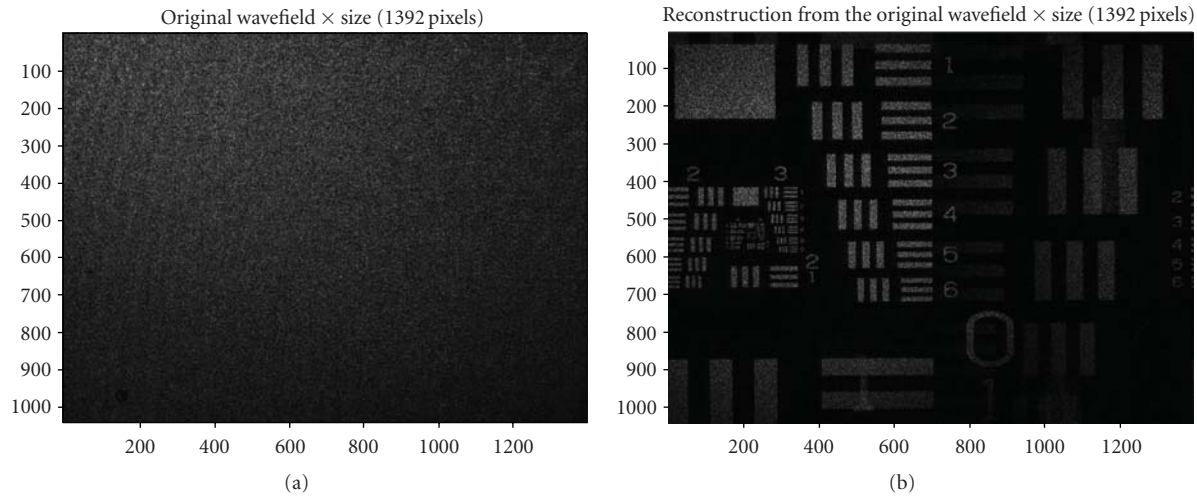



Figure 11: (a) Amplitude of a PSI hologram of a transparent USAF resolution chart having been illuminated through a diffuser. The hologram has 1040 × 1392 pixels. In (b) we show the reconstruction of the digital hologram.

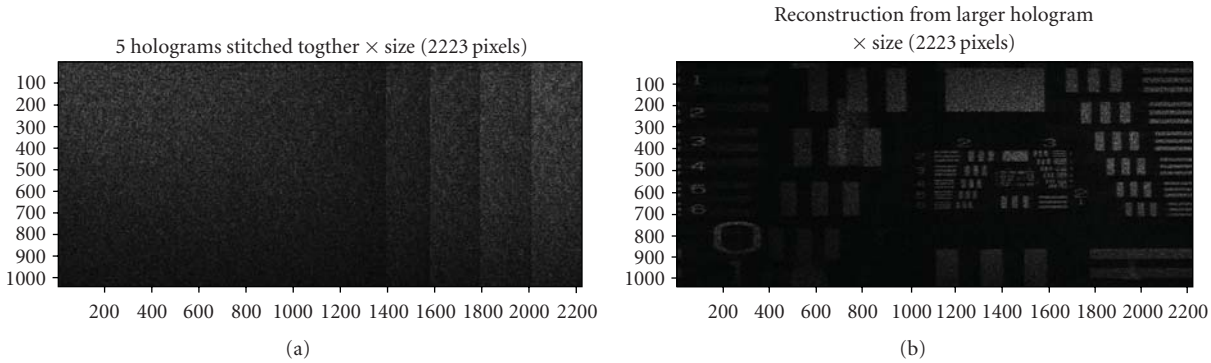



Figure 12: (a) Amplitude of a Synthetic Aperture digital hologram. Five recordings have been stitched together. The resultant hologram has clear discontinuities that would not be visible in the ideal case. Stitching has also been implemented in the spatial frequency domain. Approximately 300 pixels are added with each addition capture. The SA DH has 2223 pixels in the $x$ direction. In (b) we show the reconstruction of SA DH.

results, we note that the camera used by NUIM to capture the original hologram has 1392 × 1048 pixels, with a pixel pitch $\Delta_{\text{reg}} = 6.45\,\mu\text{m}$. The mismatch between the dimensions of captured hologram and our replay device (see Section 2) effects optical reconstruction in two ways: (i) It modifies the distance of reconstruction plane to SLM plane according to the following formula:

$$z_{\text{rec}} = z_{\text{reg}} \frac{\lambda_{\text{reg}} \Delta_{\text{rec}}^2}{\lambda_{\text{rec}} \Delta_{\text{reg}}^2}, \tag{6a}$$

where the $z_{\text{reg}}$ is the distance between detector and object planes, and (ii) A transverse magnification of

$$M = \frac{\Delta_{\text{rec}}}{\Delta_{\text{reg}}}, \tag{6b}$$

is now introduced to the reconstruction image. In Figure 13 we present a schematic to illustrate the physical optical setup that is used by WUT to optoelectronically reconstruct the captured holograms. We note that the wavelengths used to capture the original wavefield and the wavelength used for reconstruction differ, that is, $\lambda_{\text{rec}} = 532\,\text{nm}$, while $\lambda_{\text{reg}} = 785\,\text{nm}$. Using (6a) and (6b) we thus find that $z_{\text{rec}} = 281.48$, $z_{\text{reg}} = 638.95\,\text{mm}$ and $M = 1.24$. In Figure 14 we present an optoelectronic reconstruction of the captured digital hologram that was used to generate Figure 2. Since this unprocessed hologram is real, it could be replayed using a single SLM operating in amplitude mode. Experimentally however we have found that a good optical reconstruction, with a higher diffraction efficiency, is obtained when we use our SLM in phase mode and refer the reader to Sections 4.4.3 and 4.4.4 of [17] for a more complete discussion of this issue. This behavior has also been verified with some preliminary numerical simulations. One of the most disturbing terms in our optoelectronic reconstruction is the $I_{\text{ref}}$ DC term, see (1a)–(1c) and Figure 14, which can be removed numerically. The DC term covers valuable space of the holographic display. This is especially significant since SLM devices have limited viewing angles ($\sim 2^\circ$) and the majority of the reconstructed image will be located within

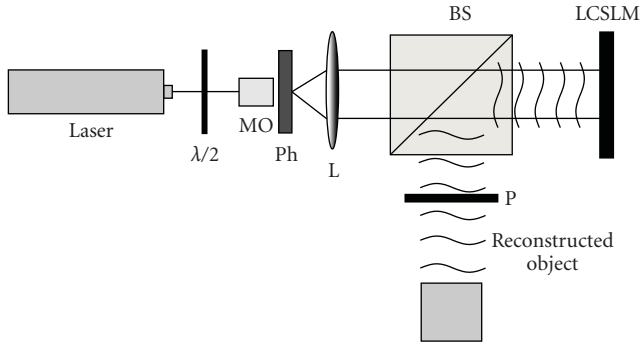



Figure 13: Scheme of setup for optical reconstruction with SLM, P: Polarizer, BS: Beam Splitter, Ph: Pinhole, and MO: Microscope Objective, L: Collimator lens, $\lambda/2$: Half-Wave Plate.

Figure 14: Optical reconstruction of holographic data with SLM: reconstruction of intensity hologram.

the zero order area, again we refer the reader to Figure 14. Also we have found that the DC term removal is especially significant for holographic reconstructions at small distances or for reconstructions of specimens with fine features.

We now turn our attention to removing the DC terms and the conjugate image term. In Section 2 we have presented two techniques for extracting the real image term from the hologram. In Section 2.1 we have presented numerical method of filtering the object field from a single hologram. The application of the method on a hologram gives us an approximation to the real image term; some features of the object field are filtered as well as the twin image term. After processing our hologram we obtain the complex information associated with the object wavefield. As we have previously noted we now discard the amplitude information displaying only the phase data on our SLM. The result of optoelectronic holographic reconstruction is presented in Figure 15(a).

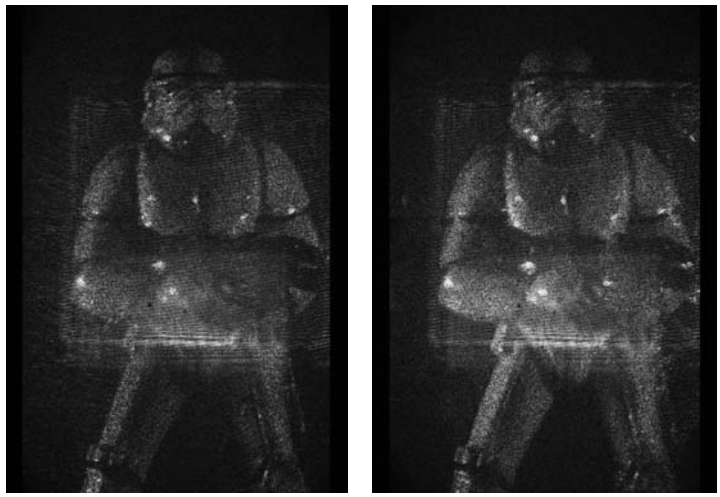
Estimating our real object field can be achieved more accurately using PSI techniques, as discussed in Section 2.2. Ideally the PSI technique returns the exact object field. Errors can be introduced due to system instability during the capture or the phase step error. In Section 2.2 we used the four frames PSI technique to recover our object field. In Figure 15(b) optoelectronic reconstruction from phase of real holographic image obtained with PSI technique is presented.

Another disturbing feature of holographic imaging technique is speckle noise. Section 3 presents a review of methods that can be used to reduce this speckle noise. The most promising is a multiple capture approach to speckle reduction. Here we apply this technique to our optoelectronic reconstruction. For our numerical reconstructions, the processing and propagation of our holograms are performed numerically. Finally the calculated intensity distributions are added together. Here however the last two steps (propagation and intensity summation) are performed optically by diffraction and then by an intensity detector, that is, our eyes or the camera we use to image our optical reconstruction. We use set of ten real object waves, filtered using our PSI technique, as described in Section 3.2. These ten phase distributions were loaded sequentially onto our SLM, at a refresh rate of 60 Hz, in a continuous manner. By adjusting the exposure time of our camera we get the CCD to average (integrate) the intensity distributions of all 10 images. The result is presented in Figure 16. Figure 16(a) shows the optoelectronic reconstruction from a single hologram, while Figure 16(b) is the reconstruction obtained from ten image set. It is clearly visible that the speckle pattern has been significantly suppressed demonstrating the usefulness of this approach for optoelectronic reconstruction.

As noted previously the spatial resolution of digital holograms may be increased using SA techniques. In Figure 17 we present some preliminary experimental results. Using SA techniques we captured a series of digital holograms that were then stitched together, see Section 4 to form a larger hologram. A section of this hologram was then selected and displayed on the SLM. When replayed optoelectronically it produces the result shown in Figure 17(a). A different section of the hologram was then loaded onto the SLM and replayed to produce Figure 17(b).

## 6. Conclusion

In this manuscript we have examined the feasibility of using digital holographic techniques to record and replay 3D scenes. We have investigated some fundamental difficulties associated with this imaging technique and suggested several theoretical solutions to overcome them. We have demonstrated the effectiveness of our solutions with experimental results. In Section 2 we examined how to remove the DC and conjugate image terms that arise due to the holographic recording process using a numerical approach and a multiple capture approach based on Phase Shifting Interferometry. In Section 3 we focused on eliminating/reducing the disturbing effect of speckle so as to minimize its negative impact on image quality. Two different approaches were again discussed, one numerical and another based on a multiple



(a) (b)

Figure 15: Optoelectronic reconstruction from phase of real object beam obtained with: (a) numerical method of real object beam filtering from single hologram, (b) 4 frames PSI technique of real object beam filtering.

(a) (b)

Figure 16: Removal of speckle noise in optoelectronic reconstruction by CCD integration of reconstructed images: (a) reconstruction from one image, (b) reconstruction from ten images.

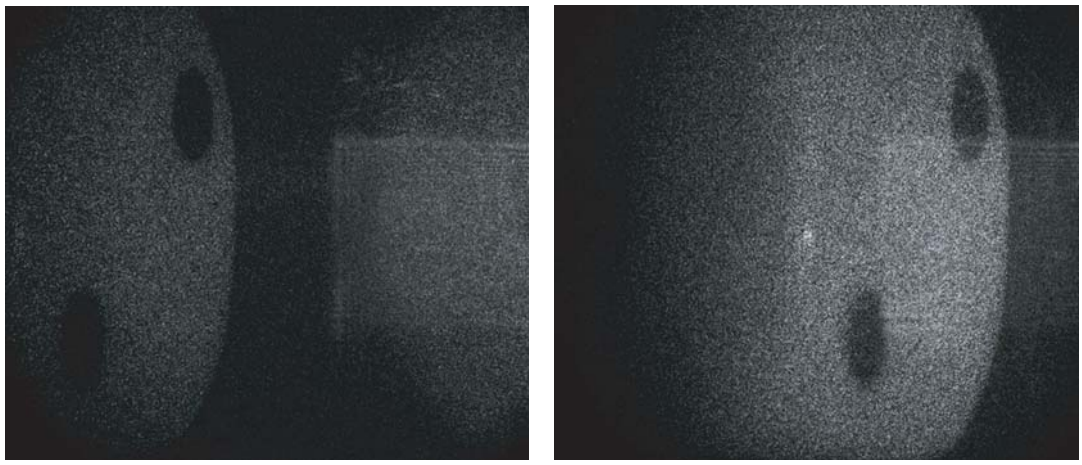


Figure 17: Reconstruction of synthetic aperture hologram.

capture technique. We do note that there are many other techniques for reducing speckle which we are continuing to investigate. In Section 4 we attempted to address the limited angular perspective of digital holograms using a synthetic aperture approach. Multiple holograms, slightly displaced from each other were recorded and stitched together using numerical techniques that we have developed. Finally in Section 5 we examined the replay or display of digital holograms using liquid crystal Spatial Light Modulators (SLM). We demonstrated experimentally that it is possible to optoelectronically replay digital holograms in one location (Poland) that have been captured elsewhere (Ireland). We have replayed synthetic aperture holograms, experimentally demonstrating the increased perspective that follows from this approach for the first time. We hope to shortly investigate the live broadcasting of 3D scenes from Ireland to Poland which may also have applications in metrology [45].

Much work remains to be done. We envisage a continued improvement in both CCD cameras and their pixel count which is essential for capturing high resolution digital holograms. We also expect to see continual improvements in spatial light modulators, both their pixel count and their ease of use and flexibility. From this collaborative research exercise we can conclude that while several major obstacles still stand in the way of real-time holographic capture and display systems, a lot of these issues will become less significant as current SLM and CCD/CMOS technologies improve. In the design of future 3D systems it will be essential to consider how the human perception effects the quality of the 3D experience. Already there are preliminary investigations underway [46]. Improvements in these areas will dramatically alter the landscape perhaps making digital holography a more feasible approach for future 3D display and capture systems.

## Acknowledgment

This research is funded from the European Community's Seventh Framework Programme FP7/2007 2013 under Grant agreement 216105 "Real 3D."

## References

[1] H. M. Ozaktas and L. Onural, *Three-Dimensional Television*, Springer, Berlin, Germany, 2008.

[2] R. Millman, "3D digital imax cinemas open for business," December 2008.

[3] N. A. Dodgson, "Autostereoscopic 3D displays," *Computer*, vol. 38, no. 8, pp. 31–36, 2005.

[4] N. A. Dodgson, "Analysis of the viewing zone of the Cambridge autostereoscopic display," *Applied Optics*, vol. 35, no. 10, pp. 1705–1710, 1996.

[5] D. Gabor, "A new microscopic principle," *Nature*, vol. 161, no. 4098, pp. 777–778, 1948.

[6] T. M. Kreis, M. Adams, and W. P. O. Jüptner, "Methods of digital holography: a comparison," in *Optical Inspection and Micromeasurements II*, vol. 3098 of *Proceedings of SPIE*, pp. 224–233, Munich, Germany, June 1997.

[7] T. M. Kreis, "Frequency analysis of digital holography," *Optical Engineering*, vol. 41, no. 4, pp. 771–778, 2002.

[8] T. M. Kreis, "Frequency analysis of digital holography with reconstruction by convolution," *Optical Engineering*, vol. 41, no. 8, pp. 1829–1839, 2002.

[9] T. M. Kreis and W. P. O. Jüptner, "Suppression of the DC term in digital holography," *Optical Engineering*, vol. 36, no. 8, pp. 2357–2360, 1997.

[10] D. P. Kelly, B. M. Hennelly, N. Pandey, T. J. Naughton, and W. T. Rhodes, "Resolution limits in practical digital holographic systems," *Optical Engineering*, vol. 48, no. 9, Article ID 095801, 13 pages, 2009.

[11] A. Michalkiewicz, M. Kujawinska, J. Krezel, L. Sałbut, X. Wang, and P. J. Bos, "Phase manipulation and optoelectronic reconstruction of digital holograms by means of LCOS spatial light modulator," in *Eighth International Symposium on Laser Metrology*, vol. 5776 of *Proceedings of SPIE*, pp. 144–152, Merida, Mexico, February 2005.

[12] U. Gopinathan, D. S. Monaghan, B. M. Hennelly, et al., "A projection system for real world three-dimensional objects using spatial light modulators," *Journal of Display Technology*, vol. 4, no. 2, pp. 254–261, 2008.

[13] D. S. Monaghan, U. Gopinathan, D. P. Kelly, T. J. Naughton, and J. T. Sheridan, "Systematic errors of an optical encryption system due to the discrete values of a spatial light modulator," *Optical Engineering*, vol. 48, Article ID 027001, 7 pages, 2009.

[14] O. Schnars and W. P. Jüptner, "Direct recording of holograms by a CCD target and numerical reconstruction," *Applied Optics*, vol. 33, no. 2, pp. 179–181, 1994.

[15] U. Schnars and W. P. Jüptner, "Digital recording and numerical reconstruction of holograms," *Measurement Science and Technology*, vol. 13, no. 9, pp. R85–R101, 2002.

[16] U. Schnars and W. P. Jüptner, *Digital Holography: Digital Recording, Numerical Reconstruction and Related Techniques*, Springer, Berlin, Germany, 2005.

[17] J. Goodman, *Introduction to Fourier Optics*, McGraw-Hill, New York, NY, USA, 2nd edition, 1966.

[18] R. Bracewell, *The Fourier Transform and Its Applications*, McGraw-Hill, New York, NY, USA, 1965.

[19] D. P. Kelly, B. M. Hennelly, W. T. Rhodes, and J. T. Sheridan, "Analytical and numerical analysis of linear optical systems," *Optical Engineering*, vol. 45, no. 8, Article ID 088201, 12 pages, 2006.

[20] B. M. Hennelly and J. T. Sheridan, "Generalizing, optimizing, and inventing numerical algorithms for the fractional Fourier, Fresnel, and linear canonical transforms," *Journal of the Optical Society of America A*, vol. 22, no. 5, pp. 917–927, 2005.

[21] C. McElhinney, B. M. Hennelly, L. Ahrenberg, and T. J. Naughton, "Removing the twin image in digital holography by segmented filtering of in-focus twin image," in *Optics and Photonics for Information Processing II*, vol. 7072 of *Proceedings of SPIE*, San Diego, Calif, USA, August 2008.

[22] T. Latychevskaia and H.-W. Fink, "Solution to the twin image problem in holography," *Physical Review Letters*, vol. 98, no. 23, Article ID 233901, 4 pages, 2007.

[23] I. Yamaguchi and T. Zhang, "Phase-shifting digital holography," *Optics Letters*, vol. 22, no. 16, pp. 1268–1270, 1997.

[24] Y.-Y. Cheng and J. C. Wyant, "Phase shifter calibration in phase-shifting interferometry," *Applied Optics*, vol. 24, pp. 3049–3052, 1985.

[25] I. Yamaguchi, J.-I. Kato, S. Ohta, and J. Mizuno, "Image formation in phase-shifting digital holography and applications to microscopy," *Applied Optics*, vol. 40, no. 34, pp. 6177–6186, 2001.

[26] T. Colomb, F. Montfort, J. Kühn, et al., "Numerical parametric lens for shifting, magnification, and complete aberration compensation in digital holographic microscopy," *Journal of the Optical Society of America A*, vol. 23, no. 12, pp. 3177–3190, 2006.

[27] S. Grilli, P. Ferraro, S. De Nicola, A. Finizio, G. Pierattini, and R. Meucci, "Whole optical wavefields reconstruction by digital holography," *Optics Express*, vol. 9, no. 6, pp. 294–302, 2001.

[28] J. W. Goodman, *Statistical Optics*, John Wiley & Sons, New York, NY, USA, 1985.

[29] J. W. Goodman, *Speckle Phenomena in Optics*, Roberts, Englewood, Colo, USA, 2007.

[30] J. W. Goodman, "Some fundamental properties of speckle," *Journal of the Optical Society of America*, vol. 66, pp. 1145–1150, 1976.

[31] J. Maycock, B. M. Hennelly, J. B. McDonald, et al., "Reduction of speckle in digital holography by discrete Fourier filtering," *Journal of the Optical Society of America A*, vol. 24, no. 6, pp. 1617–1622, 2007.

[32] J. C. Dainty and W. T. Welford, "Reduction of speckle in image plane hologram reconstruction by moving pupils," *Optics Communications*, vol. 3, no. 5, pp. 289–294, 1971.

[33] X. Kang, "An effective method for reducing speckle noise in digital holography," *Chinese Optics Letters*, vol. 6, no. 2, pp. 100–103, 2008.

[34] T. Nomura, M. Okamura, E. Nitanai, and T. Numata, "Image quality improvement of digital holography by superposition of reconstructed images obtained by multiple wavelengths," *Applied Optics*, vol. 47, no. 19, pp. D38–D43, 2008.

[35] L. Ma, H. Wang, W. Jin, and H. Jin, "Reduction of speckle noise in the reconstructed image of digital hologram," in *Holography and Diffractive Optics III*, vol. 6832 of *Proceedings of SPIE*, Beijing, China, November 2008.

[36] J. Garcia-Sucerquia, J. H. Herrera Ramírez, and R. Castaneda, "Incoherent recovering of the spatial resolution in digital holography," *Optics Communications*, vol. 260, no. 1, pp. 62–67, 2006.

[37] T. Baumbach, E. Kolenović, V. Kebbel, and W. Jüptner, "Improvement of accuracy in digital holography by use of multiple holograms," *Applied Optics*, vol. 45, no. 24, pp. 6077–6085, 2006.

[38] L. Martínez-León and B. Javidi, "Synthetic aperture single-exposure on-axis digital holography," *Optics Express*, vol. 16, no. 1, pp. 161–169, 2008.

[39] V. Mico, Z. Zalevsky, P. García-Martínez, and J. García, "Synthetic aperture superresolution with multiple off-axis holograms," *Journal of the Optical Society of America A*, vol. 23, no. 12, pp. 3162–3170, 2006.

[40] F. Le Clerc, M. Gross, and L. Collot, "Synthetic-aperture experiment in the visible with on-axis digital heterodyne holography," *Optics Letters*, vol. 26, no. 20, pp. 1550–1552, 2001.

[41] R. Binet, J. Colineau, and J.-C. Lehureau, "Short-range synthetic aperture imaging at 633 nm by digital holography," *Applied Optics*, vol. 41, no. 23, pp. 4775–4782, 2002.

[42] B. M. Hennelly, T. J. Naughton, J. B. McDonald, Y. Frauel, and B. Javidi, "A method for superresolution in digital holography," in *Optical Information Systems IV*, vol. 6311 of *Proceedings of SPIE*, San Diego, Calif, USA, August 2006.

[43] J. H. Massig, "Digital off-axis holography with a synthetic aperture," *Optics Letters*, vol. 27, no. 24, pp. 2179–2181, 2002.

[44] C. Yuan, H. Zhai, and H. Liu, "Angular multiplexing in pulsed digital holography for aperture synthesis," *Optics Letters*, vol. 33, no. 20, pp. 2356–2358, 2008.

[45] W. Osten, T. Baumbach, and W. Jüptner, "Comparative digital holography," *Optics Letters*, vol. 27, no. 20, pp. 1764–1766, 2002.

[46] T. M. Lehtimäki, K. Sääskilahti, R. Näsänen, and T. J. Naughton, "Visual perception of digital holograms on autostereoscopic displays," in *Three-Dimensional Imaging, Visualization, and Display*, vol. 7329 of *Proceedings of SPIE*, Orlando, Fla, USA, April 2009.

Hindawi Publishing Corporation
International Journal of Digital Multimedia Broadcasting
Volume 2010, Article ID 732683, 17 pages
doi:10.1155/2010/732683

*Research Article*

# Transmission of 3D Scenes over Lossy Channels

**Pietro Zanuttigh, Andrea Zanella, Federico Maguolo, and Guido Maria Cortelazzo**

*Department of Information Engineering, University of Padova, 35131 Padova, Italy*

Correspondence should be addressed to Pietro Zanuttigh, pietro.zanuttigh@dei.unipd.it

Received 26 May 2009; Accepted 23 September 2009

Academic Editor: Irene Cheng



This paper introduces a novel error correction scheme for the transmission of three-dimensional scenes over unreliable networks. We propose a novel Unequal Error Protection scheme for the transmission of depth and texture information that distributes a prefixed amount of redundancy among the various elements of the scene description in order to maximize the quality of the rendered views. This target is achieved exploiting also a new model for the estimation of the impact on the rendered views of the various geometry and texture packets which takes into account their relevance in the coded bitstream and the viewpoint required by the user. Experimental results show how the proposed scheme effectively enhances the quality of the rendered images in a typical depth-image-based rendering scenario as packets are progressively decoded/recovered by the receiver.

## 1. Introduction

Free Viewpoint Video (FVV) and 3DTV are novel research fields that aim at extending the possibilities of traditional television, allowing the viewers to watch a dynamic three-dimensional scene from any viewpoint they wish instead of just the viewpoint chosen by the director. The development of such a new service type is still at early stage; nonetheless it is expected to become a reality in the next few years and then to rapidly gain in popularity.

The realization of a 3DTV streaming service basically requires four main operations: the acquisition of the 3D scene, the compression of the data, their transmission, and finally their visualization at client side. A common assumption is that the description of a three-dimensional (static or dynamic) scene is made by two key elements, the geometry description and the color (or texture) information.

The color information can be represented by means of a set of views (or video streams for dynamic scenes) of the scene corresponding to the cameras' viewpoints. These images (or videos) are, then, compressed and transmitted by adapting the highly-scalable techniques developed for standard images and videos [1], for example, H.264 or JPEG2000.

The geometry information may be coded in different ways. Three-dimensional meshes are a common representation for geometry and many recent works focus on how to transmit them in a progressive and robust way over band-limited lossy channels [2, 3]. An alternate common solution, specially used in free viewpoint video, is to represent texture by a set of images (or videos) of the scene and geometry by depth maps. Depth maps are greyscale images that associate to each pixel a range value, corresponding to the distance between the viewpoint and the camera. This allows to reproject the available images on new viewpoints, according to the so-called "Depth-Image-Based Rendering" (DIBR) approach. This approach makes it possible to reuse the same standard image (or video) compression and transmission techniques both for texture and geometry data [4, 5], thus greatly simplifying the service architecture.

Nonetheless, the transmission and interactive browsing of 3D scenes introduces new challenges compared to standard image and video streaming. A relevant issue is that the impact of the different elements of the geometry and texture description on the rendered views dynamically changes with the viewpoint. In this regards, an open research issue is how to split the connection bitrate between texture and geometry in order to maximize the quality of the service [6, 7].

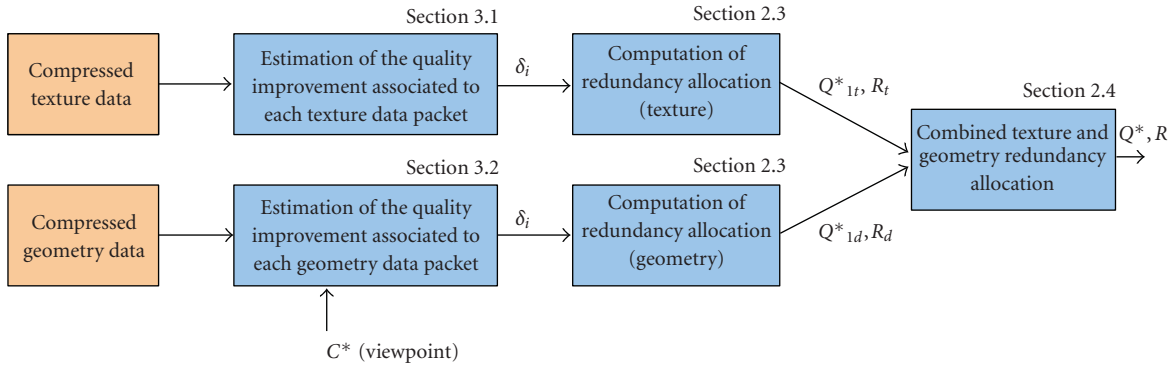



Figure 1: Block diagram of the proposed redundancy allocation scheme. The numbers over the blue boxes indicate the corresponding section in the paper.

The 3D streaming becomes even more challenging in presence of unreliable connections because packet losses may severely degrade the quality of the reconstructed content. Several methods have been proposed for the robust transmission of 3D models over lossy channels [3, 8–10] and of course many others exist for the transmission of image (texture) data [11]. However the combined transmission of both kinds of data over lossy channels is still a quite unexplored field. One of the few studies on the effects of packet losses in the combined transmission of texture and geometry data is presented in [12]. It is worth pointing out that previous literature mainly considers triangular meshes while in this paper geometry is represented as compressed depth maps. Although the transmission of 3D scenes can be performed upon either Transmission Control Protocol (TCP) or User Datagram Protocol (UDP) transport services, most of today implementations actually prefer the reliable transport service offered by TCP, which avoids the performance degradation due to packet losses. This choice is well suited to reliable and high-bitrate connections, where packet losses are rare so that the recovery and congestion control mechanisms of TCP do not impair the fluency and quality of the multimedia stream. However, in the perspective of offering 3D browsing and video services to multicast groups which reach a potentially wide population of users with heterogeneous connection capabilities, including unreliable and medium-to-low bitrate wireless connections, the reliable transport service offered by TCP will likely fail in providing the required trade off between perceived image quality and latency.

Another approach consists in abandoning the reliability of TCP in favor of a solution that employs an error recovery mechanism atop the best-effort transport service provided by UDP. A possible solution along this line consists in protecting the source data with an Unequal Error Protection (UEP) scheme, which basically consists in assigning redundancy to the different parts of the original bitstream in proportion to the importance of that part on the quality of the reconstructed view [13, 14]. UEP schemes are particularly suitable when the use of packet retransmission techniques is impractical, either because of delay constraints or in the case of multiple receivers, as for multicast transmissions. Clearly, the advantages in terms of packet loss resilience provided by UEP schemes come at the cost of an increment in the communication overhead due to the transmission of redundancy packets. Therefore, the main problem when designing a UEP scheme is to cut the best tradeoff between error protection and overhead or, from another perspective, between quality of the delivered content and latency.

This paper focuses on the transmission stage of 3D scenes over lossy channels, such as the wireless ones. We assume that both texture and geometry data are compressed in a scalable way and transmitted over a lossy channel, using the UDP transport protocol.

We first propose a UEP scheme explicitly designed for multilayer source encodings. Such a scheme is then applied to the texture plus depth map encoding technique considered in this paper. In this way we determine the distribution of a prefixed amount of redundancy between texture and geometry packets that maximizes the quality of the rendered scene in presence of losses of texture and geometry information. Figure 1 shows a block diagram of the proposed approach (the number over each block indicates the paper section describing it). The computation of redundancy allocation requires two main steps: the estimation of the impact of each data packet on the rendered images and the optimal distribution of the redundancy packets based on this information. In this paper a new model to describe the distortion on the rendered views due to texture and geometry losses in function of the required viewpoint is introduced. Its accuracy in describing the impact of texture and geometry losses on the rendered scene was assessed by a number of experiments with varying locations of the losses in the coded bitstream and different viewpoints required by the user. Such a model is used to compute the quality improvement associated to each texture and geometry data packet. This information is then used in our UEP scheme in order to compute the amount of redundancy to assign to the various scene description elements. The effectiveness of the proposed UEP scheme was tested by using an experimental testbed where a server transmits views of a 3D scene to a remote client through a connection that drops packets in a controllable manner. In order to appreciate the trade off between scene quality and latency the performance of the UEP scheme is compared against that of

a simple protection scheme, which protects only the basic layers, and that of unprotected transmission. The results reveal that even a limited number of redundancy packets, as long as appropriately allocated to the different layers, can significantly improve the quality of the reconstructed scene at every time.

In summary, the main contributions of the manuscript are threefold: the design of an UEP scheme that jointly protects texture and depth information, the definition of a simple method to estimate the relative importance of depth and texture to be used for driving the UEP scheme, and, finally, the experimental evaluation of the UEP scheme in a realistic case study.

The paper is structured as follows. Section 2 presents a UEP scheme for the transmission of three-dimensional data over lossy connections. Section 3 presents a model to estimate the impact of texture and geometry packet losses on the rendered views. Section 4 presents the experimental results and Section 5 draws the conclusions.

## 2. Unequal Error Protection Scheme for 3D Streaming

In this section we propose a scheme to allocate a prefixed redundancy budget among the different layers representing the scene, in such a way that the service quality experienced by the end user is maximized. For the sake of generality, the Unequal Error Protection (UEP) scheme is designed by considering an abstract and rather general source model, which may apply to different multimedia sources and to 3D scene browsing, in particular.

*2.1. Source Model.* We suppose that the multimedia source, generically referred to as *scene* in the following, is encoded in a basic layer plus $L-1$ enhancement layers. The basic layer carries the fundamental information from which it is possible to reconstruct a rough version of the scene. The enhancement layers, instead, contribute to progressively improve the quality of the content reconstructed at the receiver. Note that, in the case of 3D scenes transmission, this model separately applies to the texture and depth information; therefore there will be two sets of layers, one for color information and one for depth. In Section 2.4 it will be shown how to exploit this model for 3D scene browsing and in particular how to combine the depth and texture optimization procedures. Each enhancement layer is differentially encoded with respect to the previous one, so that an error in a quality layer recursively propagates to all the upper layers. For the sake of simplicity, we assume that the quality of the reconstructed scene improves only upon reception of a *complete* quality layer, whereas partial layer reception does not provide any quality enhancement. According to our experiments, this assumption is rather pessimistic since we have observed a limited quality improvement even for partially recovered layers. Therefore, the UEP scheme based on this simplified model will naturally tend to overprotect the layers with larger size, which are more likely affected by errors. Nonetheless, since the largest quality improvement comes from the lower layers which get most of the redundancy independently of their size, the impact of this approximation on redundancy allocation is not very relevant.

Let us denote by $q_i$ the function that describes the scene's quality after the correct decoding of the first $i$ layers. Function $q_i$ can be measured in terms of Peak Signal-To-Noise Ratio (PSNR), complementary Mean Squared Error (MSE), or any other metric monotonically increasing with the perceived quality. Function $q_i$ is supposed to be known by the multimedia server for each scene transmitted to the client. The actual computation of the values of $q_i$ for each element of the texture and depth information bitstreams is the subject of Section 3.

*2.2. Transmission Model.* We focus our attention on a unicast connection (or a single path of a multicast distribution tree). We assume that data are transmitted using the UDP transport protocol, in order to avoid the unpredictable delay that may derive from the congestion-control and loss-recovery mechanisms of TCP. Differential encoding generally yields to quality layers of different size. We assume that data are divided in packets of equal size $L$ and we denote by $k_i$ the number of packets of layer $i$. Therefore, the transmission of the *source* data for the whole scene all together requires

$$K = \sum_{i=1}^{L} k_i \tag{1}$$

packets. We assume that each packet can be lost with constant probability $P_{\text{loss}}$, independently of the other packets, according to the classical Bernoulli model. This type of error pattern may be observed, for instance, in case of a congested path where intermediate routers implement Random Early Dropping techniques [15, 16], or in presence of an unreliable wireless link. Every lost packet represents an *erasure*, that is, a missing packet in the stream. Erasures are easier to deal with than bit errors since the exact position of missing data is known, thanks to the sequence numbers that can be added by the transmission protocol as part of the UDP datagram payload. (When the transmission is realized by using Real Time Protocol (RTP) or the JPIP protocol, packets numbering is natively provided.) In order to increase resilience to erasures, the multimedia server adds to the $k_i$ data packets of each layer $i$ further $r_i$ redundancy packets obtained by a systematic Forward Error Correction (FEC) code for erasure channels [17]. In this way, *any* subset of $k_i$ packets suffices to reconstruct the source data. In other words, this code allows the receiver to recover from up to $r_i$ packet losses in the group of $k_i + r_i$ encoded packets of layer $i$. When more than $r_i$ packets are lost, no erasure can be recovered. (Note that, whereas the mathematical model here considered for redundancy allocation simply ignores these packets, the software tools used for the experimental results described in Section 4 do exploit even partially received layers.) Hence with independent packet losses, the probability of complete recover of layer $i$ is equal to

$$p_i(r_i) = \sum_{\ell=0}^{r_i} \binom{k_i + r_i}{\ell} P_{\text{loss}}^{\ell} (1 - P_{\text{loss}})^{k_i + r_i - \ell}. \tag{2}$$

*2.3. Redundancy Allocation Algorithm.* In order to strike a balance between overhead and robustness to erasures, we set to $R$ the total number of redundancy packets for each scene. The problem, then, is how to best distribute this redundancy among the $L$ layers.

Here, the optimality criterion we consider consists in maximizing the average quality level of the scene reconstructed by the receiver. According to our source model, scene quality progressively increases with the reception of the different layers and stops when a layer is not completely recovered by the receiver. Let $\delta_i$ denote the quality increment associated to the $i$th layer, so that $\delta_1 = q_1$ and $\delta_i = q_i - q_{i-1}$ for $i = 2, 3, \ldots, L$. Furthermore, let $\chi_i$ be the indicator function of event {Layer $i$ is correctly decoded}, which is equal to one if the event holds true and zero otherwise. The quality of the scene reconstructed by the receiver can then be expressed as

$$\widetilde{q}_1\left(\boldsymbol{\rho}_1\right) = \delta_1\chi_1 + \delta_2\chi_1\chi_2 + \cdots + \delta_L\chi_1\chi_2\cdots\chi_L = \sum_{i=1}^{L}\delta_i\prod_{j=1}^{i}\chi_j, \tag{3}$$

where $\boldsymbol{\rho}_1 = [r_1, r_2, \ldots, r_L]$ denotes the vector of redundancy packets allocated to layers 1 to $L$. Taking the expectation of (3) we then get the *mean* scene quality at the receiver after layers 1 to $L$ have been processed, that is,

$$Q_1\left(\boldsymbol{\rho}_1\right) = \mathrm{E}\left[\widetilde{q}_1\left(\boldsymbol{\rho}_1\right)\right] = \sum_{i=1}^{L}\delta_i\prod_{j=1}^{i}\mathrm{E}\left[\chi_j\right] = \sum_{i=1}^{L}\delta_i\prod_{j=1}^{i}p_j\left(r_j\right), \tag{4}$$

where the result follows from the independence of the indicator functions $\chi_i$ and the fact that $\mathrm{E}[\chi_j] = \Pr(\chi_j) = p_j(r_j)$. The objective of the allocation algorithm is to maximize (4) by optimally allocating the $R$ redundancy packets. In other words, we wish to attain

$$Q_1^*(R) = \max_{|\boldsymbol{\rho}_1|=R} Q_1\left(\boldsymbol{\rho}_1\right) = \max_{|\boldsymbol{\rho}_1|=R}\left[\sum_{i=1}^{L}\delta_i\prod_{j=1}^{i}p_j\left(r_j\right)\right]. \tag{5}$$

where $|\boldsymbol{\rho}_1| = \sum_{i=1}^{L} r_i$. The vector $\boldsymbol{\rho}_1^* = [r_1^*, r_2^*, \ldots, r_L^*]$ that attains $Q_1^*(R)$ is named the *optimal allocation policy* of the $R$ redundancy packets. This optimization problem can be solved by dynamic programming [18]. To this end, we need to express the optimization problem (5) in a recursive form which makes it possible to obtain the general solution as combination of optimal subproblems solutions. Thus, we first rewrite (4) as

$$Q_1\left(\boldsymbol{\rho}_1\right) = p_1(r_1)\left(\delta_1 + \sum_{i=2}^{L}\delta_i\prod_{j=2}^{i}p_j\left(r_j\right)\right). \tag{6}$$

Then, by the functions $Q_h(\boldsymbol{\rho}_h)$ defined as

$$Q_h\left(\boldsymbol{\rho}_h\right) = \sum_{i=h}^{L}\delta_i\prod_{j=2}^{i}p_j\left(r_j\right), \quad \text{with } \boldsymbol{\rho}_h = [r_h, r_{h+1}, \ldots, r_L], \tag{7}$$

we can rewrite (6) in the following recursive form:

$$\begin{aligned} Q_h\left(\boldsymbol{\rho}_h\right) &= p_h(r_h)\left(\delta_h + Q_{h+1}\left(\boldsymbol{\rho}_{h+1}\right)\right), \quad h = 1, \ldots, L-1, \\ Q_L\left(\boldsymbol{\rho}_L\right) &= p_L(r_L)\delta_L, \end{aligned} \tag{8}$$

where each function $Q_h(\boldsymbol{\rho}_h)$ can be interpreted as the average quality increment due to the decoding of layers from $h$ to $L$ when the redundancy packets are allocated according to vector $\boldsymbol{\rho}_h$. The optimization problem (5) can then be formulated in the following recursive manner:

$$Q_h^*(R_h) = \max_{0\le r_h\le R_h}\left[p_h(r_h)\left(\delta_h + Q_{h+1}^*(R_h - r_h)\right)\right] \tag{9}$$

for $h = 1, 2, \ldots, L-1$, whereas

$$Q_L^*(R_L) = \max_{0\le r_L\le R_L}\left[p_h(r_L)\delta_h\right]. \tag{10}$$

Equations (9) and (10) express optimization problem (5) as a Bellman equation that can be numerically solved by backwards induction [18]. In practice, the algorithm starts by evaluating the last term of the recursion, (10), for each possible value of $R_L \in \{0, 1, 2, \ldots, R\}$. Then, it evaluates (9) for $h = L-1$ and for any value of $R_h \in \{0, 1, 2, \ldots, R\}$. The algorithm iterates backwards till it determines the best allocation of the redundancy packets for each recursion step $h = 1, 2, \ldots, L$. The policy that attains the optimal quality $Q_1^*(R)$ is the final allocation vector. A pseudocode that describes the operations performed by the algorithm is shown in Algorithm 1.

*2.4. Adapting the UEP Scheme to 3D Scene Browsing.* As mentioned above, the transmission of 3D scenes actually involves two types of data flows, namely, texture and depth map. To determine the best redundancy allocation for both texture and depth map packets we firstly apply the above described method to each stream, separately, and then we merge the results within a further optimization loop. More specifically, let $Q_{1D}^*(R_D)$ denote the quality corresponding to the optimal protection strategy for depth information as a function of the amount of redundancy $R_D$ allocated to it. Let $Q_{1T}^*(R_T)$ and $R_T$ represent the same quantities for texture data. The optimal quality $Q_1^*(R)$ achievable with $R$ redundancy packets is then given by

$$Q_1^*(R) = \max_{R_D+R_T=R} Q_{1D}^*(R_D) + Q_{1T}^*(R_T). \tag{11}$$

The optimal redundancy distribution between the two elements of the scene description is given by the couple $(R_D, R_T)$ that maximizes $Q_1^*(R)$ subject to the constraint on the total amount of redundancy packets $R_D + R_T = R$. Considering that the number of redundancy packets is limited and the algorithm of Section 2.3 is very fast, the solution can be easily found by performing an outer loop on all the $(R_D, R_T)$ couples for which $R_D + R_T = R$. The two vectors $\boldsymbol{\rho}_1^*$ corresponding to $R_D$ and $R_T$ represent then the optimal allocation policy for both data streams. Note

```
/*
Redundancy allocation algorithm
Input:
L = number of layers
D(j) = quality increment of layer j
K(j) = number of packets of layer j
Ploss = Packet loss probability
Rtot = total number of redundancy packets
Output:
Ropt(j) = redundancy packets assigned layer
Q(j,h) = overall quality provided by layers j to L when
protected by a total redundancy of h
*/
function redund (L,D,K,Ploss,Rtot)
/*Variable initialization
r(j,h) = redundancy assigned to layer j given that
layersj to L have a total budget of hredundancy
blocks */
r = 0; Q=0;
for h=0 : Rtot,
  r(L,h) = h;
  Q(L,h) = binocdf(K(L)+h,h,Ploss)*D(L);
endfor
/*Recursion*/
for j=L-1:-1:1,
  for h=0:Rtot,
    for g=0:h;
      p_jh=binocdf(K(j)+g,g,Ploss);
      Qtmp = p_jh*(D(j)+Q(j+1,h-g));
      if Qtmp > Q(j,h)
        Q(j,h) =Qtmp;
        r(j,h) = g;
      endif
    endfor
  endfor
endfor
/*Search for the overall best allocation of the total
redundancy Rtot */
Rres = Rtot;
for j=1:L,
  Qopt = 0;
  for h=0:Rres,
    if Q(j,h)>Qopt,
      Qopt = Q(j,h)
      Ropt(j) = r(j,h);
    endif
  Rres = Rres-Ropt(j);
endfor
return Q
return Ropt
/*Binomial CDF*/
function binocdf(n,m,p)
  f= n!/(m!(n-m)!)*p^m*(1-p) ^ (n-m)
  return f
```

Algorithm 1: Redundancy allocation algorithm.

that, as reported by perceptual studies, for example [19], the quality of the rendered scene is not always the sum of the values due to the two contributions. Nevertheless, experimental evidence in Section 4 (compare, e.g., Figures 19 and 20) shows how the additive model gives an estimate of the total distortion not too far from the actual values, at least for the considered cases. Therefore the approximation of (11) is sufficiently accurate, at least for the purposes of this paper (i.e., redundancy allocation). Further research will be devoted to the development of a more accurate way of combining the two quality values. In this connection it is worth noting that since the considered quality model provides just a rough approximation of the actual impact of texture and depth losses, the redundancy allocation provided by the proposed model is no longer optimal. Nonetheless, we experimentally observed that the redundancy allocation of the proposed UEP scheme is rather robust to variations of the quality measures. Therefore, in spite of its suboptimality, the proposed quality model is remarkably effective for redundancy allocation.

## 3. A Model for the Relevance of Depth and Texture Packets in 3D Streaming

The UEP scheme described in Section 2 requires the server to be able to determine the effects of the packet losses affecting different parts of the coded stream. In this section, we present a model to estimate the impact of the loss of each element of the compressed geometry and texture bitstream on the quality of the rendered images (i.e., how to estimate the $q_i$ values in the model of Section 2).

Since both texture and depth information are represented as images or videos, before analyzing the three-dimensional case, it is useful to briefly recall a couple of basic characteristics of scalable compression, found in many current schemes.

(i) The first packets to be transmitted typically correspond to the basic quality layers or to lower resolutions in multiresolution schemes. They usually have a much greater impact on visual quality than the subsequent ones. However an accurate model is rather difficult to obtain since it depends on the data that is being transmitted and on the selected compression standard (for JPEG2000 image compression an estimation strategy is presented in [20]).

(ii) In some compression standards, like JPEG2000, the image can be decoded from any subset of the codestream. However, typically the loss of a packet can affect the usefulness of the following ones. In the video transmission case it is also necessary to consider that losses in the intra frames affect also all the subsequent frames predicted from them.

*3.1. Loss of Texture Information.* As far as the loss of texture information is concerned, the only difference with standard image transmission is that in our case the images are reprojected to novel viewpoints and this process can in principle change the impact of the lost packets.

For example's sake we will illustrate this point referring to the case of JPEG2000; however similar results can be derived for other scalable compression schemes. In JPEG2000 images are decomposed by the wavelet transform into a set of subbands, and then the subband samples are quantized and compressed. The compressed data are stored in a set of code-blocks corresponding to the different subbands, spatial positions, and quality levels. The effects of a 3D warping operation on the distortion in the wavelet subbands have been analyzed in a previous work [5]. It was shown that the quantization error of a wavelet subband sample in the source view is mapped through wavelet synthesis, warping and further wavelet decomposition of the warped view to different samples and subbands of the decomposed warped image (in spatial locations close to the one of the reprojected quantized sample). The amount of distortion in the wavelet samples of the warped image can be computed exploiting a collection of precomputed weights. Without further detail, for the current discussion it suffices to note that the weight values depend both on the 3D warping operation and on the source and destination subbands of the code-block in the wavelet decomposition. The procedure to compute these weights is described in [5]. The distortion on each sample of the warped image can so be approximated by multiplying the distortion in each sample of the transmitted image (prior to warping) by the corresponding weighting coefficients and by summing all the contributions falling on that sample. The total distortion in the rendered sample at location $\mathbf{p}$ can thus be approximated by

$$D^{i\to *}_{\text{quant},d}[\mathbf{p}] = \sum_b W_{b\to d}[\mathbf{p}] \cdot D^i_b\left[\left(\mathcal{W}^i_{b\to d}\right)^{-1}(\mathbf{p})\right], \quad (12)$$

where $D^i_b[\mathbf{k}]$ denotes the mean squared distortion at location $\mathbf{k}$ in subband $b$ of the source view $V^i$ and the weights $W_{b\to d}[\mathbf{p}]$ represent how the distortion is mapped from the source subband $b$ in $V^i$ to the target subband $d$ on the rendered view $V^*$ on the basis of the surface warping operator $\mathcal{W}^i$. We are using $(\mathcal{W}^i_{b\to d})^{-1}$ to indicate the operator which maps locations $\mathbf{p}$ in the warped resolution subband $d$ back to the corresponding location in subband $b$ of $V^i$. A complete discussion of this framework can be found in [5] and is behind the scope of this paper. What is important to observe here is that the warping operation could change the impact of a lost packet, expanding the relevance of some packets and reducing the one of others. This effect is due to the fact that the image regions corresponding to some packets may shrink while other may enlarge. However the changes in the distortion measured on the rendered views due to such shrinkages or expansions are relevant only for large viewpoint shifts. Simulation results clearly show that these effects do not have such a big impact when averaged on the whole image. In typical 3D video setups where the cameras are quite close each other, the distortion on the rendered view due to a random packet loss on texture data can indeed be considered almost independent of the selected viewpoint. Future research however will also analyze the impact of warping in selecting the amount of redundancy to be assigned to the various elements.

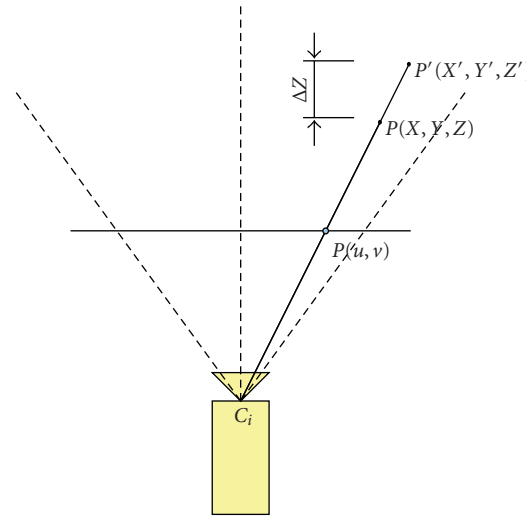



Figure 2: Pinhole camera model with depth information.

*3.2. Loss of Depth Information.* In 3D video systems depth information is used in order to allow the warping of the video stream corresponding to a camera to a novel viewpoint corresponding to a "virtual" camera that is not within the set of the available viewpoints. This is what allows the user at client side to observe the 3D scene from any viewpoint. One of the main difficulties in compressing and transmitting 3D data is to understand how uncertainty in the depth information, due to compression or network issues, affects the reconstruction of arbitrary views from novel viewpoints. Depth maps can be compressed as standard grayscale images. From image (or video) compression results it is possible to understand how packet losses or lossy compression affect the pixel values in such images (for the case of JPEG2000 an efficient estimation strategy is presented in [20]). However such a distortion should first be mapped to the representation of the scene geometry and from the scene geometry to the distortion on the rendering of novel viewpoints.

Let us denote with Cam$^i$ an available camera with optical center in $\mathbf{C^i}$ and with Cam$^*$ the target "virtual" camera with optical center in $\mathbf{C}^*$. Figure 2 shows that each pixel $\mathbf{p} = (u, v)$ in the depth map of camera Cam$^i$ corresponds to a point $\mathbf{P} = (X, Y, Z)$ in the 3D space. The depth map value represents the distance of $\mathbf{P}$ from the optical center $\mathbf{C^i}$ projected on the camera's viewing direction. An error on the depth map pixel value will cause the sample to be mapped to a different point $\mathbf{P}'$ along the optical ray. The error will so correspond to a displacement $\Delta\mathbf{P} = (\mathbf{P} - \mathbf{P}')$ of the 3D point position along the direction of the optical ray $\mathbf{C^iP}$. The 3D point is then mapped to pixel $\mathbf{p}^*$ in the image space of camera Cam$^*$ and the uncertainty in the position of the 3D point translates into a translational uncertainty $\Delta\mathbf{p}^*$ on the position of the sample within the warped image. The variance (error power) on the sample values of the depth map can be mapped to a

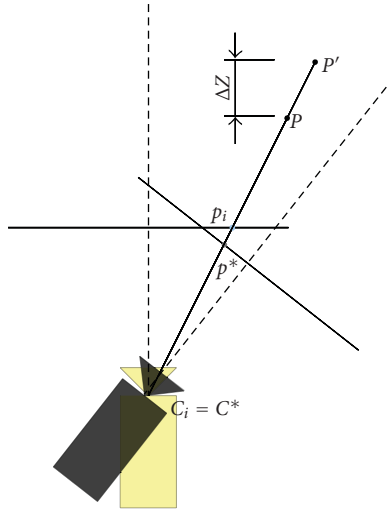



Figure 3: Rotation of the camera.

corresponding variance (error power) in the sample position by the following equation:

$$|\Delta\mathbf{p}^*|^2 = |\Delta Z[\mathbf{p}]|^2 g^{i\rightarrow *}[\mathbf{p}], \tag{13}$$

where the factor $g^{i\rightarrow *}[\mathbf{p}]$ depends upon the parameters of cameras Cam$^i$ and Cam$^*$ and on the scene geometry. A complete description of this model together with the equations for the computation of $g^{i\rightarrow *}[\mathbf{p}]$ can be found in [5].

Unfortunately this model requires complex computations. If an accurate estimate of the distortion is not required, as it is the case of this work where distortion is used only to compute the relative amount of redundancy to be assigned to depth and texture, it is possible to approximate $g^{i\rightarrow *}[\mathbf{p}]$ with simpler functions. To this end a few observations are in order.

(i) In a pure rotational camera movement (see Figure 3), depth information is not relevant and displacement $\mathbf{\Delta P}$ does not affect the reconstruction of a novel view (indeed this is the only case where the novel view can be estimated without knowledge on depth information).

(ii) A well-known result from stereo vision is that in a binocular system the projection of the same point in 3D space to two different views corresponding to a pair of cameras is shifted by an amount proportional to the distance between the two cameras (*baseline*). In particular in the simple configuration of Figure 4 it can be easily shown that the samples' shift from one view to the other is [21]

$$\mathbf{p}^* - \mathbf{p_i} = \frac{(\mathbf{C}^* - \mathbf{C^i})f}{Z} = \frac{bf}{Z}, \tag{14}$$

where $b = (\mathbf{C}^* - \mathbf{C^i})$ is the stereo system baseline and $f$ is the camera's focal length. If we introduce an error $\Delta Z = Z' - Z$ on the depth values, the difference between the position of a point in the warped image computed with the correct depth value $Z$ and the position of the same point in the warping computed with the corrupted depth value $Z'$ is

$$\Delta\mathbf{p}^* = bf\frac{1}{Z} - bf\frac{1}{Z'} = bf\frac{\Delta Z}{ZZ'}. \tag{15}$$

This equation shows that in this case $g^{i\rightarrow *}_{\mathbf{n}}$ is roughly proportional to the distance between the two camera viewpoints. Strictly speaking this holds only for cameras looking in the same direction. Furthermore term $1/(ZZ')$ leads to different shifts for objects at different distances from the viewpoint. However for small rotation angles and small camera movements in the direction of the optical ray, (15) can be assumed as a reasonable estimate of the displacement term. If a limited precision suffices, one may just average $g^{i\rightarrow *}[\mathbf{p}]$ on the whole image and assume that it varies linearly with the baseline.

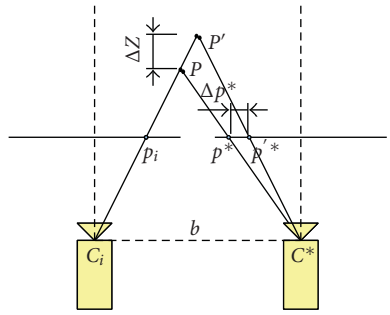



Figure 4: Translation of the camera.

As expected real configurations are much more complex and include both camera translations and rotations. Experimental results, nevertheless, indicate that assuming that the rendering distortion depends only on the depth distortion and on the distance between the two camera viewpoints is reasonable for small viewpoint changes.

The final step is the conversion of the positional uncertainty to amplitude distortion of the samples in the rendered views. This operation can be performed using the method developed in [22]:

$$D^{i\rightarrow *}[\mathbf{p}] = |\delta Z^*|^2[\mathbf{p}] \cdot g^{i\rightarrow *}[\mathbf{p}] \cdot |\omega|^2 \cdot E^{i\rightarrow *}[\mathbf{p}], \tag{16}$$

where $E^{i\rightarrow *}[\mathbf{p}]$ is a local measure of the corresponding color image variance in the vicinity of $\mathbf{p}$ and $\omega$ can be approximated by a constant factor. For the value and meaning of $\omega$ see [5, 22] where the proposed scheme is also applied independently on each of the subbands of the wavelet transform. It is worth noting that even if the cited works refer to JPEG2000, the distortion model of (16) is general and does not depend on the selected compression scheme. The distortion can then be summed up on all the image samples to get the estimation of the total distortion on the warped image corresponding to $\mathbf{C}^*$.

A final possibility is to simply warp the available view and a corrupted version of the depth information to build a novel view corresponding to a camera with the same orientation

placed at a distance $\Delta\mathbf{C} = \mathbf{C}^{\mathbf{i}} - \mathbf{C}^{*}$ from the original camera. The rendered view can then be compared with the image obtained by performing the same operation with the correct depth map. The procedure can be repeated for a limited set of viewpoints and the mean square error corresponding to all the others can be approximated on the basis of the camera's distances using the model of (15).

These approaches are very simple and provide a limited accuracy on the estimation of the distortion due to depth uncertainty; however they make it possible to build a practical real-time transmission system and provide a reasonable estimate of the relative weight of depth and texture data that can be used to select the amount of redundancy to be applied to each of them.

## 4. Experimental Results

In this section we describe the simulation environment used to test the performance of the proposed error correction scheme and we present the experimental results obtained by using both synthetic and real world multiview data with depth information. This section is organized in the following way: firstly we present the simulation environment, and then we analyze the effects of the loss of texture and depth packets. Finally, we show the performance of the proposed protection scheme.

*4.1. Remote 3D Browsing with JPEG2000 and JPIP.* As previously said many different transmission schemes for remote 3D browsing are possible. In this section for clarity's sake we briefly overview the client-server scheme of [5] which was used in order to test the performance of the algorithms presented in this paper. Let us emphasize that in spite the results of this paper were obtained with the system of [5], they are valid for every remote visualization scheme based on progressive transmission and on a depth-image based rendering scheme.

In the proposed approach the 3D scene description is available at server side as a set of images (or videos) and depth maps together with the corresponding acquisition points and camera parameters. To achieve an efficient browsing over band-limited channels, all the available information at server side, that is, both images and depth maps is compressed in a scalable way using JPEG2000. The server is able to select and transmit only the parts of the scalably compressed bitstreams that best fit the user's required viewpoint and the available bandwidth exploiting the rate-distortion optimization framework presented in [23]. The client then exploits the data received from the server to render the views required by the user during the interactive browsing. In [5] this is achieved by reprojecting all the available images onto the required viewpoint (that is usually different from the available ones) using depth information and then combining all the warped views into the requested rendering. This is achieved by using a novel multiresolution scheme based on the wavelet transform and on the estimation of the distortion on the rendered views coming from the different contributions.

The adopted transmission system relies on the JPIP interactive protocol [24] originally developed for the interactive transmission of JPEG2000 images over the internet. This protocol allows the server to decide what information needs to be transmitted to the client on the basis of the received requests. Figure 5 shows the architecture of the proposed transmission scheme and provides a framework for understanding the interaction between a JPIP server and its client. In JPIP image transmission the client just makes a request with the parameters of the view of interest (the region and resolution for an image can be replaced by the 3D viewpoint in our case) letting the server to decide the data most suited to satisfy the client's needs. The server streams JPIP messages to the client, where each JPIP message consists of a single byte-range from a single element (*data-bin*) of one of the compressed images or depth maps. JPEG2000 content can be rendered from any arbitrary subset of the precinct data-bins which might be available. This characteristic makes possible the rendering at client side completely asynchronous with respect to server communications and also makes the system robust with respect to packet losses and network issues.

*4.2. Analysis of Packet Loss Issues.* For the first test we used a synthetic scene of a cartoon character (*Goku*). In this case images and depth maps were generated from a synthetic 3D model. This makes available a ground truth and avoids the data uncertainties typical of real acquisition systems. The images were compressed in JPEG2000 at quite good quality (0.8 bit per pixel) and transmitted to the client using JPIP. For the *Goku* model we transmitted a single view with the corresponding depth map and analyzed the rendered views at 3 different viewpoints along a circle surrounding the object, respectively 5, 10, and 45 degrees apart from the original one obtaining the images of Figure 6. The camera positions are shown in Figure 7; note that the camera movement is both rotational and translational.

To test the performance of the proposed transmission scheme in a real environment we used the *Breakdance* sequence from Microsoft Research [25], commonly used to validate the performance of 3D video compression algorithms. This sequence is composed of 8 video streams acquired by a set of cameras placed along a nearly linear path (Figures 9 and 10). The dataset includes also depth information for each frame of each camera computed by stereo vision algorithms. As previously stated, all the transmitted information is compressed in JPEG2000 in a scalable way. We used 7 quality layers of increasing size. The impact of the packet losses was measured with the following procedure: the central view (Cam 4 in Figure 9) was transmitted together with the corresponding depth map and these data were used to render novel views. In particular the plots of this section show the results obtained by looking at the scene from the viewpoints of camera 3 (that is quite close to the available view) and of camera 1 (that is farther from the available viewpoint) in order to show also the dependency between the viewpoint's distances and the relevance of depth information presented in Section 3.2. To

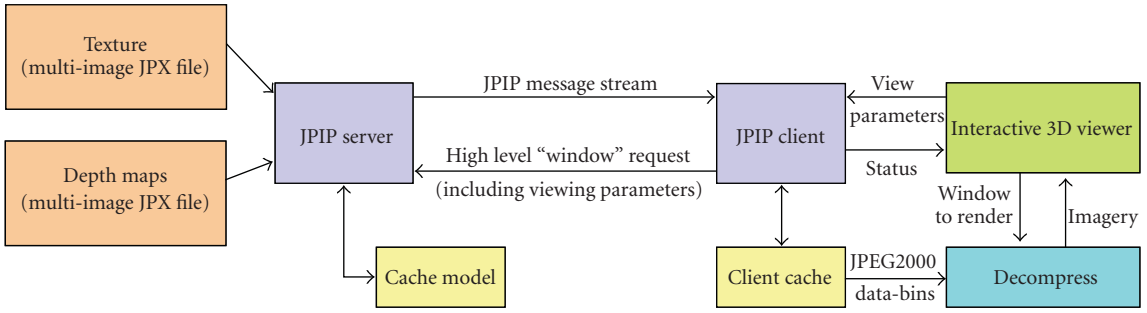
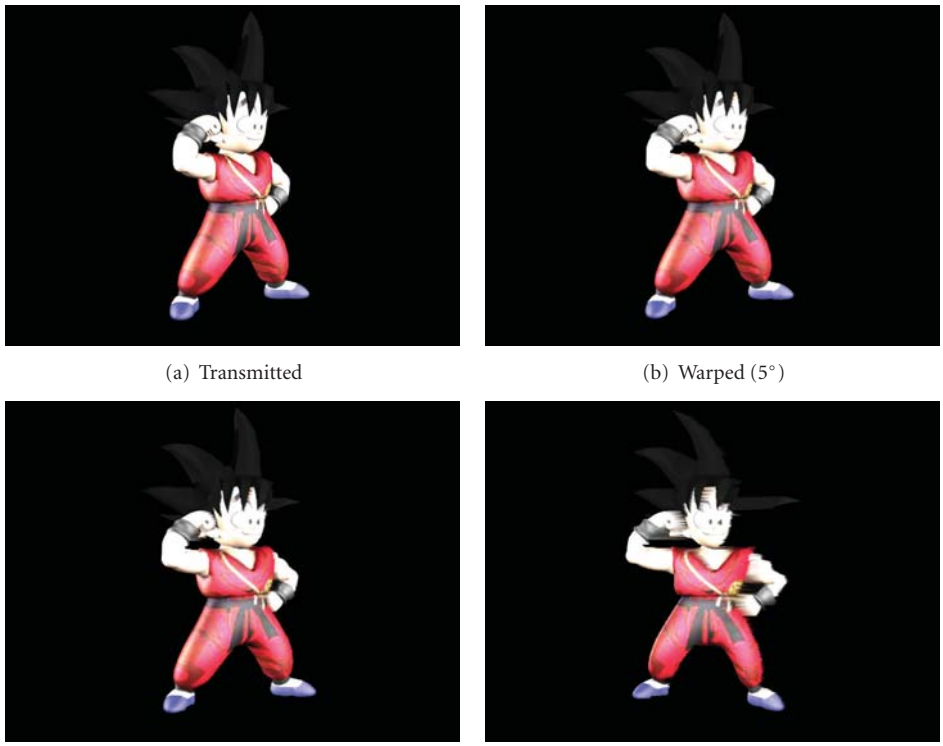



Figure 5: Architecture of the transmission system.



(a) Transmitted (b) Warped (5°)

(c) Warped (10°) (d) Warped (45°)

Figure 6: *Goku* model: transmitted image and rendered views obtained by warping the available one to novel viewpoints.

compute the rendered views' quality we took as reference the image obtained by transmitting the available data at maximum quality without any packet loss and performing the warping on these data. Even if it was possible to compare the images with the actual data from a camera in the novel viewpoint, this approach has the advantage of decoupling the loss due to transmission issues (that are the focus of this paper) from the distortion due to the warping algorithm, 3D reconstruction errors, occlusions, regions outside the reference camera field of view, and others. Examples of these issues are the black region on the side and the occluded regions close to the dancer's edge in Figure 14(b).

Figure 10 shows the PSNR corresponding to the loss of each single packet of texture information as a function of the lost packet position for the *Breakdance* data. Figure 11 instead refers to the *Goku* model and shows the distortion due to the loss of a batch of 5 consecutive texture packets. The plot shows clearly how the impact of packet losses is progressively less critical when losses occur towards the tail of the codestream. As previously said JPEG2000 divides the image in different codeblocks corresponding to different resolution levels, quality layers, and spatial regions. The JPIP server then estimates the distortion gain associated to each packet and transmits them in order of relevance. Therefore the first packets have a much greater importance than the last ones. A loss in the first layers specially in the lowest subbands data can lead to completely distorted images (Figure 12), while loss of higher resolution data can lead to blurred images. Moving towards the last quality layers the impact on the rendered views becomes less noticeable (Figure 13).

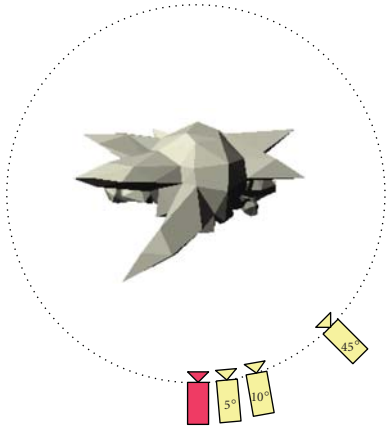



Figure 7: *Goku* model: camera positions (the available view and depth map correspond to the red camera).

(a)

(b)

Figure 8: *Breakdance* sequence: (a) sample frame of the sequence and (b) sample depth map.

In particular the loss of the first packet of the datastream (containing the main header and the key information for the first layer) results in the impossibility of decoding the whole image. This suggests that this packet must be protected with particular care. Another interesting observation is that the measured distortion is almost independent from the selected viewpoint (the plots of the warpings to viewpoints 1 and 3 are almost superimposed). Although the warping operation can affect the distortion due to the loss of each packet as shown in Section 3.2, on the average (specially if the

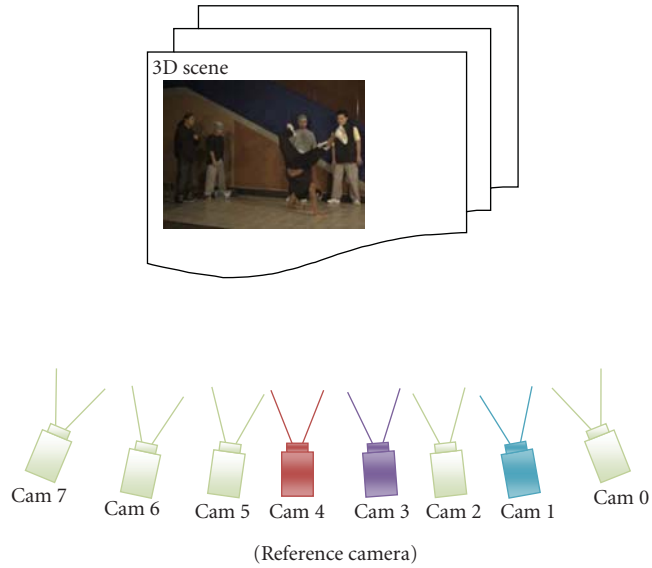



Figure 9: Camera positions for the *Breakdance* sequence (*Courtesy of Microsoft Research*). The colored cameras correspond to the viewpoints used for the experiments in this paper.

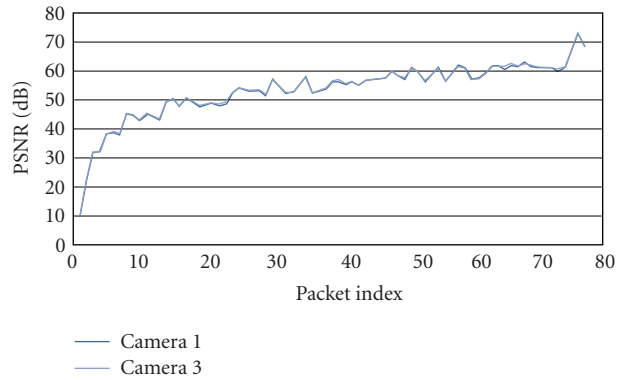



Figure 10: *Breakdance*: loss of a single packet of texture information (note how the two curves in the plot are almost superimposed).

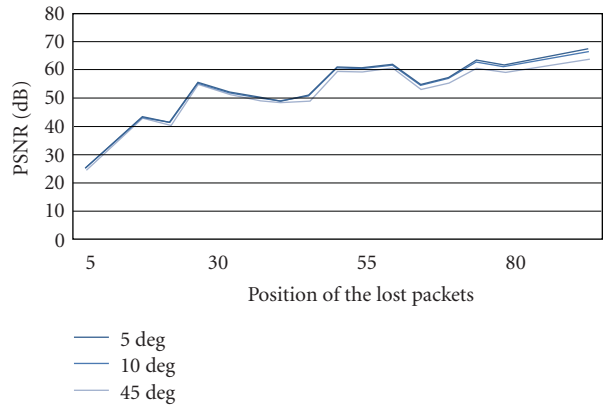



Figure 11: *Goku*: loss of a batch of 5 consecutive packets of texture information. The different curves correspond to the different viewpoints as shown in Figure 7. Note how the 5° curve is almost invisible because the 10° one is almost perfectly superimposed to it.

Figure 12: View of the *Goku* model corrupted by a loss in the fist part of the texture codestream.

viewpoint is not very far from the available ones) distortion due to texture packet losses for practical purposes can be considered independent from the selected viewpoint. Figures 14(c) and 14(d) show an example of how texture losses artefacts remain quite similar after the warping (at least for small viewpoint changes).

Figures 15 and 16 concern instead the loss of depth packets. In particular, Figure 15 refers to the *Breakdance* data and shows the distortion due to a single depth packet loss. Figure 16 shows instead the distortion due to the loss of 5 packets of the *Goku* dataset. As shown in Section 3.2, the loss of depth packets causes samples to be misplaced in the 3D space and so to be warped in wrong positions in the rendered views. This of course impacts the quality of the rendered images; Figures 17 and 18 show a couple of examples of the artefacts due to this effect. Usually the edges of the objects (corresponding to the regions with depth discontinuities) experience the major impact from depth transmission errors. Note that also in this case the plots show the distortion in the rendered images, not the displacement of the samples or other measures in the 3D space. This allows to directly compare these results with the texture ones and it is also consistent with the target of the proposed scheme that is the maximization of the rendered views' quality and not the accuracy of the three-dimensional description.

There are two key differences with respect to the previous case.

(i) The distortion still decreases with the packet index but the shape of the curve is less regular. This is due to the fact that JPIP streams the packets sorted by their impact on image distortion (therefore it considers depth maps just as regular images ignoring their 3D meaning and the way they will be used in the warping). Artefacts on the texture directly map to similar ones in the warped views, while the mapping of depth distortion to the warped views is more complex as pointed out in Section 3.2.

(ii) In the depth case the measured distortion depends on the viewpoint: when warping to farther locations the same loss on depth data leads to larger errors in the warpings as shown in Section 3.2. Figures 18(c) and 18(d) clearly show how the same loss on the depth data leads to quite different results when warping to viewpoint 3 (close) or viewpoint 1 (farther). Note also how in the *Goku* setup, where the viewpoint change is much larger, the relative relevance of depth information is rather high if compared with the *Breakdance* sequence.

*4.3. 3D Transmission over Lossy Channels with the Proposed FEC Protection Scheme.* In this section we compare the image quality corresponding to the transmission of the image and depth data with and without error protection schemes. As shown in Section 4.2 the loss of the first block prevents image decoding and has a dramatic impact on average distortion, so we decided to compare three strategies:

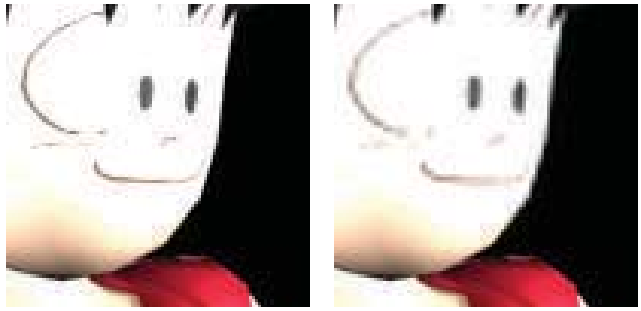
(i) Transmission without any protection scheme.

(ii) A simple protection scheme that protects only the first packet in order to ensure that the image is decodable.

(iii) The ad hoc protection scheme of Section 2.

To analyze the performance of the proposed transmission scheme we transmitted the information on the 4th view of the *breakdance* sequence over a network with random packet loss probability of 1%, 5%, and 10%. The received data are then rendered from the viewpoint of view 3 (quite close) and of view 1 (farther).

Figure 19 shows the Mean Squared Error (MSE) for a 10% random packet loss probability. The first two columns refer to the transmission of depth information (assuming that texture has been correctly received). In the case of transmission without any protection, performance is very poor because when the first packet is lost, the entire depth map becomes useless. In this case (assuming that the warping procedure cannot be performed without depth data), the MSE is equal to 650, which is an order of magnitude larger than the average MSE values obtained when the first packets are successfully decoded. Therefore, to preserve the graphs readability, we do not report in the plots the MSE for completely unprotected transmissions. Instead, our benchmark for assessing the quality gain due to smart redundancy allocation is provided by a very simple forward error correction scheme, called "PROT1," which protects only the first packet, guaranteeing its recovery probability very close to one. The performance obtained by PROT1 is shown by the right-most column of each group of bars in Figure 19. The middle column of each group ("FEC4") shows the performance of the proposed scheme with $R = 4$ redundancy packets, corresponding to a redundancy of 6% (there are 62 data packets for depth information in this frame). With just 4 packets of redundancy, most of them (three in this case) are allocated to the protection of the first packet. However the proposed protection scheme still



(a) (b)

Figure 13: *Goku*: (a) detail of the character's face and (b) same detail corrupted by the loss of the 23th packet.

provides a small gain compared to the "PROT1" approach. Such redundancy is quite limited for a 10% packet loss and better results can be obtained using $R = 8$ packets of redundancy, corresponding to a 12% of data redundancy, and shown by the left-most column denoted as "FEC8." In this way it is possible to protect a larger part of the depth codestream and the gain on the distortion in the rendered views is about 3.5 dB. Note also how the distortion due to depth information depends on the selected viewpoint and is higher for viewpoint 1 which is farther from the available view.

The second group of columns of Figure 19 instead shows the case where depth is assumed to be correctly transmitted and analyzes texture data transmission. Again the loss of the first packet does not allow image decoding and it is not shown in the plots. Due to the different organizations and meanings of the compressed data, even with a redundancy of 4 packets (out of 77 data packets, corresponding to a 5% of redundancy information) it is possible to obtain a 2 dB gain over the protection of the first packet. By doubling the number of redundancy packets the MSE is also decreased by a factor of 2. The biggest difference with the previous case is that here distortion is almost independent of the selected viewpoint.

Until this point depth and texture losses have been considered separately but one of the main targets of the proposed scheme is the allocation of the redundancy between the two kinds of data. Figure 20 concerns the case where packet loss affects both texture and geometry and the redundancy must be distributed between the two types of information. The right-most column shows the results obtained by protecting just the first packet while the other two correspond to 8 total packets of redundancy (for $62 + 77 = 139$ packets of source data) and to 16 redundancy packets. Again as expected, the MSE decreases as redundancy increases (from 15 if we protect just the first packet to 12 for $R = 8$ and to 6.5 for $R = 16$). The results depend on the selected view but not as strongly as in the depth case (the distortion in the rendered views shown in the figure is due both to texture distortion, roughly independent from the view, and to the distortion caused by depth losses which instead is view-dependent). An interesting observation is that in this case the distribution of $R$ between texture and geometry is not fixed but depends on the selected viewpoint, as explained in Section 3. Figure 21 shows the distribution of $R = 16$ redundancy packets between the various quality layers of depth and texture information: in the case of viewpoint 3 which is quite close (and so it does not require accurate depth data) we have 13 redundancy packets on texture (shown in various tones of blue) and just 3 for depth (green tones), while for the farther viewpoint 1 redundancy is almost equally distributed between the two types of data. To give an idea of the usefulness of this approach, the average MSE on the rendered frames from viewpoint 3 is 2.9 while it would be 4 if the redundancy would have been divided in half between the two elements of the description.

Figure 22 refers to the same set of experiments of the previous plot on a network with a 5% packet loss. Results are similar to the previous case but it can be observed that when transmitting depth information the FEC scheme with 5% of redundancy (4 packets) in this case works better, because the lower error rate permits an easier recovery of the lost packets.

Finally Figure 23 summarizes the previous results for the case of viewpoint 1. It shows the distortion as a function of the packet loss rate for the viewpoint corresponding to camera 1. As expected the distortion increases with the loss rate; however the plot clearly shows how the previously shown results for a 10% loss rate remain valid also in the case of lower loss rates. Table 1 shows also the distribution of the redundancy packets between the various quality layers in texture transmission for the different loss rates while Table 2 shows the same data for depth information. As previously said the first quality layers have a much bigger protection. In analyzing the data in the tables it should be noted that the lower layers have less packets, and so an equal amount of redundancy corresponds to higher protections. It should also be noted how at high loss rates almost all of the redundancy must be allocated to the first layers while in the case of a more reliable network some redundancy is allocated also to less relevant data.

*4.4. Time Analysis of 3D Transmission with the Proposed FEC Protection Scheme.* As shown in the previous section FEC protection schemes allow to obtain better quality. However they also require a longer transmission time due to the

(a)

(b)

(c)

(d)

Figure 14: *Breakdance*: (a) first frame of camera 4, (b) warping of the frame to the viewpoint of camera 3, (c) first frame of camera 4 corrupted by texture packet losses, (d) warping of the corrupted frame to viewpoint 3.

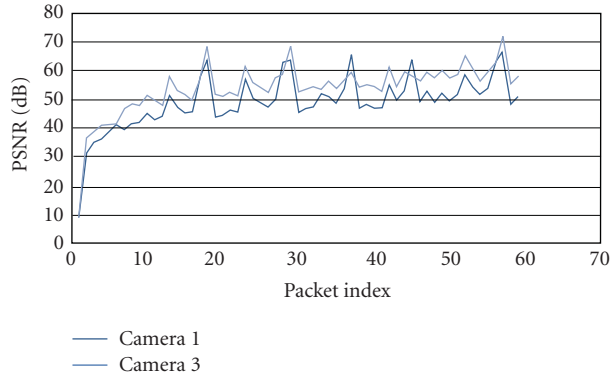



Figure 15: *Breakdance*: loss of a single packet of depth information.

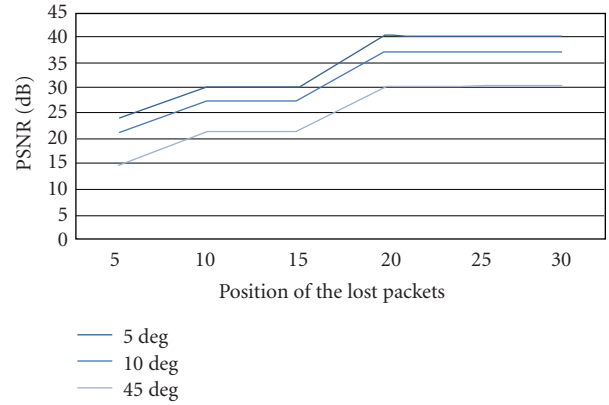



Figure 16: *Goku*: loss of a batch of 5 consecutive packets of depth information.

(a)

(b)

Figure 17: Example of artefacts due to depth packets loss.

(a)

(b)

(c)

(d)

Figure 18: *Breakdance*: (a) depth map from camera 4, (b) depth map from camera 4 corrupted by depth packet losses, (c) warping of the corresponding frame to viewpoint 3 using the corrupted depth, (d) warping of the corresponding frame to viewpoint 1 using the corrupted depth.

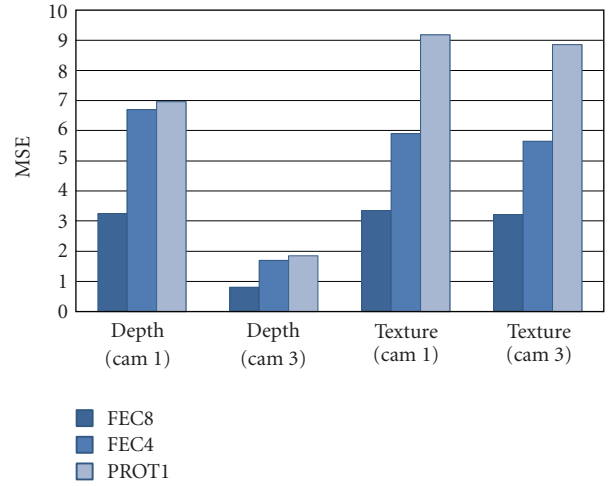



Figure 19: Performance at 10% packet loss rate: distortion due to depth (columns 1 and 2) and texture (columns 3 and 4) packet losses with different error protection schemes.

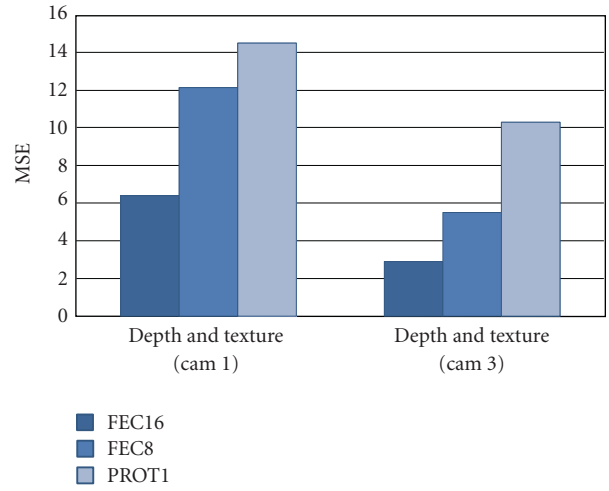



Figure 20: Performance with 10% packet loss rate on both texture and geometry: distortion due to packet losses with different error protection schemes.

overhead caused by redundancy information. In this section we will show an example of the quality versus latency trade off. Figure 24 refers to a network with a 10% packet loss and a FEC scheme with $R = 16$ redundancy packets. It shows the distortion as a function of the amount of transmitted data. Redundancy packets are concentrated on the first (more relevant) layers, so their transmission requires longer times, and at the very beginning the image quality is lower with the proposed FEC scheme (as shown by the zoomed detail of the plot). However it grows up faster and after a short while it gets better. Notice how average quality continues to grow with the proposed scheme, while unrecovered losses on early packets dominate the average quality without the protection scheme and cause the average quality curve to almost saturate.

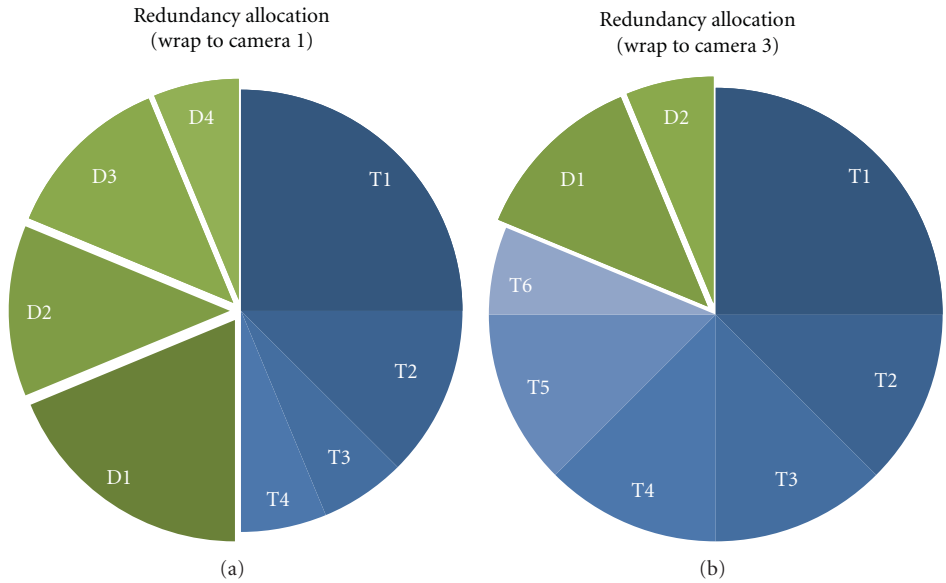



Figure 21: Allocation of the redundancy information between the various quality layers of texture (blue tones) and depth (green tones) information when observing from the viewpoints of Cam1 and Cam3.

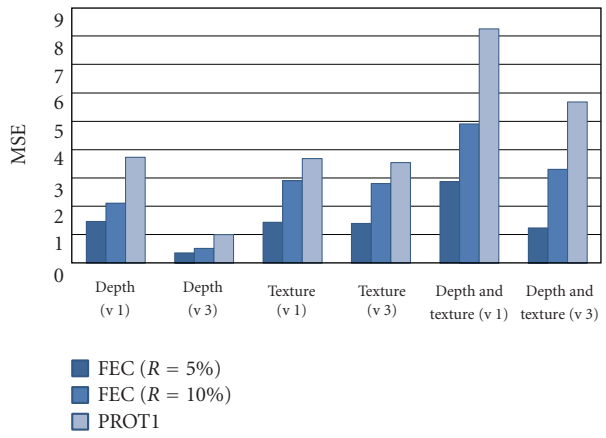



Figure 22: Performance at 5% packet loss rate: distortion due to depth (columns 1 and 2), texture (columns 3 and 4), and joint depth and texture (columns 5 and 6) packet losses with different error protection schemes.

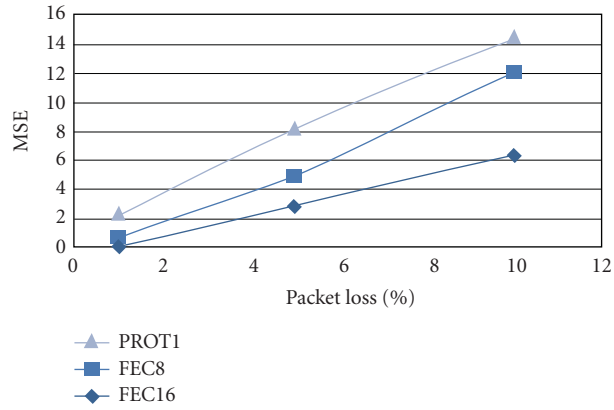



Figure 23: Loss of both depth and texture packets: distortion as a function of the packet loss rate with different error protection schemes (warping to viewpoint v1).

## 5. Conclusions and Future Work

In this paper we propose a novel error correction scheme for the combined transmission of geometry and texture information in depth image-based rendering schemes. The contributions of this paper are several. A first contribution is a novel Forward Error Correction strategy based on Unequal Error Protection that assigns the redundancy to the various elements of depth and texture information on the basis of their relevance and of the scene's geometry and selected viewpoints. The proposed scheme has been tested in the transmission of three-dimensional scenes and experimental results show a considerable improvement in the actual rendering quality over lossy networks. This indicates that the proposed method for assessing the relevance of the different depth and texture elements on the basis of the rendered views' quality is rather effective for practical purposes. A second contribution is a model that theoretically describes the impact of the different texture and geometry elements on the rendering of novel views from arbitrary viewpoints. This model allows to estimate the effects of packet losses in the transmission of compressed depth and texture information in a remote 3D browsing system. Different approximation strategies have been proposed in order to cut a trade off between the accuracy and the computational requirements. The approximate computation can play valuable services in real-time applications, specially whenever distortion may be evaluated with limited accuracy, as when it is used to

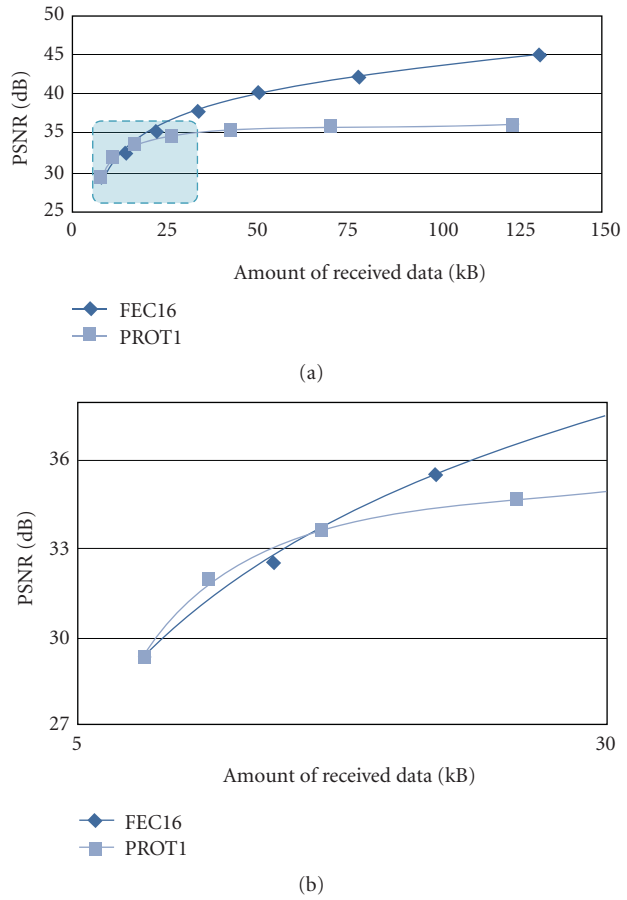



Figure 24: (a) Rendering distortion as a function of the amount of transmitted data (10% packet loss, warping to viewpoint v3); (b) Enlarged image of the highlighted region.

Table 1: Redundancy allocation for texture information ($R = 8$).

| Quality layer | 1 | 2 | 3 | 4 | 5 | 6 | 7 | Tot |
|---|---|---|---|---|---|---|---|---|
| Data packets | 2 | 2 | 3 | 6 | 10 | 19 | 35 | 77 |
| Red. packets (1% loss) | 2 | 1 | 1 | 1 | 1 | 1 | 1 | 8 |
| Red. packets (5% loss) | 3 | 1 | 1 | 1 | 1 | 1 | 0 | 8 |
| Red. packets (10% loss) | 4 | 2 | 1 | 1 | 0 | 0 | 0 | 8 |

balance redundancy between geometry and texture data. Experimental results confirm the theoretical findings and show that while the distortion due to the loss of texture packets is roughly independent of the selected viewpoint, the impact of loss of depth data becomes bigger and bigger while the viewpoints move farther apart from the one of the available images. The current version of the model estimates the MSE of the rendered views and uses this measure as an

Table 2: Redundancy allocation for depth information ($R = 8$).

| Quality layer | 1 | 2 | 3 | 4 | 5 | 6 | 7 | Tot |
|---|---|---|---|---|---|---|---|---|
| Data packets | 1 | 2 | 4 | 6 | 10 | 14 | 25 | 62 |
| Red. packets (1% loss) | 2 | 1 | 1 | 1 | 1 | 1 | 1 | 8 |
| Red. packets (5% loss) | 3 | 2 | 1 | 1 | 1 | 0 | 0 | 8 |
| Red. packets (10% loss) | 3 | 2 | 2 | 1 | 0 | 0 | 0 | 8 |

index of the image quality. Further research will be devoted to the introduction of more accurate and up-to-date metrics into the quality estimation model. Also the critical issue of how to combine image distortion due to depth and texture losses will be the subject of further research, and a more accurate model than the one presented in Section 2.4 will be developed and introduced in the system.

Further research will also focus on the interactivity issue with the target of efficiently applying the proposed scheme in free viewpoint video and 3DTV applications. While the experimental results have been obtained with a JPEG2000/JPIP remote browsing system, the proposed method applies to any compression scheme and its performance with other compression standards will be tested, with a special attention to video compression. The model for the estimation of the impact of depth losses will be improved and extended in order to deal with multiple images and depth maps. This aspect introduces very challenging new issues related to the possibility of replacing lost information from one view or depth map with data coming from other available viewpoints. We finally plan to reinforce the redundancy allocation procedure with a more accurate modeling of the dependency between the various data packets.

## References

[1] A. Smolic, K. Mueller, P. Merkle, et al., "3D video and free viewpoint video—technologies, applications and MPEG standards," in *Proceedings of the IEEE International Conference on Multimedia and Expo (ICME '06)*, pp. 2161–2164, July 2006.

[2] S. Yang, C.-S. Kim, and C.-C. J. Kuo, "A progressive view-dependent technique for interactive 3D mesh transmission," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 14, pp. 1249–1264, 2004.

[3] D. Tian and G. AlRegib, "On-demand transmission of 3D models over lossy networks," *Signal Processing: Image Communication*, vol. 21, no. 5, pp. 396–415, 2006.

[4] N.-H. Lin, T.-H. Huang, and B.-Y. Chen, "3D model streaming based on JPEG 2000," *IEEE Transactions on Consumer Electronics*, vol. 53, no. 1, pp. 182–190, 2007.

[5] P. Zanuttigh, N. Brusco, D. Taubman, and G. Cortelazzo, "A novel framework for the interactive transmission of 3D

scenes," *Signal Processing: Image Communication*, vol. 21, no. 9, pp. 787–811, 2006.

[6] S. Yang, C.-H. Lee, and C.-C. J. Kuo, "Optimized mesh and texture multiplexing for progressive textured model transmission," in *Proceedings of the 12th Annual ACM International Conference on Multimedia (MULTIMEDIA '04)*, pp. 676–683, ACM, New York, NY, USA, 2004.

[7] I. Cheng and A. Basu, "Perceptually optimized 3D transmission over wireless networks," in *Proceedings of the International Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '05)*, Los Angeles, Calif, USA, August 2005.

[8] F. Wu, E. Agu, C. Lindsay, and C. Chen, "Unequal error protection (uep) for wavelet-based wireless 3D mesh transmission," in *Proceedings of the 8th IEEE International Symposium on Network Computing and Applications (NCA '09)*, pp. 242–249, Cambridge, Mass, USA, July 2009.

[9] Z. Chen, J. F. Barnes, and B. Bodenheimer, "Hybrid and forward error correction transmission techniques for unreliable transport of 3D geometry," *Multimedia Systems*, vol. 10, no. 3, pp. 230–244, 2005.

[10] G. AlRegib, Y. Altunbasak, and J. Rossignac, "An unequal error protection method for progressively transmitted 3D models," *IEEE Transactions on Multimedia*, vol. 7, no. 4, pp. 766–776, 2005.

[11] J. Thie and D. Taubman, "Optimal erasure protection strategy for scalably compressed data with tree-structured dependencies," *IEEE Transactions on Image Processing*, vol. 14, no. 12, pp. 2002–2011, 2005.

[12] I. Cheng, L. Ying, and A. Basu, "Packet-loss modeling for perceptually optimized 3D transmission," *Advances in Multimedia*, vol. 2007, no. 1, Article ID 95218, 10 pages, 2007.

[13] A. Bilgin, Z. Wu, and M. W. Marcellin, "Decompression of corrupt jpeg 2000 codestreams," in *Proceedings of the Data Compression Conference (DCC '03)*, Snowbird, Utah, USA, March 2003.

[14] F. Dufaux, G. Baruffa, F. Frescura, and D. Nicholson, "JPWL—an extension of JPEG 2000 for wireless imaging," in *Proceedings of the IEEE International Symposium on Circuits and Systems (ISCAS '06)*, pp. 3870–3873, 2006.

[15] S. Floyd and V. Jacobson, "Random early detection gateways for congestion avoidance," *IEEE/ACM Transactions on Networking*, vol. 1, no. 4, pp. 397–413, 1993.

[16] S. Floyd and K. Fall, "Promoting the use of end-to-end congestion control in the internet," *IEEE/ACM Transactions on Networking*, vol. 7, no. 4, pp. 458–472, 1999.

[17] L. Rizzo, "Effective erasure codes for reliable computer communication protocols," *ACM SIGCOMM Computer Communication Review*, vol. 27, no. 2, pp. 24–36, 1997.

[18] R. Bellman, *Dynamic Programming*, Princeton University Press, Princeton, NJ, USA, 2003.

[19] Y. Pan, I. Cheng, and A. Basu, "Quality metric for approximating subjective evaluation of 3D objects," *IEEE Transactions on Multimedia*, vol. 7, no. 2, pp. 269–279, 2005.

[20] D. S. Taubman, "Localized distortion estimation from already compressed jpeg2000 images," in *Proceedings of the IEEE International Conference of Image Processing (ICIP '6)*, pp. 3089–3092, Atlanta, Ga, USA, October 2006.

[21] R. I. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision*, Cambridge University Press, London, UK, 2nd edition, 2004.

[22] A. Secker and D. Taubman, "Highly scalable video compression with scalable motion coding," *IEEE Transactions on Image Processing*, vol. 13, no. 8, pp. 1029–1041, 2004.

[23] P. Zanuttigh, N. Brusco, D. Taubman, and G. M. Cortelazzo, "Server policies for interactive transmission of 3D scenes," in *Proceedings of the 8th IEEE International Workshop on Multimedia Signal Processing (MMSP '06)*, pp. 59–64, Victoria, Canada, October 2006.

[24] D. Taubman and R. Prandolini, "Architecture, philosophy and performance of JPIP: internet protocol standard for JPEG 2000," in *The International Symposium on Visual Communications and Image Processing (VCIP '03)*, vol. 5150 of *Proceedings of SPIE*, pp. 649–663, Lugano, Switzerland, July 2003.

[25] C. Zitnick, S. Kang, M. Uyttendaele, S. Winder, and R. Szeliski, "Highquality video view interpolation using a layered representation," *ACM Transactions on Graphics*, vol. 23, no. 3, pp. 600–608, 2004.

Hindawi Publishing Corporation
International Journal of Digital Multimedia Broadcasting
Volume 2010, Article ID 247108, 12 pages
doi:10.1155/2010/247108

*Research Article*

# Multicamera Real-Time 3D Modeling for Telepresence and Remote Collaboration

**Benjamin Petit,[1] Jean-Denis Lesage,[2] Clément Menier,[3] Jérémie Allard,[4] Jean-Sébastien Franco,[5] Bruno Raffin,[6] Edmond Boyer,[7] and François Faure[7]**

[1] *INRIA Grenoble, 655 avenue de l'Europe, 38330 Montbonnot Saint Martin, France*
[2] *Université de Grenoble , LIG, 51 avenue Jean Kuntzmann, 38330 Montbonnot Saint Martin, France*
[3] *4D View Solutions, 655 avenue de l'Europe, 38330 Montbonnot Saint Martin, France*
[4] *INRIA Lille-Nord Europe, LIFL, Parc Scientifique de la Haute Borne, 59650 Villeneuve d'Ascq, France*
[5] *Université Bordeaux, LaBRI, INRIA Sud-Ouest, 351 cours de la Libération, 33405 Talence, France*
[6] *INRIA Grenoble, LIG, 51 avenue Jean Kuntzmann, 38330 Montbonnot Saint Martin, France*
[7] *Université de Grenoble, LJK, INRIA Grenoble, 655 avenue de l'Europe, 38330 Montbonnot Saint Martin, France*

Correspondence should be addressed to Benjamin Petit, benjamin.petit@inrialpes.fr

Received 1 May 2009; Accepted 28 August 2009

Academic Editor: Xenophon Zabulis



We present a multicamera real-time 3D modeling system that aims at enabling new immersive and interactive environments. This system, called Grimage, allows to retrieve in real-time a 3D mesh of the observed scene as well as the associated textures. This information enables a strong visual presence of the user into virtual worlds. The 3D shape information is also used to compute collisions and reaction forces with virtual objects, enforcing the mechanical presence of the user in the virtual world. The innovation is a fully integrated system with both immersive and interactive capabilities. It embeds a parallel version of the EPVH modeling algorithm inside a distributed vision pipeline. It also adopts the hierarchical component approach of the FlowVR middleware to enforce software modularity and enable distributed executions. Results show high refresh rates and low latencies obtained by taking advantage of the I/O and computing resources of PC clusters. The applications we have developed demonstrate the quality of the visual and mechanical presence with a single platform and with a dual platform that allows telecollaboration.

## 1. Introduction

Teleimmersion is of central importance for the next generation of live and interactive 3DTV applications. It refers to the ability to embed persons at different locations into a shared virtual environment. In such environments, it is essential to provide users with a credible sense of 3D telepresence and interaction capabilities. Several technologies already offer 3D experiences of real scenes with 3D and sometimes free-viewpoint visualizations, for example, [1–4]. However, live 3D teleimmersion and interaction across remote sites is still a challenging goal. The main reason is found in the difficulty to build and transmit models that carry enough information for such applications. This not only covers visual or transmission aspects but also the fact that such models need to feed 3D physical simulations as required for interaction purposes. In this paper, we address these issues and propose a complete framework allowing the full body presence of distant people into a single collaborative and interactive environment.

The interest of virtual immersive and collaborative environments arises in a large and diverse set of application domains, including interactive 3DTV broadcasting, video gaming, social networking, 3D teleconferencing, collaborative manipulation of CAD models for architectural and industrial processes, remote learning, training, and other collaborative tasks such as civil infrastructure or crisis management. Such environments strongly depend on their ability to build a virtualized representation of the scene of interest, for example, 3D models of users. Most existing systems use 2D representations obtained using mono-camera systems [5–7]. While giving a partially faithful representation of the user, they do not allow for natural interactions, including

consistent visualization with occlusions, which require 3D descriptions. Other systems more suitable for 3D virtual worlds use avatars, as, for instance, massive multiplayer games analog to *Second Life*. However, avatars only carry partial information about users and although real-time motion capture environments can improve such models and allow for animation, avatars do not yet provide sufficiently realistic representations for teleimmersive purposes.

To improve the sense of presence and realism, models with both photometric and geometric information should be considered. They yield more realistic representations that include user appearances, motions and even sometimes facial expressions. To obtain such 3D human models, multicamera systems are often considered. In addition to appearance, through photometric information, they can provide a hierarchy of geometric representations from 2D to 3D, including 2D and depth representations, multiple views, and full 3D geometry. 2D and depth representations are viewpoint dependent and though they enable 3D visualization [8] and, to some extent, free-viewpoint visualization, they are still limited in that respect. Moreover they are not designed for interactions that usually require full shape information instead of partial and discrete representations. Multiple view representations, that is, views from several viewpoints, overcome some of the limitations of 2D and depth representations. In particular, they increase the free-viewpoint capability when used with view interpolation techniques, for example, [3, 9, 10]. However, interpolated view quality rapidly decreases when new viewpoints distant from the original viewpoints are considered. And similarly to 2D and depth representations, only limited interactions can be expected. In contrast, full 3D geometry descriptions allow unconstrained free viewpoints and interactions as they carry more information. They are already used for teleimmersion [2, 4, 11, 12]. Nevertheless, existing 3D human representations, in real-time systems, often have limitations such as imperfect, incomplete, or coarse geometric models, low resolution textures, or slow frame rates. This typically results from the complexity of the method applied for 3D reconstruction, for example, stereovision [13] or visual hull [14] methods, and the number of cameras used.

This article presents a full real-time 3D-modeling system called Grimage (http://grimage.inrialpes.fr/) (Figure 1). It is an extended version of previous conference publications [15–19]. The system relies on the EPVH-modeling algorithm [14, 20] that computes a 3D mesh of the observed scene from segmented silhouettes and robustly yields an accurate shape model [14, 20] at real-time frame rates. Visual presence in the 3D world is ensured by texturing this mesh with the photometric data lying inside the extracted silhouettes. The 3D mesh also enables a mechanical presence, that is, the ability for the user to apply mechanical actions on virtual objects. The 3D mesh is plugged into a physics engine to compute the collisions and the reaction forces to be applied to virtual objects. Another aspect of our contribution is the implementation of the pipeline in a flexible parallel framework. For that purpose, we rely on FlowVR, a middleware dedicated to parallel interactive application [21–23]. The application is structured through a hierarchy of components, the leaves being computation tasks. The component hierarchy offers a high-level of modularity, simplifying the maintenance and upgrade of the system. The actual degree of parallelism and mapping of tasks on the nodes of the target architecture are inferred during a preprocessing phase from simple data like the list of cameras available. The runtime environment transparently takes care of all data transfers between tasks, being on the same node or not. Embedding the EPVH algorithm in a parallel framework enables to reach interactive execution times without sacrificing accuracy. Based on this system we developed several experiments involving one or two modeling platforms.

In the following, we detail the full pipeline, starting with acquisition steps in Section 2, the parallel EPVH algorithm in Section 3, the textured-model rendering and the mechanical interactions in Section 4. A collaborative set up between two 3D-modeling platforms is detailed in Section 6. Section 7 present a few experiments and the associated performance results, before concluding in Section 8.

## 2. A Multicamera Acquisition System

To generate real-time 3D content, we first need to acquire 2D information. For that purpose we have built an acquisition space surrounded by a multicamera vision system. This section will focus on the technical characteristics needed to obtain an image stream from multiple cameras and to transform it into suitable information for the 3D-modeling step, that is, calibrated silhouettes.

*2.1. Image Acquisition.* As described previously, the 3D-modeling method we use is based on images. We thus need to acquire video streams from digital cameras. Today digital cameras are commodity components available from low cost webcams to high-end 3-CCD cameras. Images provided by current webcams proved to be of insufficient quality (low resolution and refresh rates, important optical distortion), which made them unsuitable for our purpose. Consequently we use mid range firewire cameras acquiring up to 30 fps and 2 megapixels color images. Our acquisition platform is equipped with up to 16 cameras. Each camera is connected to a cluster node dedicated to image processing. A software library is used to control our cameras, managing cameras' configurations and frame grabbing under Linux.

Cameras are set to surround the scene. The number of cameras required depends on the size of the scene and the complexity of the objects to model as well as on the quality required for texturing and 3D-modeling. Beyond a certain number of cameras, the accuracy of the model obtained does not significantly improve while the network load increases resulting in higher latencies. Experiments have shown that 8 cameras is generally a good compromise between the model accuracy and the CPU and network load.

The camera locations in the acquisition space usually depends on the application: one can choose to emphasize a side of the set up to have better texture quality in a particular direction, the user's face for example, or to place more cameras in a certain area to get better models of the arms and hands for interaction purposes.

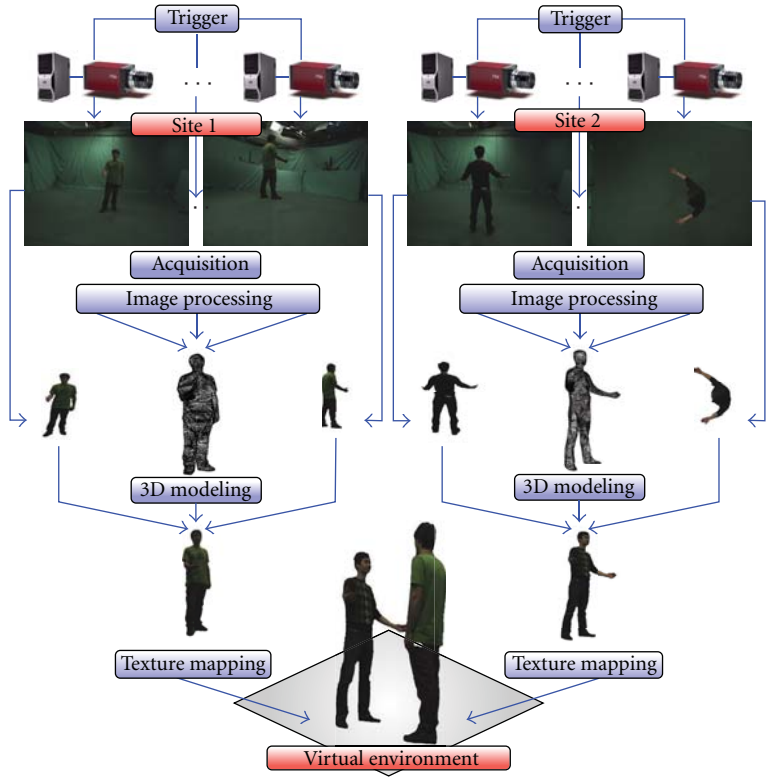



Figure 1: An illustration of the telepresence application pipeline.

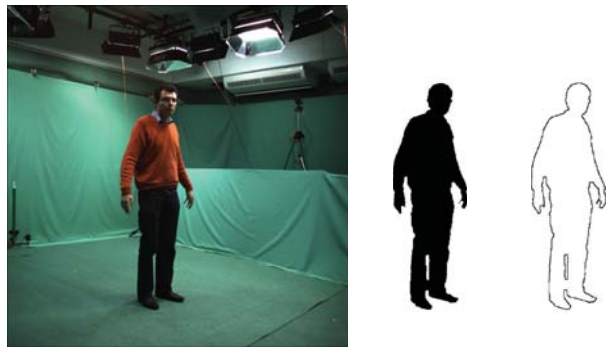
*2.2. Synchronization.* Dealing with multiple input devices raises the problem of data synchronization. In fact all our applications rely on the assumption that the input images captured from the different cameras are coherent, that is, that they relate to the same scene event. Synchronization information could be recovered directly from silhouettes using their inconsistency over several viewpoints as suggested in [24]; however, a hardware solution appears to be more practical and efficient in a dedicated environment such as ours. The image acquisition is triggered by an external signal sent directly through the cameras' genlock connector. This mechanism leads to delays between images below 100 microseconds

*2.3. Calibration.* Another issue when dealing with multiple cameras is to determine their spatial organization in order to perform geometric computations. In practice we need to determine the position and orientation of each camera in the scene as well as its intrinsic characteristics such as the focal length. This is done through a calibration process that computes the function giving the relationship between real 3D points and 2D-image points for each camera.

As with the synchronization step, the silhouette information could also be used to recover calibration information, using for instance [24, 25]. However practical considerations on accuracy favor again a solution which is specific to our dedicated environment. We perform this step using a software we developed which is based on standard off-the-shelf calibration procedures, see for instance [26, 27]. The calibration process consists in sweeping around the scene a wand with four lights with known relative positions on the wand. Once the lights are tracked through time in each image, a bundle adjustment iteratively lowers the reprojection error of the computed 3D light positions into the original images by adjusting the extrinsic and intrinsic parameters of each camera.

*2.4. Background Subtraction.* Regions of interest in the images, that is, the foreground or silhouette, are extracted using a background subtraction process. We assume that the scene is composed of a static background, the appearance of which can be learned in advance. As most of the existing techniques [28, 29], we rely on a per-pixel color model of the background. For our purpose, we use a combination of a Gaussian model for the chromatic information (UV) and an interval model for the intensity information (Y) with a variant of the method by Horprasert et al. [28] for shadow detection (Figure 2(b)). A crucial remark here is that the accuracy of the produced 3D model highly depends on this process since the modeling approach is exact with respect to the silhouettes. Notice that a high-quality background subtraction can easily be achieved by using a dedicated environment (blue screen). However, for prospective purposes, we do not limit our approach to such specific environments in our set up.

*2.5. Silhouette Polygonalization.* Since our modeling algorithm computes a surface and not a volume, it does not use



(a) (b) (c)

Figure 2: The different steps of the image processing: (a) image acquisition, (b) background subtraction (binary image), and (c) exact silhouette polygon (250 vertices).

image regions as defined by silhouettes, but instead their delimiting polygonal contours. We extract such silhouette contours and vectorize them using the method of Debled-Rennesson et al. [30]. Each contour is decomposed into an oriented polygon, which approximates the contour to a given approximation bound. With a single-pixel bound, obtained polygons are strictly equivalent to the silhouettes in the discrete sense (Figure 2(c)). However in case of noisy silhouettes this leads to numerous small segments. A higher approximation bound results in significantly fewer segments. This enables to control the model complexity, and therefore the computation time of the 3D-modeling process, in an efficient way.

## 3. 3D Modeling

To obtain a 3D geometric model of objects and persons located in the acquisition space, we use a shape-from-silhouette method, which builds a shape model called the visual hull. Shape-from-silhouette methods are well adapted to our context for several reasons. First, they yield shape models as required later in the process, for example, texture mapping or interaction. Second, they provide such models in real-time. Even though more precise approaches exist, for example, [31–33], most will fail at providing 3D models in real-time over long period of time and in a robust and efficient way as shape-from-silhouette approaches do. Below, we precise our shape-from-silhouette method.

*3.1. Visual Hull.* The visual hull is a well-studied geometric shape [34, 35] which is obtained from scene object's silhouettes observed in $n$ views. Geometrically, the visual hull is the intersection of the *viewing cones*, the generalized cones whose apices are the cameras' projective centers and whose cross-sections coincide with the scene silhouettes (Figure 3). When considering piecewise-linear image contours for silhouettes, the visual hull becomes a regular polyhedron. A visual hull cannot model concavities but can be efficiently computed and yield a very good human shape approximation.

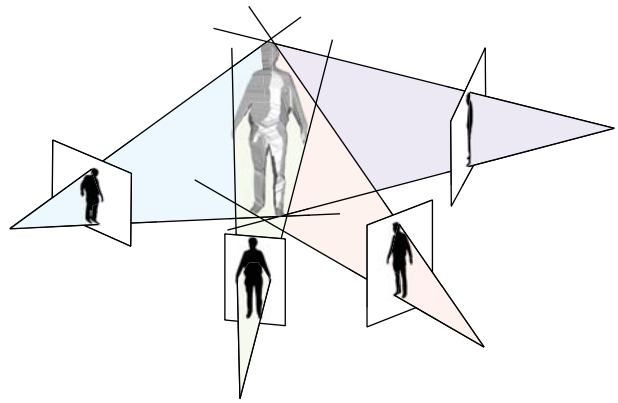


Figure 3: Visual hull of a person with 4 views.

Our work is based on the exact polyhedron visual hull (EPVH) algorithm [14, 20]. The EPVH algorithm has the particularity of retrieving an exact 3D model, whose projection back into the images coincides with the observed silhouettes. This is an important feature when the models need to be textured as it makes textures, extracted from silhouettes, directly mappable on the 3D model.

The method we present here recovers the visual hull of a scene object in the form of a polyhedron. As previously explained, silhouette contours of the scene object are retrieved for each view as a 2D polygon. Such a discrete polygonal description of silhouettes induces a unique polyhedron representation of the visual hull, the structure of which is recovered by EPVH. To achieve this, three steps are performed. First, a particular subset of the polyhedron edges is computed: the viewing edges, which we describe below. Second, all other edges of the polyhedron mesh are recovered by a recursive series of geometric deductions. The positions of vertices not yet computed are gradually inferred from those already obtained, using the viewing edges as an initial set. Third, the mesh is consistently traversed to identify the faces of the polyhedron.

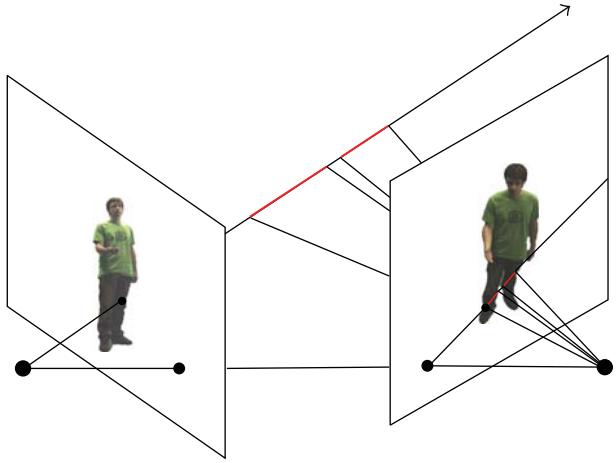


Figure 4: Viewing edges (in bold) along the viewing line.

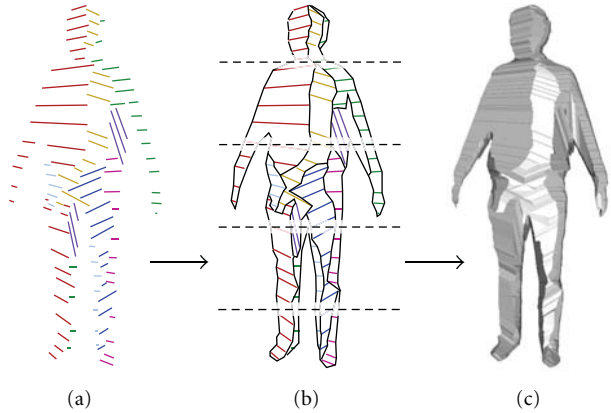



Figure 5: The three main steps of the EPVH algorithm, (a) viewing edge computation, (b) mesh connectivity (horizontal slices depict the partitioning used for the parallel version of the algorithm), and (c) face generation.

We first give an overview of the sequential algorithm. For more details refer to [14]. Then we explain how we distribute this algorithm in order to reach real-time performance.

*3.1.1. Computing the Viewing Edges.* *Viewing edges* are the edges of the visual hull induced by viewing lines of contour vertices, see Figure 4. There is one viewing line per silhouette 2D vertex. On each viewing line, EPVH identifies segments that project inside silhouettes in all other images. Each segment, called a viewing edge, is an edge of the visual hull and each segment extremity a 3D vertex. Each 3D vertex is trivalent, that is, the intersection point of 3 edges. Higher valence is neglected because it is highly unlikely in practice.

*3.1.2. Computing the Visual Hull Mesh.* After the first step, the visual hull is not yet complete. Some edges are missing to fulfill the mesh connectivity. Some vertices, called triple points, are also missing. A triple point is a vertex of the visual hull generated from the intersection of three planes defined by silhouette segments from three different images. EPVH completes the visual mesh by traveling along 3D edges as defined by two silhouette edges as long as these 3D edges project inside all silhouettes. At the limit, that is, when it projects on a silhouette contour, it identifies new triple points or recognize already computed visual hull vertices.

A last step consists in traversing the mesh to identify the polyhedron faces. 3D face contours are extracted by walking through the complete oriented mesh while always taking left turns at each vertex. Orientation data is inferred from silhouette orientations (counterclockwise oriented outer contours and clockwise oriented inner contours).

### 3.2. Distributed Algorithm

*3.2.1. Algorithm.* For real-time execution we developed a parallel version of the EPVH algorithm using a three-stage pipeline:

*Stage 1: Viewing Edges.* Let $V$ be the number of thread—each thread being distributed on different CPUs across the cluster's hosts—in charge of computing the viewing edges. The silhouettes extracted by all image processing hosts are broadcasted to the $V$ threads. Each thread computes locally the viewing edges for $n/V$ viewing lines, where $n$ is the total number of viewing lines (Figure 5(a)).

*Stage 2: Mesh Connection.* Let $M$ be the number of thread in charge of computing the mesh. The $V$ threads from the previous step broadcast the viewing edges to the $M$ threads. Each thread is assigned a slice of the space (along the vertical axis as we are usually working with standing humans) where it computes the mesh. Slices are defined to have the same number of vertices. Each thread completes the connectivity of its submesh, creating triple points when required. The submeshes are then gathered on one host that merges the results, taking care of the connectivity on slice boundaries removing duplicate edges or adding missing triple points (Figure 5(b)).

*Stage 3: Face Identification.* The mesh is broadcasted to $K$ threads in charge of face identification. Workload is balanced by evenly distributing the set of generator planes among processors (Figure 5(c)).

*3.2.2. Evaluation.* Consider an acquisition space surrounded by up to 8 cameras and with 1 or 2 persons. In that case, the algorithm reaches the cameras' refresh rates (tuned in between 20 and 30 frames per second) and ensures a latency below 100 milliseconds (including video acquisition and 2D-image processing) with between 4 and 8 processors. While the algorithm is flexible and allows for more processors, it did not prove to significantly increase the performance in this experimental context. More information and results about the parallelization of this algorithm can be found in [15].

Using a large number of cameras raises several issues. The algorithm complexity is quadratic in the number of

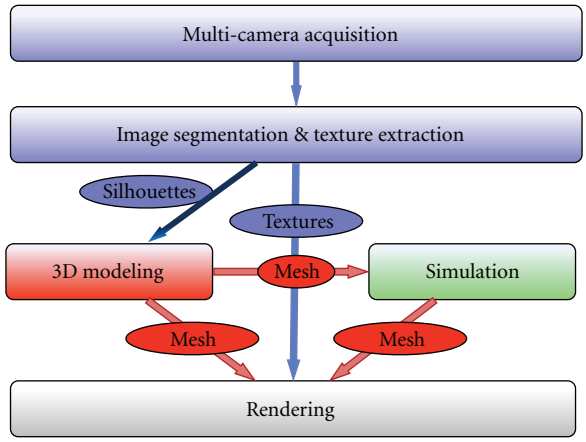



Figure 6: Application architecture coupling a multicamera acquisition space and a virtual physical environment.

cameras, quickly leading to non acceptable latencies. Today our efforts focus on using higher resolution cameras rather than significantly more cameras. The algorithm complexity is in $n\log(n)$, where $n$ is the maximum number of segments per silhouette, making it more scalable on this parameter. Having more cameras makes sense for large acquisition spaces, where 3D models are computed per subsets of cameras.

*3.3. Texture Extraction.* We also extract from each silhouette the photometric data that will be used later during the rendering process for photorealistic rendering.

## 4. Interaction and Visualization

The 3D mesh, and the associated textures, acquired by the multicamera system are sent over the network to the visualization node and the physical simulation node that handle interactions (Figure 6). These tasks are detailed below.

### *4.1. Simulation*

*4.1.1. Collision Based Interactions.* Coupling real-time 3D-modeling with a physical simulation enables interaction possibilities that are not symbolic and feel therefore natural to the user. Using the SOFA (http://www.sofa-framework.org/) framework, we developed a distributed simulator that handles collisions between soft or rigid virtual objects and the user's body. Unlike with most traditional interactive applications, it allows to use any part of the body or any accessories seen inside the acquisition space without being invasive. Some interactions are intricate, the prehension of objects, for example, is very difficult as there is no force information linked to the model.

*4.1.2. SOFA.* SOFA (simulation open framework application) is an open source framework primarily targeted at medical simulation research [36]. Its architecture relies on several innovative concepts, in particular the notion of multimodel representation. In SOFA, most simulation components, for instance, deformable models, collision models or instruments, can have several representations, connected together through a mechanism called mapping. Each representation is optimized for a particular task such as mechanical computations, collision detection or visualization.

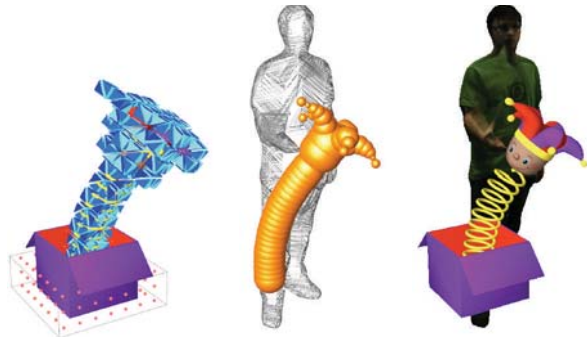
Integrating a SOFA simulation in our applications required adding a new component, receiving the stream of 3D meshes modeling the user and packaging it as an additional collision model (Figure 7). The triangulated polyhedron as computed by the 3D-modeling step can directly be used for collision detection. It is seen from the physics simulation point of view as a rigid mesh insensible to external forces (infinite mass), similar to predefined obstacles such as the floor, with the difference that it is changed each time a new mesh is received.

In order to obtain accurate interactions, the collision response additionally requires the speed and direction of motion at collision points, so that the user can push virtual objects, for example, kicking a ball, instead of only blocking them. We currently provide this information by querying the minimum distance of the current surface point to the previous modeled mesh. This is efficiently implemented by reusing the proximity-based collision detection components in SOFA. The computed distance is an estimation of the user's motion perpendicular to the surface, which is enough to give the colliding objects the right impulsion. However tangential frictions cannot be captured. The different parameters, like mass, spring stiffness, of the simulated scene are empirically tuned based on a trade-off between real-time constraints and a visibly plausible behavior.

Another key aspect of SOFA is the use of a scene-graph to organize and process the components while clearly separating the computation tasks for their possibly parallel scheduling. This data structure, inspired by classical rendering scene-graphs like OpenSG, is new in physically-based animation. Physical actions such as force accumulation or state vector operations are implemented as traversal actions. This creates a powerful framework for differential equation solvers suitable for single objects as well as complex systems made of different kinds of interacting physical bodies: rigid bodies, deformable solids or fluids.

We use an iterative implicit time integration solver. The maximum number of iterations is tuned to limit the computation time. This creates a trade-off between accuracy and computation time that allows us to reach the real-time constraint without sacrificing stability. Parallel versions of SOFA on multicore processors [37] and on GPU have been developed, allowing to interactively simulate rich environments.

*4.2. Rendering.* Data to be rendered, either provided by the simulation software, a static scene loader, or from the 3D-modeling algorithm, are distributed to dedicated rendering nodes. The rendering can be performed on heterogeneous display devices such as standard monitors, multiprojector walls, head-mounted displays or stereoscopic displays.



(a) (b) (c)

Figure 7: An interactive deformable object, (a) collides with the 3D-reconstructed mesh, (b) allowing interactions with the user in (c).

*4.2.1. FlowVR Render.* To distribute efficiently the rendering part, we use FlowVR Render [38]. Existing parallel or remote rendering solutions rely on communicating pixels, OpenGL commands, scene-graph changes or application-specific data. We rely on an intermediate solution based on a set of independent graphics primitives that use hardware shaders to specify their visual appearance. Compared to an OpenGL based approach, it reduces the complexity of the model by eliminating most fixed function parameters while giving access to the latest functionalities of graphics cards. It also suppresses the OpenGL state machine that creates data dependencies making primitive reordering and multistream combining difficult.

Using a retained-mode communication protocol transmitting changes between each frame, combined with the possibility to use shaders to implement interactive data processing operations instead of sending final colors and geometry, we are able to optimize the network load. High-level information such as bounding volumes is used to set up advanced schemes where primitives are issued in parallel, routed according to their visibility, merged and reordered when received for rendering. Different optimization algorithms can be efficiently implemented, saving network bandwidth or reducing texture switches for instance.

*4.2.2. 3D Model and Texture Mapping.* Rendering the 3D model is quite simple as it is already a polygonal surface. To apply the textures extracted from the silhouettes, we use a shader that projects the mesh vertices in source images consistently with camera calibration parameters. The exact polyhedral visual hull algorithm guarantees that the 3D model can be projected back to the original silhouette with minimal error, a property that leads to a better quality texture mapping. Taking into account the surface normal, viewing direction, as well as self-occlusions, the pixel shader smoothly combines the contributions from the different cameras. Having access to the full 3D surface enables interactive and unconstrained selection of rendering viewpoints, and yields realistic views of the reconstructed person.

## 5. Platform Integration

Coupling the different software components involved into this project and distributing them on the nodes of a PC cluster for reaching real-time executions is performed through the FlowVR (http://flowvr.sourceforge.net/) middleware [21, 23], a middleware we developed conjointly with the Grimage project.

FlowVR enforces a modular programming that leverages software engineering issues while enabling high performance executions on distributed and parallel architectures. FlowVR relies on a dataflow and component-oriented programming approach that has been successfully used for other scientific visualization tools. Developing a FlowVR application is a two-step process. First, modules are developed. Modules are endless loops consuming data on input ports at each iteration and producing new data on output ports. They encapsulate a piece of code, imported from an existing application or developed from scratch. The code can be multithreaded or parallel, as FlowVR supports parallel code coupling. In a second step, modules are mapped on the target architecture and assembled into a network to define how data are exchanged. This network can make use of advanced features, from bounding-box-based routing operations to complex message filtering or synchronization operations.

The FlowVR runtime engine runs a daemon on each node of the cluster. This daemon is in charge of synchronization and data exchange between modules. It hides all networking aspects to modules, making module development easier. Each daemon manages a shared memory segment. Messages handled by modules are directly written and read from this memory segment. If data exchange is local to a node, it only consists in a pointer exchange, while the daemon takes care of transferring data through the network for internode communications.

The largest FlowVR applications are composed of thousand of modules and connections. To be able to deal with the network design of such applications, FlowVR is based on hierarchical component model [22]. This model introduces a new kind of components called composite. A composite is designed by assembling other FlowVR components. This

hierarchy enables to create a set of efficient and reusable patterns or skeletons, for example, one-to-many broadcast or scatter collective communications are encapsulated into communication tree patterns made generic and parametric to be used in various contexts. A compilation step instantiates skeletons parameters to fit to the target architecture, for example, in case of communication trees, parameters to be set are the tree arity and the mapping of each node. Using description files or parameters, the compilation step unfolds the hierarchical description and produces a flat FlowVR network optimized for the target architecture.

The Grimage network (Figure 8) is a thousand modules and connections application developed by assembling this set of skeletons. The network relies on several communications patterns, for example, in the acquisition component, a pipeline is associated to each camera. The 3D reconstruction algorithm needs a strong coherency between these pipelines. To reach real-time execution, application needs sometimes to discard a metaframe because it will not be able to compute it under real-time constraints. Therefore a pattern is in charge to do this sampling and keep the coherency. This pattern synchronizes all pipelines and discards the metaframe in the distributed context. The FlowVR compilation process enforces the modularity of the application. The hierarchical description of Grimage is totally independent from the acquisition set up and the target architecture. A file describing the architecture and the acquisition set up is used to compile the network. The compilation process will create the appropriate number of pipelines or reconstruction parallel processes based on the architecture description file. Therefore, in case of set up modification (add a stereoscopic display, integration of a new SMP node in the PC cluster or modification of the number of cameras), the only change needed is to update the architecture description file. This modularity is critical to the Grimage application that has been developed over several years by various persons. FlowVR is a key component that made it possible to aggregate and efficiently execute on a PC cluster the various pieces of code involved. The level of modularity achieved significantly eases the maintenance and enhancements of the application.

## 6. Collaborative Environment

Virtual environments, such as multiplayer games or social network worlds, need a representation of each user that is often an avatar controlled with a keyboard and a mouse. In contrast, our system virtualizes the user into a model that has the user's geometry and appearance at any instant, hence relaxing the need for control devices and enabling new types of interactions with virtual worlds as discussed below.

*6.1. Visual Presence.* From the user's point of view, the sense of presence is drastically improved with an avatar that has the user's shape and aspect instead of those of a purely synthetic avatar taken from a 3D model database. In addition, the avatar is moving with respect to the user' body gestures and not to a preprogrammed set of actions. Different users can therefore recognize themselves and have life-like conversations. Also emotions can be communicated through face expressions and body gestures.

*6.2. Mechanical Presence.* Sharing our appearance is not the only advantage of our environment. 3D meshes can also be used to interact with shared virtual objects. The server managing the virtual environment receives user information (geometric 3D models, semantic actions, etc.), runs the simulation and sends back the transformation of the virtual scene to the users (Figure 9). The dynamic deformable objects are handled by this server while heavy static scenes can be loaded at initialization on each user's rendering node. Such an environment can be used by multiple users to interact together from different locations with the same virtual objects. For each iterative update the physical simulation detects collisions and computes the effect of each user interaction on the virtual world. It is of course impossible to change the state of the input models themselves as there are no force-feedback devices on our platforms. Physically simulated interactions between participants are also impossible for the same reason.

*6.3. Remote Presence.* Remote site visualization of models requires the transfer of 3D model streams and their associated textures under the constraints of limited bandwidth and minimal latency. The mesh itself is not bandwidth intensive and can be easily broadcasted over the network. The textures, one per camera in our current implementation, induce much larger transfers and represent the bulk of the data load. We will provide data bandwidth measurements in Section 7 for a particular set up. We do not consider any specific transfer protocol, which is beyond the scope of this work.

The FlowVR middleware handles the synchronization of both texture and mesh streams to deliver consistent geometric and photometric data, that is, the texture stream gathered from the acquisition nodes must be rendered at the same time as the 3D model reconstructed with this same image stream, otherwise it would lead to visual artifacts. It also prevents network congestion by resampling the streams (discarding some 3D metaframe) in order to send only up-to-date data to the end-user nodes. As the physical simulation only needs the mesh, each site sends it only the meshes as soon as available. We did not experience incoherency issues requiring to enforce a strong time synchronization between meshes.

## 7. Experiments

We report below on preliminary experiments that were conducted with two platforms located in the same room.

*7.1. Practical Set up.* The first acquisition platform is built with 8 firewire cameras with 1 MP resolution, allowing an acquisition space of 2 by 2 meters, suitable for a full person. The PC cluster used is composed of 10 dual xeon PCs connected through a gigabit Ethernet network.

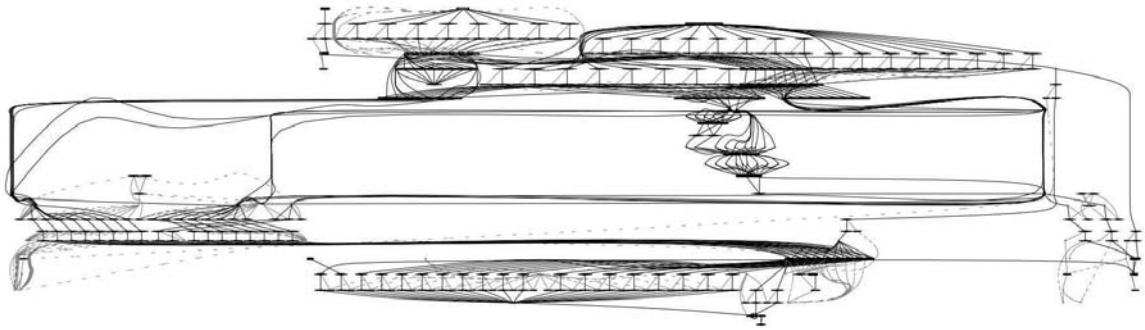


Figure 8: The FlowVR flat network of the Grimage application. Nodes are modules and edges communication channels. This network is compiled for 8 cameras and EPVH parallelized on 6 CPUs.

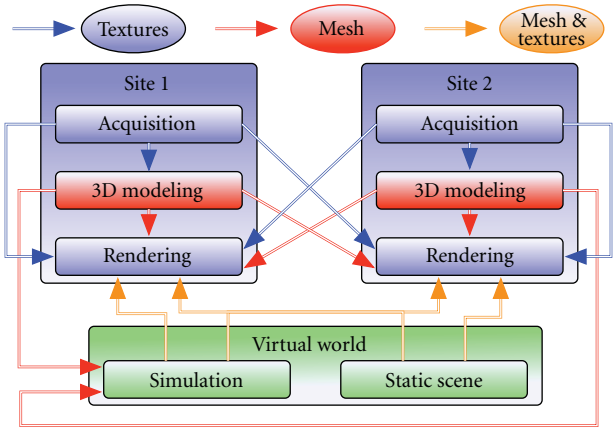



Figure 9: Application architecture for two multicamera acquisition spaces and a virtual physical environment.

The second acquisition platform is a portable version of the first one with an acquisition space of 1 square meter at table height used for demonstration purpose. It uses 6 firewire cameras and is suitable for hand/arm interactions. The cluster is built with 6 mini-PCs used for camera acquisition, 1 dual xeon server for computation, and a laptop for supervision. This platform was presented at Siggraph in 2007 [36].

The two platforms are connected by a gigabit Ethernet network using one PC as gateway between the two platforms. This PC gathers the data from the two platforms and handles the physical simulation.

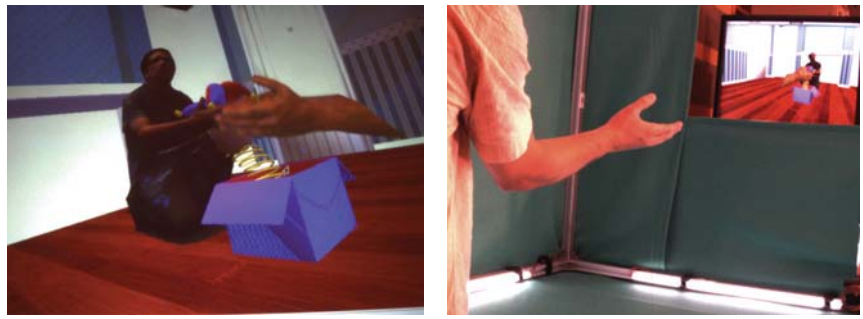
*7.2. Data Estimation.* Our 8-camera platform produces 1 MP images, yielding 3 MB images and thus a theoretical 24 MB multiimage frame throughput. In practice the only image data needed for texturing lies inside the silhouettes, which we use to reduce transfer sizes. When one user is inside the acquisition space the silhouettes occupy usually less than 20% of the overall image in a full-body set up. Thus an average multitexture frame takes 4.8 MB. We also need to send the silhouette mask to decode the texture. A multisilhouette mask frame takes about 1 MB. The overall estimated stream is about 5.8 MB. To decrease the needed bandwidth we decided to use the full resolution of the camera for 3D model computation but only half the resolution for texture mapping, reducing the full multitexture frame to a maximum of 1.45 MB to transfer at each iteration. The mesh itself represents less than 80 KB (about 10000 triangles).

Running at 20 frames per second, which is reasonable for good interactions, the dual platform requires a 29 MB/second bandwidth for 3D frame streaming which is easily scalable to a Gigabit Ethernet network (120 MB/s).

*7.3. Results.* We are able to acquire images and to generate the 3D meshes at 20 fps on each platform. The simulation and the rendering processes are running respectively at 50–60 fps and 50–100 fps, depending of the load of the system. As they run asynchronously from the 3D model and texture generation we need to resample the mesh and the texture streams independently. In practice the mesh and texture transfer between sites oscillates between 15 fps and 20 fps, depending on the size of the silhouette inside the images. Meanwhile the transfer between the 3D-modeling and the rendering node inside a platform and the transfer going to the simulation node are always running at 20 fps. We do not experience any extra connection latency between the two platforms. During execution, the application does not overload the gigabit link.

The accuracy of the model obtained using EPVH is satisfactory both for visual experience and for physical simulation precision. The level of detail of the model is good enough to distinguish the user's fingers. Our application is robust to input noise, the obtained 3D model is watertight (no holes) and manifold (no self intersection). It is also robust to network load change as the data transmitted could be resampled to avoid latency.

We did not conduct any user study about the sense of presence achieved through this system (Figure 10). However the numerous users that experienced the system, in particular during the Emerging Technology show at Siggraph 2007 [36], were generally impressed by the quality of the visual and mechanical presence achieved without requiring handheld devices, markers or per-user calibration steps. Interaction was intuitive, often requiring no explanation, as it was direct,



(a) (b)

Figure 10: (a) the 3D virtual environment with a "full-body" user and a "hand" user, interacting together with a virtual puppet, and (b) the "hand-size" acquisition platform.

full-body and relying on physical paradigms that somehow mimicked a common real world experience. This positive feedback was achieved despite the non immersive display used (a 2D display located 75 cm in front of the user) and the third-person visualization. Future work will focus on associating Grimage and an immersive visualization environment such as a HMD to enable first-person visualization and allow for better depth perception.

A similar experiment was showcased at VRST 2008 in Bordeaux [39], involving two "hand-size" platforms in the same room. Visitors could see each other's hands immersed in the same virtual room. A virtual puppet was animated by the physics simulation. Each user could push or grab the puppet. They could also try to collaborate for instance to grab the puppet using two hands, one from each user. No direct user-to-user physical interaction was possible. The meshes would simply intersect each other when both hand positions superpose in the virtual world. The videos (http://grimage.inrialpes.fr/telepresence/) of our experiments give a good overview of the sense of presence achieved.

## 8. Conclusion

We presented in this article the full Grimage 3D-modeling system. It adopts a software component-oriented approach offering a high-level of flexibility to upgrade part of the application or reuse existing components in different contexts. The execution environment supports the distribution of these components on the different nodes of a PC cluster or Grid. We can thus harness distributed I/O and computing resources to reach interactive execution times but also to build multiplatform applications. 3D-modeling relies on the EPVH algorithm that computes from 2D-images a 3D mesh corresponding to the visual hull of the observed scene. We also retrieve photometric data further used for texturing the 3D mesh. Experiments show that Grimage is suitable for enforcing the visual and mechanical presence of the modeled users.

The actual state of development shows some limitations. For instance we do not extract a complete velocity field on the mesh surface, our algorithm only provide an estimation of the normal velocity and does not provide any tangential velocity. This lack of data limits the range of possible mechanical interactions. As a consequence, the user can modulate the force it applies to a given virtual object but has difficulties to keep an object on his hand or to grab anything. The first steps of the vision pipeline are crucial for the accuracy of the final 3D model. We experienced that a higher quality background subtraction could significantly improve the 3D mesh. We are working on advanced background subtraction algorithms using fine-grain parallel processing to keep the computation time low.

This paper includes a preliminary experiment with a dual platform. We are today conducting telepresence and collaboration experiments between distant sites, each one having its own multicamera environment. This context will require further optimizations to control the amount of data exchanged between sites to keep an acceptable latency.

## Acknowledgment

This work was partly funded by Agence Nationale de la Recherche, contract ANR-06-MDCA-003.

## References

[1] P. J. Narayanan, P. W. Rander, and T. Kanade, "Constructing virtual wolrds using dense stereo," in *Proceedings of the 6th International Conference on Computer Vision*, pp. 3–10, Bombay, India, 1998.

[2] M. Gross, S. Würmlin, M. Naef, et al., "Blue-c: a spatially immersive display and 3D video portal for telepresence," *ACM Transactions on Graphics*, vol. 22, no. 3, pp. 819–827, 2003.

[3] W. Matusik and H. Pfister, "3D TV: a scalable system for real-time acquisition, transmission, and autostereoscopic display of dynamic scenes," *ACM Transactions on Graphics*, vol. 23, no. 3, pp. 814–824, 2004.

[4] G. Kurillo, R. Bajcsy, K. Nahrsted, and O. Kreylos, "Immersive 3D environment for remote collaboration and training of physical activities," in *Proceedings of IEEE Virtual Reality Conference*, pp. 269–270, 2008.

[5] L. Gharai, C. S. Perkins, R. Riley, and A. Mankin, "Large scale video conferencing: a digital amphitheater," in *Proceedings of*

*the 8th International Conference on Distributed Multimedia Systems*, San Francisco, Calif, USA, September 2002.

[6] H. Baker, D. Tanguay, I. Sobel, et al., "The coliseum immersive teleconferencing system," in *Proceedings of the International Workshop on Immersive Telepresence*, Juan Les Pins, France, December 2002.

[7] D. Nguyen and J. Canny, "MultiView: spatially faithful group video conferencing," in *Proceedings of of the SIGCHI Conference on Human Factors in Computing Systems*, pp. 799–808, 2005.

[8] Philips 3D Solutions, "WoWvx technology".

[9] E. Chen and L. Williams, "View interpolation for image synthesis," in *Proceedings of the 20th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '93)*, pp. 279–288, Anaheim, Calif, USA, 1993.

[10] C. L. Zitnick, S. B. Kang, M. Uyttendaele, S. Winder, and R. Szeliski, "High-quality video view interpolation using a layered representation," in *Proceedings of International Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '04)*, pp. 600–608, 2004.

[11] J. Mulligan and K. Daniilidis, "Real time trinocular stereo for tele-immersion," in *Proceedings of IEEE International Conference on Image Processing*, vol. 3, pp. 959–962, 2001.

[12] P. Kauff and O. Schreer, "An immersive 3D videoconferencing system using shared virtual team user environments," in *Proceedings of International Conference on Collaborative Virtual Environments*, pp. 105–112, 2002.

[13] D. Scharstein and R. Szeliski, "A taxonomy and evaluation of dense two-frame stereo correspondence algorithms," *International Journal of Computer Vision*, vol. 47, no. 1–3, pp. 7–42, 2002.

[14] J.-S. Franco and E. Boyer, "Efficient polyhedral modeling from silhouettes," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 31, no. 3, pp. 414–427, 2009.

[15] J.-S. Franco, C. Ménier, E. Boyer, and B. Raffin, "A distributed approach for real time 3D modeling," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshop (CVPRW '04)*, p. 31, June 2004.

[16] J. Allard, E. Boyer, J.-S. Franco, C. Ménier, and B. Raffin, "Marker-less real time 3D modeling for virtual reality," in *Immersive Projection Technology*, 2004.

[17] J. Allard, J.-S. Franco, C. Ménier, E. Boyer, and B. Raffin, "The GrImage platform: a mixed reality environment for interactions," in *Proceedings of the 4th IEEE International Conference on Computer Vision Systems (ICVS '06)*, pp. 46–52, 2006.

[18] J. Allard, C. Ménier, B. Raffin, E. Boyer, and F. Faure, "GrImage: markerless 3D interactions," in *Proceedings of the International Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '07)*, San Diego, Calif, USA, 2007.

[19] J. Allard, C. Ménier, E. Boyer, and B. Raffin, "Running large VR applications on a PC cluster: the FlowVR experience," in *Immersive Projection Technology*, 2005.

[20] J.-S. Franco and E. Boyer, "Exact polyhedral visual hulls," in *Proceedings of the British Machine Vision Conference*, vol. 1, pp. 329–338, 2003.

[21] J. Allard, V. Gouranton, L. Lecointre, et al., "FlowVR: a middleware for large scale virtual reality applications," in *Euro-Par Parallel Processing*, vol. 3149 of *Lecture Notes in Computer Science*, pp. 497–505, Springer, Berlin, Germany, 2004.

[22] J.-D. Lesage and B. Raffin, "A hierarchical component model for large parallel interactive applications," *The Journal of Supercomputing*, vol. 7, no. 1, pp. 67–80, 2008.

[23] J.-D. Lesage and B. Raffin, "High performance interactive computing with FlowVR," in *Proceedings of IEEE Virtual Reality SEARIS Workshop*, pp. 13–16, 2008.

[24] S. N. Sinha and M. Pollefeys, "Synchronization and calibration of camera networks from silhouettes," in *Proceedings of the 17th International Conference on Pattern Recognition (ICPR '04)*, vol. 1, pp. 116–119, Cambridge, UK, August 2004.

[25] E. Boyer, "On using silhouettes for camera calibration," in *Proceedings of the 7th Asian Conference on Computer Vision (ACCV '06)*, vol. 3851 of *Lecture Notes in Computer Science*, pp. 1–10, Hyderabad, India, January 2006.

[26] A. Hilton and J. Mitchelson, "Wand-based multiple camera studio calibration," Tech. Rep. VSSP-TR-2, CVSSP, 2003.

[27] Z. Zhang, "Camera calibration with one-dimensional objects," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 26, no. 7, pp. 892–899, 2004.

[28] T. Horprasert, D. Harwood, and L. S. Davis, "A statistical approach for real-time robust background subtraction and shadow detection," in *Proceedings of the 7th IEEE International Conference on Computer Vision (ICCV '99)*, vol. 99, pp. 1–19, Kerkyra, Greece, September 1999.

[29] G. K. M. Cheung, T. Kanade, J.-Y. Bouguet, and M. Holler, "Real time system for robust 3D voxel reconstruction of human motions," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 2, pp. 714–720, 2000.

[30] I. Debled-Rennesson, S. Tabbone, and L. Wendling, "Fast polygonal approximation of digital curves," in *Proceedings of the International Conference on Pattern Recognition*, vol. 1, pp. 465–468, 2004.

[31] S. M. Seitz, B. Curless, J. Diebel, D. Scharstein, and R. Szeliski, "A comparison and evaluation of multi-view stereo reconstruction algorithms," in *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR '06)*, vol. 1, pp. 519–526, New York, NY, USA, June 2006.

[32] D. Vlasic, I. Baran, W. Matusik, and J. Popović, "Articulated mesh animation from multi-view silhouettes," *ACM Transactions on Graphics*, vol. 27, no. 3, article 97, 2008.

[33] J. Gall, C. Stoll, E. de Aguiar, C. Theobalt, B. Rosenhahn, and H.-P. Seidel, "Motion capture using joint skeleton tracking and surface estimation," in *Proceedings of International Conference on Computer Vision and Pattern Recognition (CVPR '09)*, Miami, Fla, USA, June 2009.

[34] A. Laurentini, "The visual hull concept for silhouette-based image understanding," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 16, no. 2, pp. 150–162, 1994.

[35] S. Lazebnik, E. Boyer, and J. Ponce, "On how to compute exact visual hulls of object bounded by smooth surfaces," in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, vol. 1, pp. 156–161, 2001.

[36] J. Allard, S. Cotin, F. Faure, et al., "SOFA—an open source framework for medical simulation," in *Medicine Meets Virtual Reality*, pp. 1–6, 2007.

[37] E. Hermann, B. Raffin, and F. Faure, "Interactive physical simulation on multicore architectures," in *Proceedings of Symposium on Parallel Graphics and Visualization (EGPGV '09)*, pp. 1–8, Munich, Germany, March 2009.

[38] J. Allard and B. Raffin, "A shader-based parallel rendering framework," in *Proceedings of the IEEE Visualization Conference*, pp. 127–134, 2005.

[39] B. Petit, J.-D. Lesage, J.-S. Franco, E. Boyer, and B. Raffin, "Grimage: 3D modeling for remote collaboration and telepresence," in *Proceedings of the ACM Symposium on Virtual Reality Software and Technology (VRST '08)*, pp. 299–300, 2008.

Hindawi Publishing Corporation
International Journal of Digital Multimedia Broadcasting
Volume 2010, Article ID 419493, 11 pages
doi:10.1155/2010/419493

*Research Article*

# Stereo 3D Mouse Cursor: A Method for Interaction with 3D Objects in a Stereoscopic Virtual 3D Space

**Hossein Azari, Irene Cheng, and Anup Basu**

*Department of Computing Science, 2-21 Athabasca Hall,University of Alberta, Edmonton, Alberta, Canada T6G 2E8*

Correspondence should be addressed to Anup Basu, anup@cs.ualberta.ca

Received 30 April 2009; Accepted 11 September 2009

Academic Editor: Pietro Zanuttigh



We introduce a different approach of applying stereoscopy principles to implement a virtual 3D pointing technique called stereo 3D mouse cursor (S3D-Cursor) based on two or more views of an ordinary mouse cursor. The basics of such an idea have already been applied as a by-product of some stereo-based visualization applications with usually less attention to its strengths or weaknesses as a generic alternative of its 2D counterpart in stereoscopic 3D space. Here, we examine if such an idea satisfies all or the main expected requirements of an abstract 3D cursor. Moreover, we analyze its accuracy and evaluate the applicability of this approach in terms of different efficiency factors. For this purpose, we have adapted a real-time point-based rendering software called QSplat to a multiview rendering version named QSplatMV. We have implemented the S3D-Cursor on top of this new application and developed a simple editing toolset for manipulating the virtual 3D objects. Our user evaluation results suggest the effectiveness of the approach in terms of detection accuracy and user satisfaction compared to using an ordinary mouse cursor on a conventional 2D screen.

## 1. Introduction

The stereopsis capability of the human vision is one of the primary means used by our vision system to give us a 3D perception of the world [1]. This ability allows us to have a sense of the 3rd dimension from the two slightly different images of the world projected onto our eyes retina. Benefiting from this intrinsic ability, a wide variety of different stereoscopic devices and techniques are employed to create the illusion of depth from the visual stereo contents. Recent advances in these technologies enable the users to watch contents in 3D without using any filtering glasses (multiview autostereoscopic 3D displays [2]) or watch 3D contents in full-colour on conventional displays (colour code glasses [3]). Along with these advances in stereoscopic display technologies it is also necessary to evolve the current 2D/3D interaction techniques and devices for interplay with these new attractive virtual 3D environments. Among these, pointing to the targets (objects or other GUI components) using some devices such as mouse is probably the most appealing interaction method especially useful to work within the graphics environments [4]. For this reason, providing the possibility of pointing to any arbitrary voxel of the 3D space is one of the first expectations of any user who wants to migrate from 2D to 3D space applications. In this regard, several 3D pointing techniques (or in broader category 3D selection techniques) have been introduced during the years [4]. On the other hand, several commercial or experimental 3D input devices such as 3D space navigator [5], OptiBurst [6], and 3D air mouse [7] and many others are introduced to facilitate working with 3D objects in 2D/3D space by simplifying the fundamental tasks such as zooming, rotating, panning, and so forth.

Stereo-based 3D cursor is among the 3D pointing techniques, which usually has been implemented as a by-product of the stereoscopic 3D visualization (and/or 3D object manipulation) systems or applications, while less attention has been paid to its capabilities as a generic replacement of 2D mouse cursor in stereoscopic 3D space. The stereo 3D cursor (hereafter S3D-Cursor) can be achieved through providing two or more different views of the 2D mouse cursor at a specific disparity. The cursor depths can be controlled by adjusting the amount of the disparity. This enables the user to point to any arbitrary 3D location inside the virtual 3D space projected by a stereoscopic display.

In this paper, we study different aspects of this technique and show how this method satisfies the main functionality requirements of an abstract 3D cursor. Furthermore, in addition to discussing some issues such as accuracy and occlusion handling, we will show how the scene stereo content may be used to virtually enhance the performance of the technique according to 3D Fitts' law [4]. To realize the applicability of these ideas we have improved the QSplat [8], a single-view point-rendering software, to QSplatMV which enables the fast multi-view rendering of a 3D object described by a collection of 3D points or a triangular 3D mesh. We have implemented the S3D-Cursor on the top of this new application and developed a simple 3D editing toolset to manipulate the virtual 3D objects. The application is used to evaluate the capability of the 3D versus 2D cursor in manipulating 3D visualized data. Our user evaluation results suggest the effectuality of using the technique in term of several factors including detection accuracy, simplicity of usage, and overall user satisfaction compared to using an ordinary mouse on a conventional 2D screen. Regarding these advantages, the idea potentially can be used as a generic method in many applications including 3D games, 3D medical data manipulation, and 3D GUIs.

The remaining sections of this paper are organized as follow. In Section 2 we briefly discuss the relevant devices and techniques. In Section 3 we explain the fundamentals of stereo imaging and describe the mathematical model behind the S3D-Cursor. Section 4 is dedicated to the design and implementation aspects of S3D-Cursor. In Section 5 we focus on S3D-Cursor as a generic pointing technique in stereoscopic 3D space. In Section 6 we discuss the issues related to the accuracy of the stereo mouse and possible methods of improving its accuracy. Section 7 discusses the application aspects of our method and presents some of our application toolset outputs and user evaluation results. Finally, Section 8 is devoted to concluding remarks and possible future extensions.

## 2. Background and Related Works

Several efforts have been done during the years to simplify interaction with 3D application environments. On the one hand, benefiting from different mechanical, electromagnetic, optical, acoustic, and inertial sensors and technologies [9], several kind of 3D input devices are introduced to facilitate working with 3D models and applications. Among the recent commercial ones we may refer to some 3D mice such as space navigator and its counterparts which benefit from a pressure sensing technology on a controller cap to provide simultaneous panning, zooming, and rotation of the 3D object. In fact, the movements such as push, pull, twist, and tilt applied to the cap are translated to the appropriate movement of the 3D object [10]. The 3D air-mouse is another type of 3D computer mouse which uses the ultrasonic technology. A small ultrasonic transmitter is part of the mouse equipments which is worn on the index finger like a ring, and an array of receivers is used to track the movements of the transmitter (finger). These movements are translated to appropriate actions on the 3D object. For example, zooming is achieved by moving the hand closer to or farther from the screen [7]. IR tracking is also used in some products like OptiBurst to map the natural hand movement composed of translations and rotations (6DOF) to the appropriate actions inside the 3D application [6]. Some other ideas such as GlobeFish, GlobePointer, SqureBone and others are introduced in [11] which use a combination of sensor technologies to provide 6 or more DOF interaction devices.

On the other hand, several task-specific or general-purpose interaction techniques are developed [4] which in combination with the 3D input devices and technologies provide an easier, more natural way of interfacing with the 3D environments. These interactions may be classified into three different task domains: object selection and manipulation, viewpoint manipulation (navigation and travel), and application (system) control [11, 12]. Pointing to the targets (objects or other application components) may be considered as one of the most widely used interaction techniques in GUIs which is pre-requirement of performing many tasks in each of above-mentioned task domains. The pointing is usually achieved by manipulation of a 2D/3D cursor position and/or orientation using an appropriate input device. A *chronological* survey of several 3D pointing techniques and 3D cursor concepts are presented in [4]. This includes *Skitters and Jacks* (1987), *Ray casting* (1994), *Spotlight* (1994), *Virtual hand* (1995) and many other older or recent techniques. According to the review, all these methods are mainly based on virtual hand, ray, or spotlight pointing techniques. Later in the paper, a more formal definition for a 3D cursor is presented. The definition assumes 6 or more (at least 3 translational and 3 rotational) DOF for the pointing device and assumes that the selectable target has to be visible to the user and implies that a typical 3D cursor has to satisfy all the following requirements and constraints.

(a) Visual presentation—the 3D cursor has a graphical presentation which makes its position and orientation observable to the user.

(b) Behaviour—the 3D cursor movements are controllable by the user using an appropriate input device.

(c) Constraints—the 3D cursor reaches all the positions on the (3D) graphical display and is able to touch only one target at a moment.

Then, according to this definition, two main types of 3D cursors are proposed for 3D UIs: the 3D point cursor and the 3D line cursor. The author describes how these two main types satisfy all above-mentioned requirements and how all aforementioned 3D pointing techniques can be derived from these two main types. In fact, the other 3D pointing techniques may be considered as the result of exploiting some possible virtual enhancements to improve the performance of the 3D pointing (or the 3D target acquisition time). Here, a virtual enhancement means changing the values of one or more parameters effective in the 3D target acquisition time according to the 3D Fitts' law. The 3D Fitts' law states that the

target acquisition time or the average *movement time* ($MT$) to select a target depends on the *distance to move* ($A$), the *target size* ($W$: width, $H$: height, and $D$: depth of the target), and also the *viewing angle* ($\theta$) at which the target is seen by the user through the following equation:

$$MT \approx 56 + 508 \log_2 \left( \sqrt{f_W(\theta)\left(\frac{A}{W}\right)^2 + \frac{1}{92}\left(\frac{A}{H}\right)^2 + f_D(\theta)\left(\frac{A}{D}\right)^2} + 1 \right), \quad (1)$$

where $f_W(0^\circ) = 0.211$, $f_W(90^\circ) = 0.717$, $f_W(45^\circ) = 0.242$, $f_D(0^\circ) = 0.194$, $f_D(0^\circ) = 0.312$, and $f_D(45^\circ) = 0.147$ [4]. In this regard, reducing the movement distance $A$, increasing size of the target (or equivalently size of the cursor), or a combination of these changes are examples of such virtual enhancements.

Even though, the review in [4] does not refer to the stereo 3D cursor as a 3D pointing method, the technique is known for years and already applied in some stereoscopic 3D visualization and manipulation applications and GUIs. OrthoEngine 3D Stereo offers a 3D stereo cursor among its advanced tools for 3D viewing and manipulation of aerial pictures or satellite imagery data [13]. BioMedCache as an application for molecular design also offers stereoscopic 3D visualization and 3D stereo cursor [14]. In [15], authors present their success in modifying the functionality of X Window System with the purpose of constructing generic tools for displaying 3D-stereoscopic content. In this context, they refer to the implementation of the 3D pointer via creating a shadow pointer which follows the motion of the real pointer in both fields of the stereo window. In their implementation the depth of the 3D cursor is controlled by automatic adjustment of the disparity of the shadow pointer with respect to the real one. Moreover, recently the prototype of a simultaneous 2D/3D GUI for (auto)stereoscopic displays is introduced that refers to the implementation of a 3D stereo cursor. The stereo cursor is enabled upon entering of the mouse pointer into the 3D GUI area [16]. Here again the disparity is automatically adjusted to keep the virtual 3D cursor in touch with the surface of the 3D objects and 3D GUI components.

In spite of all these efforts, there are less focus on the studying the capabilities of the stereo cursor itself as a generic extension of 2D cursor to the stereoscopic 3D space. While such a study is necessary considering increasing popularity of some emerging technologies such as autostereoscopic displays and other stereo-based display techniques. In this paper, we will study different theoretical and implementation aspects of stereo cursor. Our implementation, which simply named as S3D-Cursor, composed of two or more views of the same 2D mouse cursor presented on the (auto)stereoscopic display screen at a specific disparity which essentially follows the same principles applied by others to form the 3D cursor. However, our implementation supports two disparity adjustment modes (manual and automatic) which in fact enables us to show that such a simple technique not only satisfies all the above-mentioned requirements of an abstract 3D cursor but also it is possible to improve the performance, flexibility, and simplicity of object selection and pointing tasks by applying several virtual enhancements. Furthermore, we discuss some practical issues such as S3D-Cursor accuracy and ambiguities caused by occlusions on autostereoscopic displays. Our implementation does not imply any assumption about the input pointing device and fully complies with the current functionality of the 2D mice. However, special input devices such as available 3D mice may facilitate performing 3D tasks. Moreover, the disparity calculations in most cases happen concurrent to the 3D model rendering process, therefore, it does not impose extensive additional computational load to the application. This issue totally will be resolved by providing a system level support for stereo cursor.

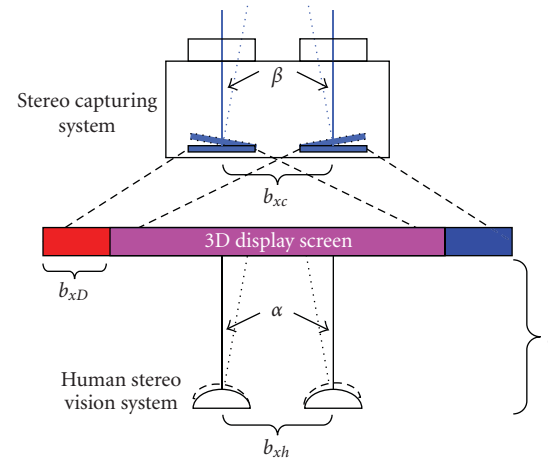



Figure 1: Process of capturing, display, and watching stereo images.

## 3. S3D-Cursor Mathematical Model

Before going into the details of the stereo 3D mouse design and implementation we briefly describe the stereoscopy process through establishing a supporting mathematical model. This mathematical model equally can be applied to the stereo 3D cursor formation process as well. We will use the following notation and definitions in this and the subsequent sections (see Figure 1):

$(X, Y, Z)$ : a 3D point, and $(\hat{X}, \hat{Y}, \hat{Z})$: the corresponding estimated 3D point

$(x_{l/r}, y_{l/r})$ : projection of a 3D point on the left/right image plane

$(\hat{x}_{l/r}, \hat{y}_{\mathbf{l/r}})$ : estimation of the projection of a 3D point on the left/right image plane, considering the nearest pixel

$e_x(e_y)$ : distance between two neighboring pixels in the $x$ ($y$) direction

$b_x$ : baseline of the stereo setup or horizontal displacement between the left and right images on a display screen

$\alpha(\beta)$ : eyes (cameras) vergence angle

$d$ : viewing distance

$f$ : focal length

$R$ : total resolution, that is, the total number of pixels over the unit square.

The subscripts $c$, $D$, and $h$ are also used to refer to a stereo imaging system (cameras), 3D display, and human eyes features, respectively. For example $f_c$ stands for the focal length of cameras while $f_h$ means the eyes' focal length.

Figure 1 illustrates the process of capturing, displaying and watching stereo images. Two different stereo systems are involved in this process: stereo capturing and human stereo vision. As showed in this figure, either of these two systems can have its own configuration independent of the other one. Restricting ourselves to the two basic stereo camera configurations, that is, parallel and with vergence ones, four different scenarios may happen in this process. In the simplest scenario we may assume parallel geometry for both capturing and viewing sides. Then, assuming a pinhole camera model [17], the projection of a 3D point $(X,Y,Z)$ on left and right camera image planes are given by:

$$x_{rc} = \frac{f_c X}{Z}, \qquad x_{lc} = \frac{f_c (X - b_{xc})}{Z}, \qquad y_c = y_{lc} = y_{rc} = \frac{f_c Y}{Z}. \tag{2}$$

The images captured by the stereo cameras are scaled by a factor $S$ and presented on the display. Thus, the corresponding 2D point coordinates on the display screen can be computed as:

$$x_{rD} = S x_{rc}, \qquad x_{lD} = S x_{lc}, \qquad y_D = S y_c \tag{3}$$

Finally, the 3D point projections on the eyes through a display medium placed at distance $d$ are obtained as

$$x_{rh} = \frac{f_h x_{rD}}{(f_h + d)}, \qquad x_{lh} = \frac{f_h (x_{lD} + b_{xD} - b_{xh})}{(f_h + d)}, \qquad y_h = \frac{f_h y_D}{(f_h + d)} \tag{4}$$

From the formulae (4) the 3D point reconstructed by human eyes is theoretically given by

$$Z_h = \frac{f_h b_{xh}}{x_{rh} - x_{lh}} = \frac{(f_h + d) b_{xh}}{(x_{rD} - x_{lD} - b_{xD} + b_{xh})}, \qquad X_h = Z_h \frac{x_{rh}}{f_h} = Z_h \frac{x_{rD}}{(f_h + d)}, \qquad Y_h = Z_h \frac{y_h}{f_h} = Z_h \frac{y_D}{(f_h + d)}. \tag{5}$$

In a more realistic scenario we may assume that there is a small vergence angle $\alpha$ acting on the human eyes when they are watching a stereo pair through a stereoscopic device. From the formulations established in [18] for the vergence-stereo configuration, the 3D point reconstructed by the eyes in this case is given by

$$Z_h = \frac{b_{xh} AB}{CD + EF}, \qquad X_h = Z_h \frac{D}{B}, \qquad Y_h = Z_h \frac{y_{rh}}{F}, \tag{6}$$

where

$$A = (f_h \cos\alpha + x_{lh} \sin\alpha), \qquad B = (f_h \cos\alpha - x_{rh} \sin\alpha)$$
$$C = (f_h \cos\alpha + x_{lh} \sin\alpha), \qquad D = (f_h \sin\alpha + x_{rh} \cos\alpha) \tag{7}$$
$$E = (f_h \sin\alpha - x_{lh} \cos\alpha), \qquad F = (f_h \cos\alpha - x_{rh} \sin\alpha)$$

and $x_{rh}$, $x_{lh}$, and $y_{rh}$ are given by:

$$x_{rh} = \frac{f_h (x_{rD} \cos\alpha - (f_h + d) \sin\alpha)}{(f_h + d) \cos\alpha + x_{rD} \sin\alpha}$$
$$x_{lh} = \frac{f_h ((x_{lD} + b_{xD} - b_{xh}) \cos\alpha + (f_h + d) \sin\alpha)}{(f_h + d) \cos\alpha - (x_{lD} + b_{xD} - b_{xh}) \sin\alpha} \tag{8}$$
$$y_{rh} = \frac{f_h y_D}{x_{rD} \sin\alpha + (f_h + d) \cos\alpha}.$$

If we apply the formulae in (8) into (6) to obtain the 3D point estimation in terms of display coordinates, after some simplification, the formulae in (5) are obtained again (details are skipped here). This means that if the stereo images presented on the display screen are captured under parallel configuration, then the 3D-scene reconstructed by human eyes, theoretically does not depend on the amount of vergence of the eyes.

For the other two possible scenarios the stereo images are captured under vergence. Therefore, assuming a vergence angle $\beta$ and again using formulations given in [18] the 3D point projections on the camera image planes are obtained as

$$x_{rc} = f_c \frac{X \cos\beta - Z \sin\beta}{Z \cos\beta + X \sin\beta}$$
$$y_{rc} = f_c \frac{Y}{X \sin\beta + Z \cos\beta}$$
$$x_{lc} = f_c \frac{(X - b_{xc}) \cos\beta + Z \sin\beta}{Z cos\beta - (X - b_{xc}) \sin\beta} \tag{9}$$
$$y_{lc} = \frac{f_c Y}{Z cos\alpha - (X - b_{xc}) sin\beta}$$

Thus, the coordinates of these projections after representing on the display by a scale factor $S$ are

$$x_{rD} = S x_{rc}, \qquad y_{rD} = S y_{rc}$$
$$x_{lD} = S x_{lc}, \qquad y_{lD} = S y_{lc} \tag{10}$$

Contrary to the first and the second scenarios, here the corresponding projections on the display screen do not locate

on the same raster line. However, as we discussed in [19] assuming the eyes are able to establish conformance between the corresponding projections in the left and right views, then $Z_h$ and $X_h$ can be calculated using either Equations (5) (assuming parallel geometry for human eyes) or Equations (6) (assuming vergence geometry for human eyes). If we assume that eyes compensate for vertical differences of the corresponding points (see our justifying experiments in [19]) then we can say that the 3D point location estimation by human eyes mainly depends on the horizontal disparity of the corresponding projections in stereo pairs. Regarding this simplification assumption, we can apply Equations (5) as a good (approximate) model for 3D point estimation (3D cursor position estimation) by human eyes via stereo images (2D cursor images) presented by a stereoscopic device in all above-mentioned possible scenarios.

## 4. S3D-Cursor Implementation

For the best realization of the S3D-Cursor, it was necessary to implement an underlying (multiview) stereo rendering application. For this purpose, we have extended the QSplat to support rendering multiple views of a 3D object. QSplat is a real-time point-based rendering program which uses a progressive hierarchical method especially useful to render large geometric models composed of huge number of 3D point descriptions [8]. The program uses OpenGL as its graphic library to implement different types of rendered-point primitives called splat. Our new extension which is called QSplatMV enables the user to decide on the number of cameras (views), the distance between the cameras ($b_{xc}$), and the amount of horizontal displacement of the views on the display screen ($b_{xD}$). The current version assumes parallel configuration and the same baseline for all cameras. This allows a simplified implementation of the cameras' rotation and translation. These movements are applied to a central virtual camera and then the position and direction of all cameras are set with respect to this virtual camera. The cursors' disparities calculations are also simplified under parallel geometry. The system also supports a special red/blue rendering mode which gives the flexibility of using the application on all conventional displays just by wearing simple anaglyph glasses.

*4.1. Binocular Implementation.* The stereo mouse cursor is implemented on the top of the QSplatMV. Two different disparity adjustment modes are considered for the cursors: manual and automatic. The automatic mode is more useful in targeting the existing objects in the applications such as 3D games and 3D object manipulation tools where the scene 3D information is already available. In the automatic mode one view, say the left one, is considered as the reference view. When the left mouse cursor points at a pixel in the left view, having the depth information of the pixel (usually available in the depth buffer) and the camera parameters, the corresponding pixel on the right view (or the position of the right cursor) can be determined using the basic stereo imaging formulations as follow.

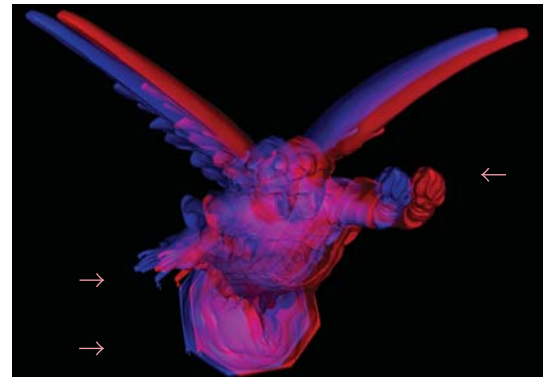


Figure 2: Lucy red-blue stereo pair with three samples of stereo mouse cursor presented at different distances (disparities).

Considering the OpenGL default perspective projection, which implies a normalization transformation as well [20], the relationship between the point depth in the virtual camera coordinate $Z_c$ and the depth maintained in the depth buffer $Z_w$ can be stated as

$$Z_c = \frac{f_c d_{\text{far}}}{Z_w (d_{\text{far}} - f_c) - f_c}, \tag{11}$$

where $d_{\text{far}}$ is the far clipping plane distance and $f_c$ is the near clipping distance or camera focal length. On the other hand, from (2) $Z_c$ can be determined as

$$Z_c = \frac{f_c b_{xc}}{x_{rc} - x_{lc}} = \frac{f_c b_{xc}}{\text{disp}} \tag{12}$$

From (11) and (12) the amount of the disparity is obtained as

$$\text{disp} = \frac{(Z_w (d_{\text{far}} - f_c) - d_{\text{far}}) b_{xc}}{d_{\text{far}}}. \tag{13}$$

The disparity calculated in (13) should be properly scaled and adjusted considering the viewport transformation settings and the amount of the views displacement $b_{xD}$.

As already mentioned, the automatic disparity adjustments causes the illusion of touching the surface of the real 3D object so that when the user slides the mouse pointer over the display screen the virtual 3D cursor follows the holes or other irregularities of the 3D object surface. Figure 2 shows a red-blue stereo pair of Lucy model with three samples of stereo mouse cursor at different disparities which actually ensemble three cursors at three different distances from viewer (watch this figure using red-blue anaglyph glasses to see the formation of these 3D cursors at different distances).

Implementing the automatic disparity adjustment over the image/video stereo pairs implies that an efficient stereo matching algorithm be incorporated into the system to find the corresponding projections in the left and right views. Contrary to the classic stereo matching algorithms, which involves establishing correspondence for all pixels, here the correspondence need to be find only for the pixel located

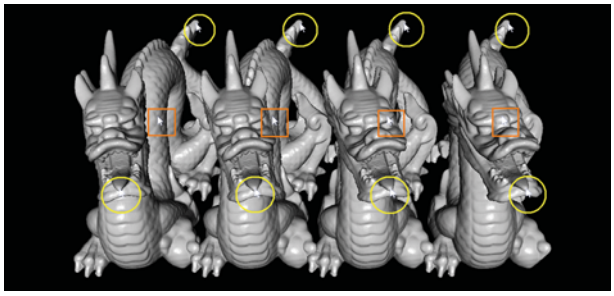


Figure 3: Dragon four-view stereo with three samples of stereo cursors on occluded (squares) and nonoccluded (circles) areas.

under the current position of the left (or alternatively right) mouse cursor. This assumption may lead to more efficient algorithms for real-time applications.

In the manual mode the user is able to change the depth of the 3D cursor by manually adjusting the disparity between the left and right cursor. The manual mode is useful when the user wants to point somewhere other than the visible surface of the 3D object or to adjust the disparity estimated by a stereo matching algorithm. The latter case especially can be used as a cost-effective, accurate method for extracting ground-truth data from stereo image pairs.

*4.2. Multiview Implementation.* The multiview implementation is particularly useful for autostereoscopic multiview displays. The implementation is essentially similar to two-view case. Assuming all cameras are parallel and located on the same baseline at equal distances then all corresponding projections of a 3D point on the display screen will be located at the same raster line with the same disparity. As result, again one of the views can be considered as reference for disparity calculations and the other corresponding projection can be determined with respect to the reference view.

Although, this implantation is pretty simple, it may cause some problems in occluded areas. This is because that in the autostereoscopic 3D displays the viewer is only able to see two consecutive views of the scene at a time. As result, as depicted in Figure 3, the implementation works fine as while as the corresponding projections, similar to the instances inside the circles, are visible in all views. The implementation becomes problematic when corresponding projections fall into occluded areas in two or more consecutive views. For example observe the third and the forth view of the cursors denoted by squares. When the viewer moves his/her head to watch the third and the forth views, the 3D position of the 3D cursor reconstructed from these two cursor views, is incorrectly estimated. The problem could be fixed if each view serves as the reference for the next adjacent view. However, this may lead to the ambiguity in converting the cursor positions to a unique 3D position. A more advanced algorithm may detect the occluded areas and hide or highlight the cursor in corresponding views. Upon receiving such a hint, the user may change the viewpoint or the cursor position to access a specific point from any desired view.

## 5. S3D-Cursor as a Generic 3D-Pointing Technique

In general, S3D-Cursor can be classified as a 3D point cursor which satisfies all assumptions and requirements of an abstract 3D cursor as follows.

(i) It has obviously three translational DOF, and three rotational DOF can be achieved by viewpoint manipulation.

(ii) Although the occlusion problem on autostereoscopic displays may necessitate a further move to grab the target, all visible parts of the scene (all visible targets) are selectable by the user.

(iii) The cursor has a visual presentation which in addition to its position may also make its orientation observable to the user.

(iv) The S3D-Cursor movements are simply controllable by the user using a conventional mouse; however, more appropriate input devices such as 3D mice may be adapted for efficiency purposes.

(v) The S3D-Cursor is able to reach to all positions in the comfortable viewing range of the stereoscopic graphical display by appropriate (manual) adjustment of the disparity. This is especially important when the user aims to point to an empty space for example for the purpose of creating a new object.

(vi) Finally, since in a stereoscopic 3D space the user only sees the surface of the opaque objects, so he/she will be able to touch only one target at a moment in automatic mode (a priority mechanism may be applied for transparent or translucent objects). In manual mode appropriate visual hints may be implemented to aware the user of moving the cursor to the physically invisible areas.

Regarding these properties, the S3D-Cursor can be applied as a general alternative of 2D cursor in stereo space. Some virtual enhancements may be implemented to improve the basic functionality of S3D-Cursor. In fact, auto disparity adjustments may already be considered as such an enhancement which virtually reduces the distance to the target (reduces $A$ in (1)) by "*removing the empty space between the cursor and the targets*"—the enhancement that according to [4] not already tried in other 3D pointing techniques. "*Increasing the target size*"—that is, increasing $W$, $H$, or $D$ in (1)—is another enhancement which especially useful in handling the accuracy deficiencies of the stereo cursor (see Section 5). If the application control components are also implemented in 3D, then several enhancements may be applied on the GUI components especially on menus and the application window itself. These include appearance of the pop-up menus on the same depth of the 3D cursor, and dynamically managing the position and size of the windows depending on the scene composition and user actions. These types of enhancements also may be considered as virtual reduction of distance to the target.

## 6. S3D-Cursor Accuracy Analysis

Considering the Equations (5) in Section 3, the whole stereoscopy system depicted in Figure 1 can be considered as a (parallel) stereo camera system whose focal length is equal to the human focal length plus the viewing distance ($f_h + d$), the distance between its cameras or its baseline length is equal to the distance of the human eyes ($b_{xh}$), and the display screen displaced to the left/right has the role of its left/right image plane. Assuming $f_h$, $d$, and $b_{xh}$ are constant values, the accuracy of the reconstructed 3D points or the stereoscopic resolution mainly depends on the size (width and height) of the pixels of the display. In fact, as understood from Equations (5), the width and the height of the pixel have different contribution on the accuracy of the 3D point estimation from stereo. Figure 4 shows the comparison between the maximum possible estimation errors on each coordinate component with respect to the different pixel aspect ratios for a single pixel using the typical values mentioned in Table 1. The errors are calculated as the difference of the values obtained for $(X_h, Y_h, Z_h)$ using Equations (5) and corresponding maximum deviated values obtained from following equations ($b_{xh}$ is assumed to be equal to $b_{xD}$):

$$\hat{Z}_h^{\max} = \frac{(f_h + d)\, b_{xh}}{(x_{rD} - x_{lD} - e_x)}$$

$$\hat{X}_h^{\max} = \frac{x_{rD} + (e_x/2)}{(x_{rD} - x_{lD} - e_x)}, \qquad \hat{Y}_h^{\max} = \frac{y_D + \left(e_y/2\right)}{(x_{rD} - x_{lD} - e_x)}. \tag{14}$$

Figure 4 shows that although the estimation error for $Y$ component is a little increased with smaller aspect ratios, the average estimation error is decreased. Particularly, the error in estimation of $Z$ is considerably decreased which means that a finer horizontal discretization on display screen yields in a finer depth resolution.

We have generalized this concept in our previous research [21] and through theoretical analysis shown that in general for a typical stereo setup and a given total resolution $R$, a more finer horizontal discretization ($e_x$) versus vertical discretization ($e_y$) yields in less 3D point estimation error. This has been achieved by putting an upper bound on the depth estimation error and then minimizing the error with respect to horizontal to vertical discretization ratio. For a typical parallel stereo setup the procedure can be briefly described as follow (see [21] for details):

$$\hat{Z} = \frac{f b_x}{\hat{x}_r - \hat{x}_l} = \frac{f b_x}{(x_r - x_l) \pm e_x} = Z\left(1 \pm e_x \frac{Z}{f b_x}\right)^{-1}. \tag{15}$$

Assuming higher order terms in Tailor expansion of (15) are negligible:

$$\hat{Z} \cong Z\left(1 \pm \frac{e_x Z}{f b_x}\right), \tag{16}$$

and then

$$\hat{Y} = \hat{Z}\frac{\hat{y}}{f} \cong \frac{Z}{f}\left(1 \pm \frac{e_x Z}{f b_x}\right)\left(y \pm \frac{e_y}{2}\right) \tag{17}$$

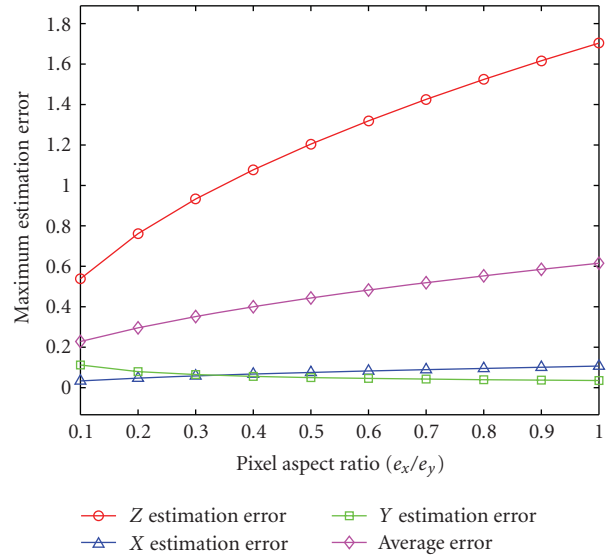



Figure 4: Comparison of maximum error on estimation of 3D point coordinate components at different pixel aspect ratios.

Table 1: Values used for calculating maximum error on estimation of 3D point coordinate components.

| Parameter | Value | Parameter | Value |
|---|---|---|---|
| $f_h$ | 17 mm | $R$ | 17.76 pixel/mm$^2$ |
| $b_{xh}$ | 65 mm | $d$ | 500 mm |
| $x_r$ | 22 mm | disparity | 22 mm |
| $y$ | 0 mm | | |

or

$$\left|\frac{\hat{Y} - Y}{Y}\right| \le f_p(e_x) = \left\{\frac{e_x Z}{f b_x} + \frac{e_y}{2|y|} + \frac{e_x e_y Z}{2f|y| b_x}\right\}. \tag{18}$$

Considering a unit viewing or image capture area:

$$\left(\frac{1}{e_x}\right)\left(\frac{1}{e_y}\right) = R \quad \text{or} \quad e_y = \frac{1}{e_x R}. \tag{19}$$

Thus, from (18) and (19) we can restate the obtained upper bound on relative estimation error on $Y$ component as:

$$f_p(e_x) = \left\{\left(\frac{Z}{f b_x}\right)e_x + \left(\frac{1}{2R|y|}\right)\frac{1}{e_x} + \frac{Z}{2f|y|Rb_x}\right\}. \tag{20}$$

Minimizing equation (20) with respect to $e_x$ gives the optimal pixel width (and then optimal pixel aspect ratio) for a single 3D point. In practice the estimation error need to be minimized over a reasonable viewing volume and some other parameters such as vergence may also be included in the optimization process (see [22, 23]).

In [19] we have applied this theory to the above-mentioned stereo setup (Equations (5)) and shown for a typical desktop computer or laptop the 2 : 3 pixel aspect ratio (i.e., three horizontal versus two vertical pixels) gives a better 3D visual experience than the uniform (square)

Table 2: S3D-Cursor user evaluation criteria.

| Experiment | Criterion |
|---|---|
| Experiment 1 | C1: Stereo mode gives better 3D visual experience |
| | C2: S3D-Cursor helps in better 3D experience |
| Experiment 2 | C3: Object (boundaries) are better distinguishable in stereo mode |
| | C4: S3D-Cursor improves distinguishability |
| Overall | C5: Simplicity of using S3D-Cursor |
| | C6: Satisfaction and usefulness of S3D-Cursor |

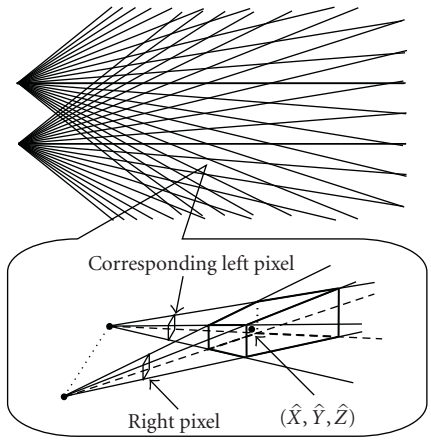



Figure 5: Top: Resolution pattern for parallel-stereo geometry. Bottom: 3D view of a typical voxel formed by two corresponding pixels.

pixel distribution. Applying the same theory to the stereo mouse cursor, we can say that a finer horizontal resolution not only yields in a better 3D visualization but also more accurate stereo-based 3D pointing especially across the depth dimension. However, this is a hardware solution which requires establishing new standards for stereo capturing and displaying devices.

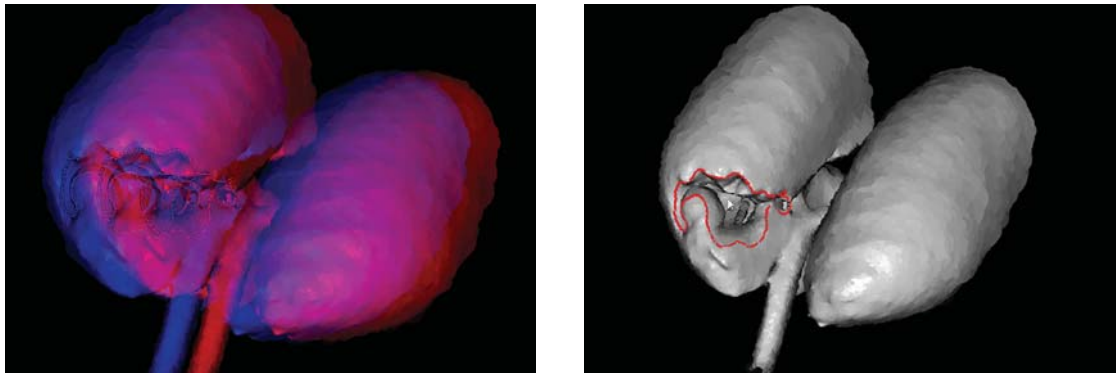
Another issue related to the stereo 3D mouse accuracy is its heterogeneous behaviour mainly across the depth dimension. This fundamentally results from the intrinsic behaviour of the perspective projection plus the discretized nature of digital images. As illustrated in Figure 5, all points located inside the 3D diamonds (voxels) formed by the corresponding pixels are estimated to the same 3D point. These voxels are non-uniformly distributed so that the resolution (especially the depth resolution) decreases with the distance to the stereo cameras (viewer) [24]. As result, the stereo mouse cursor is not accurate enough when the objects are not close enough to the viewer. Several virtual enhancements may be applied to compensate for this drawback up to some acceptable extent. Zooming across the third dimension (*bringing the target to the cursor*), enlarging the objects across the depth (*changing the target size* across the depth, i.e., changing the $D$ parameter), or manipulating the target under a fish-eye implementation (which again may be interpreted as some sort of *bringing target to the cursor*) are among possible enhancements. Another possibility is using the 3D object information when such information is available. This is different from simply calculating the underneath disparity and can be quite helpful for disambiguation of the targeted object where the objects are not close enough to be distinguished by the amount of the disparity. In this situation, the closest 3D object (or 3D object element) to the estimated 3D cursor position may be prioritized as the targeted object.

## 7. Practical Application of S3D-Cursor

Stereo 3D mouse cursor can be used in a wide range of applications including 3D computer games and 3D objects manipulation. To realize the applicability of the S3D-Cursor on manipulating 3D objects we have developed a simple 3D object manipulation toolset which contains a marking (a pen) and demarking (an eraser) tool with a few auxiliary displaying-state control tools which all together allow the user to mark/demark a desired 3D point in the virtual stereoscopic 3D space.

Figure 6 shows a sample result of applying the 3D editing toolset (automatic mode) for ground-truth data measurement. Here the S3D-Cursor is used to specify a 3D-contour surrounding a TB cavity on a 3D lung model. The red-blue mode of QSplatMV is used in order to provide stereo effect of the model in a virtual 3D environment. The left lung image shows the red-blue representation and the right image shows the corresponding single-view version of the same model and the same contour defined in the red-blue visualization mode. Although these images are degraded due to down-scaling, the reader should still be able to watch the stereo effect in 3D using simple anaglyph glasses and compare it with the corresponding 2D image.

*7.1. User Evaluation.* We used the QSplatMV and its editing toolset to evaluate stereoscopic visualization and manipulation of 3D objects versus the similar tasks in corresponding ordinary perspective representation. Again the red-blue mode of QSplatMV is used to provide the stereo effect. We conducted two sets of experiments. Table 2 shows a brief description of the criteria used in these experiments. None of the people who participated in these experiments had former experience of working with stereo cursor, but many of them had one time or more experience of watching stereo contents. Moreover, some of them were already familiar with some 3D input devices such as space navigator [10] and a few with some line cursors used to point to a 3D location within the CAVE VR systems [25].



(a) (b)

Figure 6: Marking the TB cavity boundaries on a 3D lung model using S3D toolset. Left: red-blue stereoscopic representation mode. Right: ordinary perspective representation.

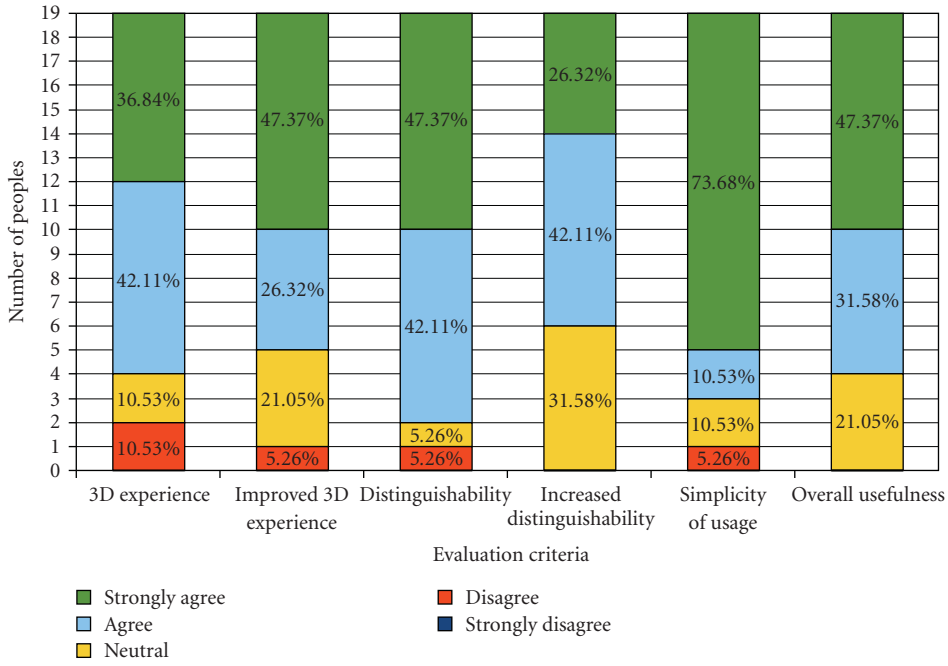



Figure 7: S3D-Cursor user evaluation results.

In the first experiment the peoples are asked to compare their visual experience with 3D stereo visualization versus 2D visualization. The purpose of this experiment was to show that if the people have better understanding of the 3D shape of the objects when they are able to touch the surface of the objects using S3D-Cursor. Three different models, that is, Lucy, Dragon, and Donna randomly chose and used for performing this experiment. As the first criterion in this experiment we wanted to make sure that the participants are able to observe the 3D effects of the stereo using anaglyph glasses and also to separate the 3D feeling caused by stereo and those possible 3D experience that may be caused by 3D cursor movements itself. Even though the red-blue representation may cause some ghosting effects for some peoples, the majority of peoples (strongly) agree that they have better 3D experience with stereo mode without the intervention of the 3D cursor (see Figure 7, column 1). The second criterion asks the peoples if the S3D-Cursor helps in a better 3D visual experience when they touch different parts of the object. Again majority of users believe that when they slide the cursor over the 3D object, the depth movements of the 3D cursor gives them a better sense of its 3D shape (Figure 7, column 2). According to these results, contrary to the 2D cursor which may cause some disturbance in the formation of the 3D environment, the S3D-Cursor helps in better visual experience via giving the users the interesting feeling of touching the surface of the virtual 3D object.

In the second experiment we asked the user to perform the same manipulation task over the 3D Lung model in both single and stereo view modes. In fact, we asked them

to mark the boundaries of a TB cavity in both 2D and 3D representations. Again our user surveys suggest that the S3D-Cursor technique provides better depth information (Figure 7 column 3) and the users can specify the region of interest more accurately compared to using a 2D mouse cursor on a 2D display (Figure 7 column 4). In fact, 3D visualization allows the user to better distinguish the convex and concave surfaces and the border areas.

Finally, we asked the users to express their overall opinion in terms of the simplicity of usage and the usability of the S3D-Cursor through their short work experience with S3D-Cursor. The majority of people strongly agreed that it can be used as simple as an ordinary mouse and many of them have found it a useful device for manipulating 3D objects.

## 8. Conclusion and Future Work

In this paper we discussed on different theoretical aspects of a stereo-based 3D pointing technique which enables the users to point to an arbitrary location inside the 3D space composed by a stereoscopic 3D display and described our approach toward the implementation of such a 3D cursor. We also discussed how such a technique satisfies all the requirements of an abstract 3D cursor, so that it potentially can be considered as a simple extension of 2D cursor to the 3D stereoscopic space. The user experience with the S3D-Cursor has shown us that it has great potential for interaction with this virtual space. Here we showed its usefulness and efficiency in working with 3D objects and presented some promising results. However, further improvement is necessary to overcome some minor drawbacks like low accuracy in the farther depths. This can be achieved through the implementation of suggested virtual enhancements. Also we need to improve our UI tools for further smart manipulation of 3D contours, surfaces and volumes based on the more advanced new colorful stereoscopic visualization techniques.

## Acknowledgments

The authors wish to thank iCORE and NSERC for financial support, the Stanford Computer Graphics Laboratory and the Stereolithography Archive at Clemson University, VCG-ISTI (the AIM@SHAPE Shape Repository) for providing 3D models, and Alexey Abadalov for extracting the 3D lung model.

## References

[1] N. Holliman, "3D display systems," Department of Computer Science, University of Durham, August 2009, http://www.dur.ac.uk/n.s.holliman/Presentations/3dv3-0.pdf.

[2] N. A. Dodgson, "Autostereoscopic 3D displays," *Computer*, vol. 38, no. 8, pp. 31–36, 2005.

[3] S. E. B. Sorensen, P. S. Hansen, N. L. Sorensen, et al., "Method for recording and viewing stereoscopic images in color using multichrome filters," US patent no. 6687003, August 2009, http://www.patentstorm.us/patents/6687003.html.

[4] N.-T. Dang, "A survey and classification of 3D pointing techniques," in *Proceedings of the IEEE International Conference on Research, Innovation and Vision for the Future (RIVF '07)*, pp. 71–80, Hanoi, Vietnam, March 2007.

[5] P. Best, "Ultimate 3D design navigation tool," *Gizmag Emerging Technology Magazine*, August 2009.

[6] "Mouse (Computing)," August 2009, http://en.wikipedia.org/wiki/Mouse_(computing).

[7] L. Blain, "The 3D air-mouse you wear as a ring," *Gizmag Emerging Technology Magazine*, August 2009.

[8] S. Rusinkiewicz and M. Levoy, "QSplat: a multiresolution point rendering system for large meshes," in *Proceedings of the ACM SIGGRAPH Conference on Computer Graphics (SIGGRAPH '00)*, pp. 343–352, New Orleans, La, USA, July 2000.

[9] C. Hand, "A survey of 3-D input devices," Tech. Rep. TR94/2, Department of Computer Science, De Montfort University, Leicester, UK, 1994.

[10] "What is a 3D mouse?" August 2009, http://www.3dconnexion.com/3dmouse/what_is_3dmouse.php.

[11] B. Frohlich, J. Hochstrate, A. Kulik, and A. Huckauf, "On 3D input devices," *IEEE Computer Graphics and Applications*, vol. 26, no. 2, pp. 15–19, 2006.

[12] C. Hand, "A survey of 3D interaction techniques," *Computer Graphics Forum*, vol. 16, no. 5, pp. 269–281, 1997.

[13] "Orthoengine 3D stereo," Technical Specification, August 2009, http://www.pcigeomatics.com/pdfs/3D_Stereo.pdf.

[14] "Biomedcache user guide," Software User Guide, August 2009, http://www.ch.ic.ac.uk/local/organic/medchem/UserGuide.pdf.

[15] S. A. Safier and M. W. Siegel, "3D-stereoscopic X windows," in *The International Society for Optical Engineering*, vol. 2409 of *Proceedings of SPIE*, pp. 160–167, San Jose, Calif, USA, 1995.

[16] F. Steinicke, G. Bruder, K. Hinrichs, and T. Ropinski, "Simultane 2D/3D user interface konzepte für autostereoskopische desktop-VR systeme," in *Proceedings of the 4. GI-Workshop AR/VR*, pp. 125–132, Shaker, Organization: GI-Fachgruppe VR/AR, 2007.

[17] R. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision*, Cambridge University Press, London, UK, 2003.

[18] H. Sahabi and A. Basu, "Analysis of error in depth perception with vergence and spatially varying sensing," *Computer Vision and Image Understanding*, vol. 63, no. 3, pp. 447–461, 1996.

[19] H. Azari, I. Cheng, and A. Basu, "Optimal pixel aspect ratio for stereoscopic 3D displays under practical viewing conditions," in *Proceedings of the 3DTV Conference: The True Vision—Capture, Transmission and Display of 3D Video (3DTV-CON '09)*, pp. 1–4, Potsdam, Germany, May 2009.

[20] K. E. Hoff, "Deriving the openGL perspective depth transformation," Tech. Rep., August 2009, http://www.cs.unc.edu/~geom/HOFF/techrep/perspective.doc.

[21] A. Basu, "Optimal discretization for stereo reconstruction," *Pattern Recognition Letters*, vol. 13, no. 11, pp. 813–820, 1992.

[22] I. Cheng, K. Daniilidis, and A. Basu, "Optimal aspect ratio under vergence for 3D TV," in *Proceedings of the 3DTV-Conference: The True Vision—Capture, Transmission and Display of 3D Video (3DTV-CON '08)*, pp. 209–212, Istanbul, Turkey, May 2008.

[23] I. Cheng and A. Basu, "Optimal aspect ratio for 3D TV," in *Proceedings of the 1st International Conference on 3DTV (3DTV-CON '07)*, pp. 1–4, Kos, Greece, May 2007.

[24] L. Matthies and S. A. Shafer, "Error modeling in stereo navigation," *IEEE Journal of Robotics and Automation*, vol. 3, no. 3, pp. 239–248, 1987.

[25] S. Zhang, Ç. Demiralp, D. F. Keefe, et al., "An immersive virtual environment for DT-MRI volume visualization applications: a case study," in *Proceedings of the IEEE Visualization Conference*, pp. 437–440, 2001.