Hindawi Publishing Corporation
Advances in Multimedia
Volume 2008, Article ID 697454, 17 pages
doi:10.1155/2008/697454

Research Article

Morphable 3D-Mosaics: A Hybrid Framework for Photorealistic Walkthroughs of Large Natural Environments

Nikos Komodakis and Georgios Tziritas

Computer Science Department, University of Crete, Heraklion, P.O. Box 2208 Heraklion, Greece

Correspondence should be addressed to Nikos Komodakis, komod@csd.uoc.gr

Received 28 September 2007; Revised 16 January 2008; Accepted 2 May 2008

Recommended by Dan Lelescu

This paper presents a hybrid (geometry- & image-based) framework suitable for providing photorealistic walkthroughs of large, complex outdoor scenes at interactive frame rates. To this end, based just on a sparse set of real stereoscopic views from the scene, a set of *morphable 3D-mosaics* is automatically constructed first, and then, during rendering, a continuous morphing between those 3D-mosaics that are nearby to the current viewpoint is taking place. The morphing is both photometric, as well as geometric, while we also ensure that it proceeds in a physically valid manner, thus remaining transparent to the user. The effectiveness of our framework has been demonstrated in the 3D visual reconstruction of the Samaria Gorge in Crete, which is one of the largest and most beautiful gorges in Europe.

1. INTRODUCTION

One of the main research problems in computer graphics is the creation of modeling and rendering systems capable to provide photorealistic representations of complex, real-world environments. Minimal human intervention during the modeling process and operation in real time during rendering, a property that allows walkthroughs at interactive frame rates, are some of the desirable characteristics for such systems. Two are the most prominent approaches that have been proposed so far in this regard. The first, more traditional, way of building this type of systems is by taking a purely geometric approach. In this case, a full 3D geometric model of the whole scene is constructed first, and that model is then used for the rendering of the scene afterwards. One big advantage of this approach is that it offers great flexibility with respect to manipulating or editing the scene's properties. For example, due to a full 3D model being available, operations such as altering the viewpoint of the camera, changing the position of light sources, modifying part of the geometry, and so forth can be attained very easily in this case. The price for this flexibility, however, lies in constructing the underlying 3D model of the scene needed in the first place. Unless done automatically, this can be an extremely time consuming and daunting task, especially for large-scale scenes. Due to this fact, a lot of effort went recently into how to obtain such a geometric model automatically by using only real images as input [1, 2]. This task is now known as *multiple view geometry* in the computer vision literature, and has been a very active research topic over the last years [3–5]. A characteristic method in this regard is the work of Pollefeys et al. [6] on 3D reconstruction from hand-held cameras, as well as the work of Debevec et al. [7] on recovering the geometry of architectural-type scenes, just to name a few of the proposed methods. For scenes with irregular geometry, however, such as outdoor scenes, or scenes with trees and plants, extracting an accurate 3D model still remains a very difficult task in the general case.

Relatively recently, however, it was realized by researchers that geometry is often unnecessary for obtaining a realistic representation of the scene [8]. As a result, the so-called image-based rendering (IBR) methods have emerged. These methods completely skip the geometric modeling of the scene and, instead, concentrate just on how to fill the image pixels corresponding to a novel view. In this case, geometric modeling is replaced by simply capturing a dense

set of real images, with each one of these images associated to a different view of the scene. When given a novel view, these methods then try to synthesize it simply by (appropriately) resampling the previously captured set of views. Intuitively, IBR methods can be thought of as trying to assemble a puzzle. The puzzle to be assembled is the novel view itself, while the pieces of the puzzle correspond to the pixels contained in the captured views. Since the invention of the concept of image-based rendering, many such methods have been proposed (see [9] for an excellent survey), with Lightfield [10] and Lumigraph [11] being two of the most characteristic examples in this regard. The biggest advantage of all IBR techniques is the extremely high level of photorealism they can attain (since they make use of actual images of the scene). The price for this, however, is the huge amount of data (i.e., input images) often required by IBR methods, which actually forms one of their main disadvantages. To see why this needs to be so in general, it is worth considering another interpretation of IBR techniques. According to it, these methods try to reconstruct the so-called *plenoptic function* [12]. This is a 7-dimensional function $\mathcal{P}(\mathcal{V}_x, \mathcal{V}_y, \mathcal{V}_z, \theta, \phi, \lambda, t)$, which models a 3D dynamic environment by recording the light rays at every space location $(\mathcal{V}_x, \mathcal{V}_y, \mathcal{V}_z)$, towards every possible direction (θ, ϕ), over any range of wavelengths λ and at any time t. Each time we capture an image by a camera, the light rays passing through the camera's center of projection are recorded and so that image can be considered as a specific sample of the plenoptic function. In this context, image-based rendering can be interpreted as a signal processing task, that is, that of *reconstructing a continuous signal (in this case the plenoptic function) based only on a discrete set of samples from that signal.* Given the high dimensionality of the plenoptic function, the reason why a huge number of input images is, in general, required by IBR techniques then follows directly from the well-known Nyquist theorem.

This issue with the huge amount of input data is also one of the reasons why the great majority of existing IBR techniques apply only to scenes of either small or medium scale. On the contrary, much fewer IBR methods have been proposed for the case of large-scale scenes. Motivated by this fact, the work presented in this article describes a novel, hybrid (i.e., both geometry-based and image-based) system, that can deal efficiently with the visual reconstruction of large-scale, natural outdoor environments. We note that this reconstruction is always photorealistic, while its visualization can take place at interactive frame rates, that is, in real time. For reducing the otherwise huge amount of input data, our system makes use of a new compact data representation of a 3D scene, called *morphable 3D-mosaics* [13, 14]. As we will see, this scene representation consists of a set of morphable (both geometrically and photometrically) 3D models. Furthermore, these morphable models are constructed automatically based just on a sparse set of real stereoscopic views. Due to the above-mentioned properties, our system can be extremely useful in applications such as 3DTV, gaming, virtual reality, and so on, where one seeks to create a 3D depth impression of the observed scene. In fact, the latter has been one of the starting motivations for our system, which has thus already been used as a part of a 3D virtual reality installation in the Natural History Museum of Crete with the goal of providing a lifelike virtual tour of the Samaria Gorge (one of the largest and most magnificent gorges in Europe).

Besides the proposal of a novel hybrid representation for a 3D scene, our system also includes the following new algorithms and techniques as part of its image-based modeling and rendering pipeline: (1) a robust photometric morphing procedure that is based on automatically extracting a dense field of 2D correspondences between wide-baseline images (Section 5.1), (2) an efficient geometric morphing algorithm based on estimating 3D correspondences between local geometric models (Section 5.2), (3) a robust algorithm for constructing 3D panoramas by combining local 3D models (Section 7) and (4) a highly optimized rendering pipeline that runs entirely on the GPU (thanks to employing pixel and vertex shaders for the morphing), while it also uses decimated, but visually correct, 3D geometric models (Section 6).

Of course, besides our system, other IBR techniques for large-scale scenes exist as well. To this end, work on unstructured/sparse lumigraphs has been proposed by various authors. One such example is the work of Buehler et al. [15]. These authors make use of a fixed geometric proxy that supposedly describes the global geometry of the scene in order to reduce the required amount of input data. However, this assumption of a fixed geometric proxy is not adequate in our case. Hence, contrary to [15], our system actually uses view-dependent geometry due to the continuous geometric morphing taking place. Another example of a sparse lumigraph is the work of Schirmacher et al. [16]. Although multiple geometric proxies are allowed in this work, any possible inconsistencies existing between these proxies (e.g., due to errors) are not taken into account during rendering. This is again in contrast to our work, where an "optical flow" estimation between wide-baseline images is used for dealing with this issue. Furthermore, this estimation significantly reduces the required number of input views (and hence the amount of input data). For these reasons, if any of the above two approaches were to be applied to large-scale scenes, like those handled in our case, many more images (than ours) would then be needed. In addition, due to our rendering path, which can be highly optimized in modern graphics hardware, we can achieve very high frame rates during rendering, whereas the corresponding frame rates listed, for example, in [16] are much lower due to an expensive barycentric coordinate computation that needs to take place. In the "interactive visual tours" approach [17], video (from multiple cameras) is being recorded as one moves along predefined paths inside a real world environment, and then image-based rendering techniques are used for replaying the tour and allowing the user to move along those paths. In this way, virtual walkthroughs of large scenes can be generated. However, a specialized acquisition system is needed in this case, whereas our method requires using just off the shelf digital cameras. Also, in the "sea of images" approach [18], a set of omnidirectional images are captured for creating interactive walkthroughs of large, indoor environments.

However, this set of images needs to be very dense with an image spacing of about 1.5 inches. Finally, the authors in [19] have proposed an IBR system for providing constrained interactive walkthroughs inside large and complex virtual environments.

2. OVERVIEW OF THE MORPHABLE 3D-MOSAICS FRAMEWORK

As already mentioned, our system is capable of providing photorealistic walkthroughs of large-scale scenes at interactive frame rates. Assuming, for simplicity, that during the walkthrough the user motion takes place along a predefined path inside the environment (we note, of course, that our system can be readily extended to handle the case where the stereoscopic views have been captured not just along a predefined path, but throughout the scene), the input to our system is then a sparse set of stereoscopic views captured at certain locations (called *key-positions* hereafter) along that path (see Figure 1). Considering, initially, the case where only one view per key-position exists, a series of *local 3D models* is then constructed based on these stereoscopic views. There exists one model per view, and each local model is assumed to approximately capture the photometric and geometric properties of the scene only at a local level. Then, instead of trying to create a global 3D model out of all these local models (a task that can prove to be extremely difficult in many cases, requiring a very accurate registration between local models), we rather follow a different approach, which is that of creating a series of so-called *morphable 3D-models*. The key idea then is that, during the transition between any two successive key-positions (say pos_1, pos_2) along the path, one of these morphable 3D-models, say $\mathcal{L}_{1\to 2}$, is displayed by the rendering process (see Figure 1). At position pos_1, this model coincides with the local model $\mathcal{L}_1$ at that position, while, as we are approaching pos_2, it is gradually transformed into the next local model $\mathcal{L}_2$, coinciding with that upon reaching key-position pos_2. Therefore, during the rendering process, and as the user traverses the predefined path, a continuous morphing between successive local 3D models takes place all the time. It is important to note that this morphing between local models is both photometric, as well as geometric. Moreover, due to the way the morphable models are constructed, we ensure that the morphing always proceeds in a physically-valid manner, thus remaining transparent to the user of the system. For this purpose, algorithms for extracting 2D correspondences between wide-baseline images, as well as algorithms for establishing 3D correspondences between local geometric models, are proposed and implemented. These are used for estimating the photometric and geometric morphings, respectively.

Our system can be also extended to handle the existence of multiple stereoscopic views per key position, all of them related by a pure rotation of the stereoscopic camera. In this case, there will also be multiple local models per key-position. Therefore, before applying the morphing procedure, a so-called *3D-mosaic* per key-position needs to be constructed as well. Each 3D-mosaic will simply comprise the multiple local models at the corresponding key-position and, intuitively, will itself be a larger local model covering a wider field of view. Morphing can then proceed in the same way as before, with the only difference being that these 3D-mosaics will be the new local 3D models to be used during the stage of morphing (in place of the smaller individual ones). Hence, in this case, a *morphable 3D mosaic* (instead of a morphable 3D model) is said to be created by the algorithm.

Based on the above discussion, it follows that the 3D modeling pipeline of our framework consists of the stages that are listed below.

(i) *Local models construction*: this construction is accomplished based on the stereoscopic views that have been captured at key-positions along the path.

(ii) *Relative pose estimation*: a coarse estimation of the relative rotation and translation between successive local models takes place during this stage.

(iii) *Generation of morphable models*: this task amounts to estimating the photometric as well as geometric morphings between successive local models.

(iv) *3D mosaics generation*: during this stage, a 3D mosaic is generated at each key-position by aggregating all local models corresponding to that position.

During the next sections, each one of the above tasks will be described briefly, while the various algorithmic choices that have been made in each case will be justified and explained.

3. LOCAL 3D MODELS CONSTRUCTION

As already mentioned in the previous section, a set of local 3D models needs to be extracted during the first stage of the 3D modeling pipeline. There will be one local 3D model per stereoscopic view (and hence per key position as well, since we initially assume there is a one-to-one correspondence between key-positions and stereoscopic views). Such a 3D model is supposed to provide both a photometric as well as a geometric representation of the scene, but only at a local level (i.e., only near the corresponding key-position along the path). To this end, a stereo matching procedure is applied between the left $\mathcal{I}_{\text{left}}$ and right $\mathcal{I}_{\text{right}}$ images of the stereoscopic view, so that pixel correspondences between the 2 images are extracted [20]. Based on these correspondences (and assuming a calibrated pair of cameras), a 3D reconstruction can then take place via triangulation (i.e., by intersecting in 3D space the camera rays passing through corresponding pixels). In this manner, the 2D maps $\mathcal{X}_k$, $\mathcal{Y}_k$, and $\mathcal{Z}_k$ are produced that contain, respectively, the x, y, z coordinates of the reconstructed 3D points for each pixel in the left image (e.g., see the depth map in Figure 2). Without loss of generality, we will assume that these geometric maps are defined only over a region $\mathcal{R}_k$ on the image plane, and this region will be referred to as their *domain* hereafter.

These maps will form the local geometric representation of the scene, whereas the left image $\mathcal{I}_{\text{left}}$ will be used as the corresponding photometric representation $\mathcal{I}_k$ (i.e., $\mathcal{I}_k = \mathcal{I}_{\text{left}}$). The set $\mathcal{L}_k = \{\mathcal{X}_k, \mathcal{Y}_k, \mathcal{Z}_k, \mathcal{I}_k, \mathcal{R}_k\}$ will thus make up

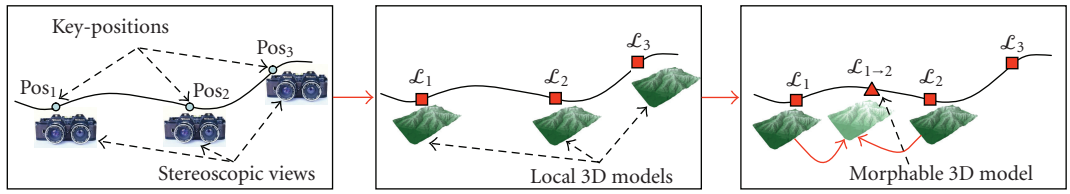


Figure 1: Initially, a sparse set of stereoscopic views is captured at key-positions along the path. Based on these stereoscopic views, a series of local 3D models is then constructed. As the user traverses the path, a morphable 3D model is displayed during rendering. In this manner, a continuous morphing between successive local models takes place at any time, with this morphing being both photometric as well as geometric.

Figure 2: A geometric depth map $\mathcal{Z}_k$ of a local model is shown on the left (this map is assumed to be defined only over region $\mathcal{R}_k$ on the image plane). By combining this geometric map with a photometric map, a local 3D model can be created, which can be used for providing rendered views of the scene around the key-position of the local model. One such view is shown here.

what we will hereafter refer to as a *local model* $\mathcal{L}_k$. In fact, this term will always implicitly refer to such a set of elements. As its name suggests, such a model can be used for reproducing a photorealistic reconstruction of the real scene, but only in a local region around the model's key-position pos_k (e.g., see the right image in Figure 2). In fact, it is only at this local region in 3D space where this model will be used during rendering, which is exactly why the geometric maps of a local model do not have to provide a very accurate representation of the whole scene's geometry.

For solving the stereo matching problem, we estimate a discrete disparity field by minimizing the energy of a 2nd-order Markov random field. The nodes of this MRF are the pixels of the left image, while the labels belong to a discrete set of disparities $\{0, 1, \ldots, d_{\max}\}$, where $d_{\max}$ represents the maximum allowed disparity. We then seek to assign a label d_p to each node p so that the following MRF energy is minimized:

$$E(\{d_p\}) = \sum_p V_p(d_p) + \sum_{(p,q)\in\aleph} V_{pq}(d_p, d_q), \qquad (1)$$

where the symbol $\aleph$ denotes a set of interacting pairs of pixels on the image grid (a 4-neighborhood system is assumed).

The single node potential $V_p(d_p)$ for assigning disparity d_p to pixel p is going to be set as $V_p(d_p) = w_p \cdot |\mathcal{I}_{\text{right}}(p - d_p) - \mathcal{I}_{\text{left}}(p)|^2$. The factor w_p expresses a confidence measure and is used for giving less weight to pixels that are less reliable for matching (e.g., their neighborhood in the image has uniform intensity). Its value is set based on the minimum eigenvalue of the following autocorrelation matrix G:

$$G = \begin{bmatrix} \sum \mathcal{I}_x^2 & \sum \mathcal{I}_x \mathcal{I}_y \\ \sum \mathcal{I}_x \mathcal{I}_y & \sum \mathcal{I}_y^2 \end{bmatrix}. \qquad (2)$$

Here, $(\mathcal{I}_x, \mathcal{I}_y)$ represents the left image gradient, while the sums are taken over a small window around the point p in the left image. The minimum eigenvalue is a measure of the "cornerness" of point p and will be large in regions with high texture variation, but will be small in uniform regions. Regarding the pairwise potentials $V_{pq}(\cdot, \cdot)$, these are set equal to the following truncated quadratic function: $V_{pq}(d_p, d_q) = \min(\mu, |d_p - d_q|^2)$, where μ denotes the maximum allowed discontinuity penalty that can be imposed. For optimizing the energy of the above discrete MRF, the recently proposed primal-dual algorithms of Komodakis et al. [21, 22] have been used, which can guarantee an approximately optimal solution for a very wide range of MRFs. For turning the resulting discrete disparity field into a continuous one (which will yield a local 3D model that is better for visualization purposes), a continuous optimization scheme (e.g., a simple gradient descent) can be applied afterwards as well.

4. RELATIVE POSE ESTIMATION BETWEEN SUCCESSIVE LOCAL MODELS

Given any two successive local models $\mathcal{L}_k$, $\mathcal{L}_{k+1}$ along the path, a rough estimation of their relative 3D poses (i.e., their 3D rotations and translations) needs to take place as well. Note that, similarly to the estimation of the models' geometry, only a very coarse and approximate estimation of the 3D pose is needed. Again, the reason is because this pose is not going to be used for performing an exact registration into a global 3D coordinate system, but merely for facilitating the morphing procedure that will take place later (i.e., during the next stage). One way for obtaining such a rough estimate is by, for example, computing a sparse set of approximate pixel correspondences between the photometric images $\mathcal{I}_k$, $\mathcal{I}_{k+1}$ of models $\mathcal{L}_k$, $\mathcal{L}_{k+1}$. Based on these correspondences, the so-called *fundamental matrix* (that relates points in images $\mathcal{I}_k$, $\mathcal{I}_{k+1}$) can be computed, from which one can easily extract an approximate estimation of the relative 3D pose [3].

Therefore, the pose estimation problem is reduced to that of extracting a sparse set of correspondences between $\mathcal{I}_k$, $\mathcal{I}_{k+1}$. A usual method for tackling the latter problem is the following: first a set of interest-points in $\mathcal{I}_k$ are extracted (using an interest-point detector). Then for each interest-point, say p, a set of candidate points CAND_p inside a large rectangular region SEARCH_p of $\mathcal{I}_{k+1}$ are examined and the best one is selected according to a similarity measure. Usually the candidate points are extracted by applying an interest-point detector to region SEARCH_p as well.

However, unlike left/right images of a stereoscopic view, $\mathcal{I}_k$ and $\mathcal{I}_{k+1}$ are separated by a wide baseline. Simple measures like correlation have been proved extremely inefficient in such cases. Assuming a smooth predefined path (and therefore a smooth change in orientation between $\mathcal{I}_k$, $\mathcal{I}_{k+1}$), it is safe to assume that the main difference at an object's appearance in images $\mathcal{I}_k$ and $\mathcal{I}_{k+1}$ comes from the forward camera motion along the Z axis (looming). The idea for extracting valid correspondences is then based on the following observation: the dominant effect of an object being closer to the camera in image $\mathcal{I}_{k+1}$ is that its image region in $\mathcal{I}_{k+1}$ appears scaled by a certain scale factor $s > 1$, that is, if $p \in \mathcal{I}_k$, $q \in \mathcal{I}_{k+1}$ are corresponding pixels: $\mathcal{I}_{k+1}(sq) \approx \mathcal{I}_k(p)$. So an image patch of $\mathcal{I}_k$ at p should look similar to an image patch of an appropriately rescaled (by s^{-1}) version of $\mathcal{I}_{k+1}$.

Of course, the scale factor s varies across the image. Therefore, the following strategy, for extracting reliable matches, can be applied.

(1) Quantize the scale space of s to a discrete set of values $S = \{s_j\}_{j=0}^{n}$, where $1 = s_0 < s_1 < \cdots < s_n$.

(2) Rescale $\mathcal{I}_{k+1}$ by the inverse scale s_j^{-1} for all $s_j \in S$ to get rescaled images $\mathcal{I}_{k+1}^{s_j}$. For any $q \in \mathcal{I}_{k+1}$, $p \in \mathcal{I}_k$, let us denote by $\text{PATCH}_{k+1}^{s_j}(q)$ a (small) fixed-size patch around the projection of q on $I_{k+1}s_j$ and by $\text{PATCH}_k(p)$ an equal-size patch of $\mathcal{I}_k$ at p.

(3) Given any point $p \in \mathcal{I}_k$ and its set of candidate points $\text{CAND}_p = \{q_i\}$ in $\mathcal{I}_{k+1}$, use correlation to find among the patches at any q_i and across any scale s_j, the one most similar to the patch of $\mathcal{I}_k$ at p:

$$(q', s') = \arg\max_{q_i, s_j} \text{corr}\left(\text{PATCH}_{k+1}^{s_j}(q_i), \text{PATCH}_k(p)\right). \quad (3)$$

This way, apart from a matching point $q' \in \mathcal{I}_{k+1}$, a scale estimate s' is provided for point p as well.

The above strategy has been proved very effective, giving a high percentage of exact matches even in cases with very large looming. Such an example can be seen in Figure 3.

5. ESTIMATION OF MORPHABLE 3D MODELS

The output of the previous stage will thus be a series of approximate local 3D models (along with approximate estimates of the relative pose between every successive two). Rather than trying to create a consistent global model by combining all local ones (a rather tedious task, requiring among others high-quality geometry and pose estimation), we will, instead, follow a different approach, which is based on the following observation. Let $\mathcal{L}_k = \{\mathcal{X}_k, \mathcal{Y}_k, \mathcal{Z}_k, \mathcal{I}_k, \mathcal{R}_k\}$, $\mathcal{L}_{k+1} = \{\mathcal{X}_{k+1}, \mathcal{Y}_{k+1}, \mathcal{Z}_{k+1}, \mathcal{I}_{k+1}, \mathcal{R}_{k+1}\}$ be any two successive local models along the path and let also pos_k, pos_{k+1} be their corresponding key-positions. Near position pos_k, the model $\mathcal{L}_k$ is ideal for representing the surrounding scene. However, as we move forward along the path and approach key-position pos_{k+1} of next model $\mathcal{L}_{k+1}$, the photometric and geometric properties of the environment are much better captured by that model. Compare, for example, the fine details of the rocks that are revealed in image $\mathcal{I}_{k+1}$ of Figure 4 (i.e., the photometric image of the next local model), but are not visible in image $\mathcal{I}_k$ of that figure (i.e., the photometric image of the previous local model). Hence, during the transition from pos_k to pos_{k+1}, we will try to gradually morph model $\mathcal{L}_k$ into a new *destination* model, which should coincide with $\mathcal{L}_{k+1}$ upon reaching position pos_{k+1}. (In fact, only part of this destination model can coincide with $\mathcal{L}_{k+1}$ since, in general, models $\mathcal{L}_k$, $\mathcal{L}_{k+1}$ will not represent exactly the same part of the scene.) This morphing should be geometric, as well as photometric (the latter wherever possible), and should proceed in a physically valid way. For this reason, we will use what we call a *morphable 3D-model*:

$$\mathcal{L}_{\text{morph}} = \mathcal{L}_k \cup \{\mathcal{X}_{\text{dst}}, \mathcal{Y}_{\text{dst}}, \mathcal{Z}_{\text{dst}}, \mathcal{I}_{\text{dst}}\}. \quad (4)$$

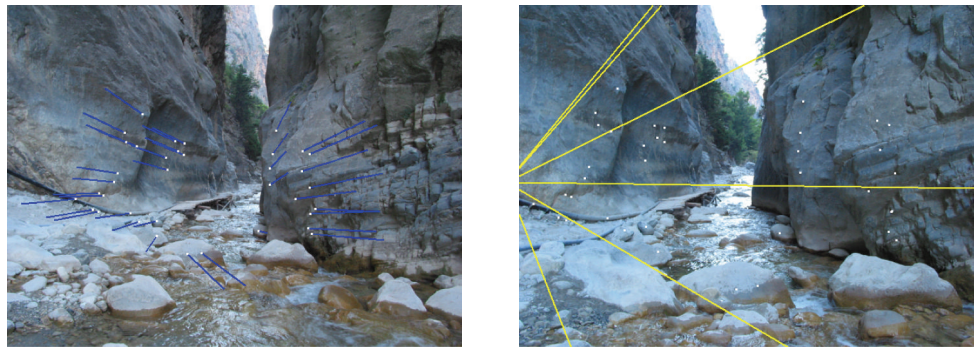
As can be seen, in addition to including all the geometric and photometric maps of model $\mathcal{L}_k$ (called the source maps hereafter), $\mathcal{L}_{\text{morph}}$ also consists of some additional destination maps: these are the geometric maps $\mathcal{X}_{\text{dst}}, \mathcal{Y}_{\text{dst}}, \mathcal{Z}_{\text{dst}}$, as well as the photometric map $\mathcal{I}_{\text{dst}}$, containing, respectively, the destination 3D vertices and colors for all points of the source model $\mathcal{L}_k$. Note that both the source and destination maps are defined over the same domain $\mathcal{R}_k$. Hence, at any time during the rendering process, the 3D coordinates and colors of the morphable model $\mathcal{L}_{\text{morph}}$ will be given by

$$\begin{bmatrix} \mathcal{X}_{\text{morph}}(p) \\ \mathcal{Y}_{\text{morph}}(p) \\ \mathcal{Z}_{\text{morph}}(p) \\ \mathcal{I}_{\text{morph}}(p) \end{bmatrix} = (1-\alpha) \cdot \begin{bmatrix} \mathcal{X}_k(p) \\ \mathcal{Y}_k(p) \\ \mathcal{Z}_k(p) \\ \mathcal{I}_k(p) \end{bmatrix} + \alpha \cdot \begin{bmatrix} \mathcal{X}_{\text{dst}}(p) \\ \mathcal{Y}_{\text{dst}}(p) \\ \mathcal{Z}_{\text{dst}}(p) \\ \mathcal{I}_{\text{dst}}(p) \end{bmatrix}, \quad \forall p \in \mathcal{R}_k, \quad (5)$$

where α is a parameter determining the amount of morphing (with $\alpha = 0$ at pos_k, $\alpha = 1$ at pos_{k+1} and $0 < \alpha < 1$ in between). Therefore, specifying $\mathcal{L}_{\text{morph}}$ simply amounts to filling-in the values of the destination maps $\{\mathcal{X}, \mathcal{Y}, \mathcal{Z}, \mathcal{I}\}_{\text{dst}}$ at each $p \in \mathcal{R}_k$.

For this purpose, a 2-step procedure will be followed, depending on whether a point in $\mathcal{L}_k$ has a physically corresponding point in $\mathcal{L}_{k+1}$ or not.

(1) The main idea is that if a point in $\mathcal{L}_k$ has a physical correspondence in $\mathcal{L}_{k+1}$, that correspondence can be directly used to fill the destination maps. Specifically, let Ψ denote the subset of region $\mathcal{R}_k$, consisting of all those p's for which there exists a physical correspondence in $\mathcal{L}_{k+1}$ (i.e., region Ψ refers to that part of the scene which is common to both models $\mathcal{L}_k$, $\mathcal{L}_{k+1}$). Let also $u_{k\to k+1}$ be a function that maps each $p \in \Psi \subset \mathcal{R}_k$ to its counterpart in R_{k+1}. Since model



(a) (b)

Figure 3: (a) Image $\mathcal{I}_k$ along with computed optical flow vectors (blue segments) for all points marked white. (b) Image $\mathcal{I}_{k+1}$ along with matching points (also marked white) for all marked points of (a). A few epipolar lines are also shown. 10 scales $S = \{1, 0.9^{-1}, \ldots, 0.1^{-1}\}$ have been used during matching.

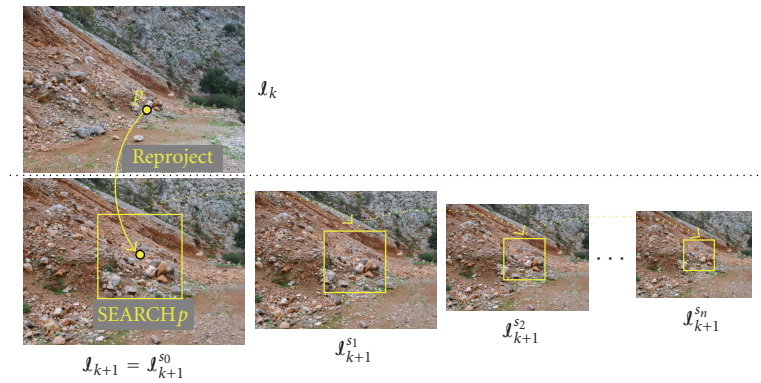


Figure 4: When seeking a correspondence for a pixel p of image $\mathcal{I}_k$, we reproject that pixel onto image $\mathcal{I}_{k+1}$ (using the approximate geometry of the local models and their relative pose) and then restrict our search only in a small region SEARCH$_p$ around the reprojection point. Furthermore, to take scale differences into account, we search not only inside region SEARCH$_p$ of image $\mathcal{I}_{k+1}$, but also inside the projection of SEARCH$_p$ onto the images $\mathcal{I}_{k+1}^s$, which are rescaled versions of image $\mathcal{I}_{k+1}$, that is, we search not only for a corresponding pixel, but also for a scale estimate. In fact, all the possible pixel-scale combinations will form the candidate labels of pixel p and to choose the best one of them, we will use an MRF optimization procedure.

$\mathcal{L}_k$ (after morphing) should coincide with $\mathcal{L}_{k+1}$, it must then hold as follows:

$$\begin{bmatrix} \mathcal{X}_{\text{dst}}(p) \\ \mathcal{Y}_{\text{dst}}(p) \\ \mathcal{Z}_{\text{dst}}(p) \\ \mathcal{I}_{\text{dst}}(p) \end{bmatrix} = \begin{bmatrix} \mathcal{X}_{k+1}(u_{k\to k+1}(p)) \\ \mathcal{Y}_{k+1}(u_{k\to k+1}(p)) \\ \mathcal{Z}_{k+1}(u_{k\to k+1}(p)) \\ \mathcal{I}_{k+1}(u_{k\to k+1}(p)) \end{bmatrix}, \quad \forall p \in \Psi. \tag{6}$$

This means that, for all $p \in \Psi$, the destination maps at p can be filled simply by copying the appropriate values from the maps of model $\mathcal{L}_{k+1}$. As a result, all points of $\mathcal{L}_k$ in region Ψ can be transformed both photometrically as well as geometrically.

(2) The rest of the points of $\mathcal{L}_k$ (i.e., all points defined over region $\overline{\Psi} = \mathcal{R}_k - \Psi$) do not have counterparts in model $\mathcal{L}_{k+1}$. Hence, no photometric morphing is possible to be applied to those points. As a result, these points will retain their color value from model $\mathcal{L}_k$, or

$$\mathcal{I}_{\text{dst}}(p) = \mathcal{I}_k(p), \quad \forall p \in \overline{\Psi}. \tag{7}$$

Nevertheless we still need to apply geometric morphing to them, so that no distortion/discontinuity in the 3D structure is observed during the transition from pos$_k$ to pos$_{k+1}$. This means that we will still have to fill-in the destination 3D coordinates (i.e., the destination geometric maps) for all $p \in \overline{\Psi}$.

The 2 important remaining issues (that also constitute the core of the morphing procedure) are as follows:

(i) how to compute the mapping $u_{k\to k+1}$. This is equivalent to estimating a 2D displacement field (also called optical flow field hereafter) between images $\mathcal{I}_k$ and $\mathcal{I}_{k+1}$,

(ii) and how to obtain the missing values of the destination geometric-maps for $p \in \overline{\Psi}$. These values will be needed for fully specifying the geometric morphing of model $\mathcal{L}_k$ over the whole domain $\mathcal{R}_k = \Psi \cup \overline{\Psi}$.

Both of these issues will be the subject of the two sections that are following.

5.1. Photometric morphing via optical flow estimation

In general, obtaining a reliable, dense optical flow field between images $\mathcal{I}_k$ and $\mathcal{I}_{k+1}$ is a particularly difficult problem. In fact, without additional input, usually only a sparse set of optical flow vectors can be obtained in the best case. The main difficulties stem from the fact that, unlike the left and right images of a stereo pair, images $\mathcal{I}_k$ and $\mathcal{I}_{k+1}$ are separated by a wide baseline (since they are captured at totally different positions pos_k, pos_{k+1}).

(i) On the one hand, this means that the resulting displacement vectors can be large, which, in turn, implies that, when searching for the correspondence of a pixel $p \in \mathcal{I}_k$, one needs to examine a very large region of image $\mathcal{I}_{k+1}$ (the search region corresponding to pixel p will be denoted by SEARCH_p hereafter). In this manner, however, the chance of an erroneous optical flow vector increases significantly (and so does the computational cost).

(ii) On the other hand, for extracting pixel correspondences, one usually needs to compare patches from images $\mathcal{I}_k$ and $\mathcal{I}_{k+1}$ (where the assumption is that patches associated with corresponding pixels should look similar to each other). In the case of wide-baseline images, however, simple similarity measures for comparing patches (such as the correlation between patches) will typically produce very inaccurate correspondences, when used naively. The reason is that, due to the wide baseline, the appearance of a patch may change significantly from one image to the next (e.g., the patch may have been rescaled and thus appear larger). Therefore, more appropriate similarity measures have to be used in this case.

(iii) Finally, even if both of the above problems are solved, optical flow estimation is inherently an ill-posed problem and so additional regularization assumptions need to be imposed.

For dealing with the first of these issues, we will make use of the underlying geometric maps $\mathcal{X}_k$, $\mathcal{Y}_k$, $\mathcal{Z}_k$ of model $\mathcal{L}_k$, as well as the relative pose between views $\mathcal{I}_k$ and $\mathcal{I}_{k+1}$. Theoretically, by using these quantities, we can find the correspondence of a pixel p of $\mathcal{I}_k$ by reprojecting it onto image $\mathcal{I}_{k+1}$. However, since all of the above quantities have been estimated approximately, this only allows us to narrow the search region SEARCH_p to a smaller region around the reprojection point, for example, see Figure 4 (in fact, the search region can be further restricted by taking its intersection with a small zone around the epipolar line corresponding to p). In addition, since we are interested in searching only for pixels of $\mathcal{I}_{k+1}$ that belong to region $\mathcal{R}_{k+1}$ (this is where $\mathcal{L}_{k+1}$ is defined), the final search region for p will be $\text{SEARCH}_p \cap \mathcal{R}_{k+1}$. If the resulting region is empty, then no optical flow vector will be estimated and pixel p will be considered as not belonging to region Ψ.

For dealing with the second problem, we will use a technique similar to the one described in Section 4 for getting a sparse set of correspondences. As already stated therein, the dominant effect due to a looming of the camera is that pixel neighborhoods in image $\mathcal{I}_{k+1}$ are scaled by a factor varying across the image. The solution proposed therein (and which we will also follow here) was to compare image patches of $\mathcal{I}_k$ with patches not only from $\mathcal{I}_{k+1}$ but also from rescaled versions of the latter image. We will again use a discrete set of scale factors $\mathcal{S} = \{1 = s_0 < s_1 < \cdots < s_n\}$ and $\mathcal{I}_{k+1}^s$ will denote image $\mathcal{I}_{k+1}$ rescaled by s^{-1}.

However, even after the above enhancements, estimating correspondences independently for each pixel will produce a lot of errors. This is because optical flow estimation is an inherently ill-posed problem. Hence, to obtain high-quality results, one needs to regularize this problem by taking advantage of the fact that optical flow vectors at neighboring pixels typically exhibit a high correlation. Discrete Markov random fields (MRFs) [23] are ideal candidates for modeling this type of local correlations and have also been used with great success in many other cases. For this reason, we will reduce optical flow estimation to a discrete MRF optimization task, that is to a discrete labeling problem. In this case, based on our earlier discussion, each label $l = (u_x, u_y, s) \in \mathbb{R}^2 \times \mathcal{S}$ will consist not only of an optical flow vector (u_x, u_y), but also of a scale factor $s \in \mathcal{S}$ (hereafter, the optical flow vector of any label l will be denoted by $u(l)$ and its scale by $s(l)$). Furthermore, the labels that can be assigned to a pixel $p \in \mathcal{I}_k$ will belong to the set $\text{LABELS}_p = \{p' - p : p' \in \text{SEARCH}_p\} \times \mathcal{S}$. This definition of set LABELS_p simply encodes the fact that, when seeking a correspondence for a pixel p of image $\mathcal{I}_k$, we are allowed to search across all scales $s \in \mathcal{S}$ (i.e., in all images I_{k+1}^s), but only inside region SEARCH_p at each scale (see Figure 4).

Given all these label sets, getting then an optical flow field is equivalent to picking one element from the cartesian product $\text{LABELS} = \prod_{p \in \Psi} \text{LABELS}_p$. In MRF optimization, this is typically done by trying to choose an element $\mathbf{l} = \{l_p\}_{p \in \Psi}$ that minimizes the MRF energy. This energy is defined as follows:

$$E(\{l_p\}) = \sum_{p \in \Psi} V_p(l_p) + \sum_{(p,q) \in \aleph} V_{p,q}(l_p, l_q).$$

Here, the functions $V_p(\cdot)$, $V_{pq}(\cdot, \cdot)$ denote, respectively, the so-called unary and pairwise MRF potentials, while $\aleph$ denotes all pairs of neighboring pixels in region Ψ. These two functions serve a different role in MRF optimization: the terms $V_p(l_p)$ try to penalize labelings $\{l_p\}$ that do not fit well to the observed data, whereas the terms $V_{pq}(l_p, l_q)$ try to enforce smoothness between labels l_p, l_q of neighboring pixels. In our case, each term $V_p(l_p)$ will measure the dissimilarity between corresponding patches according to label l_p. According to that label, a patch around $p \in \mathcal{I}_k$ will correspond to a patch around $p' = p + u(l_p) \in \mathcal{I}_{k+1}$ that has been rescaled by $s(l_p)$. Hence, to determine $V_p(l_p)$, pixel $p' \in \mathcal{I}_{k+1}$ is projected onto the rescaled image $\mathcal{I}_{k+1}^{s(l_p)}$ and a fixed size patch, denoted by $\text{PATCH}_{k+1}^{s(l_p)}(p')$, is taken around this projection. We then compare this patch with an equal-size patch $\text{PATCH}_k(p)$ around $p \in \mathcal{I}_k$, or

$$V_p(l_p) = \left|\left|\text{PATCH}_k(p) - \text{PATCH}_{k+1}^{s(l_p)}(p')\right|\right|_1, \qquad (8)$$

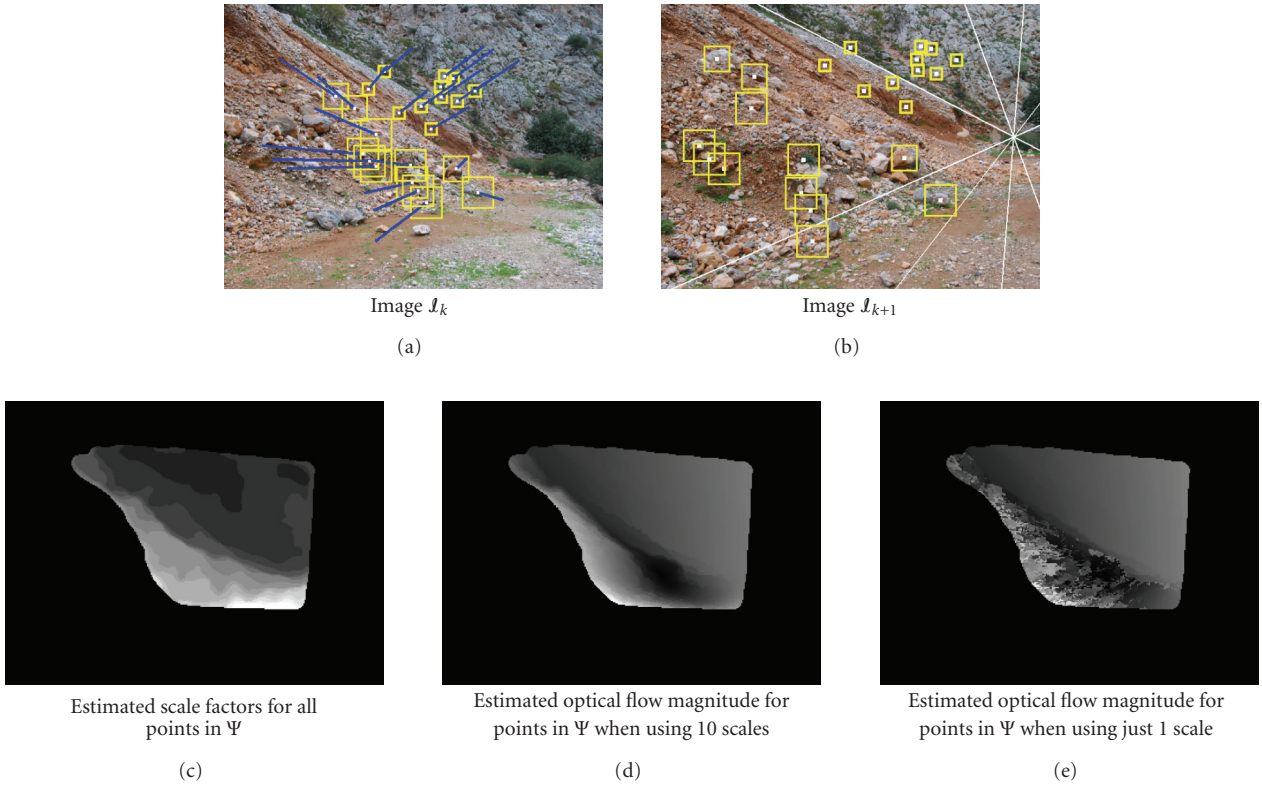


Image $\mathcal{I}_k$ (a)

Image $\mathcal{I}_{k+1}$ (b)

Estimated scale factors for all points in Ψ (c)

Estimated optical flow magnitude for points in Ψ when using 10 scales (d)

Estimated optical flow magnitude for points in Ψ when using just 1 scale (e)

FIGURE 5: On the top row, we show again the two images $\mathcal{I}_k$, $\mathcal{I}_{k+1}$ from Figure 4 along with computed optical flow vectors (blue segments) for a sparse set of points (a few epipolar lines are also shown in $\mathcal{I}_{k+1}$). In both of these images, the yellow square around a point is proportional to the point's estimated scale factor (10 scales $S = \{1, 0.9^{-1}, \ldots, 0.1^{-1}\}$ have been used). On the bottom row, we show the estimated scale factors for all the points in region Ψ, as well as the corresponding optical flow magnitudes at these points. Regarding optical flow, two results are shown: when using 10 possible scales $S = \{1, 0.9^{-1}, \ldots, 0.1^{-1}\}$, as well as when using just one scale $S = \{1\}$. As expected, in the latter case, the estimated flow is quite noisy, since the algorithm fails to produce exact optical flow for points that actually undergo a large change of scale. We note that darker pixels (in a grayscale image) correspond to smaller values.

where $\|\cdot\|_1$ denotes the l_1-norm and thus $V_p(l_p)$ measures the sum of absolute intensity differences between corresponding patches.

On the other hand, as already mentioned, the role of the pairwise terms $V_{pq}(\cdot,\cdot)$ is to regularize the solution by imposing smoothness on the resulting optical flow vectors. These terms can be defined using the following truncated distances:

$$V_{pq}(l_p, l_q) = \lambda_0 \min\left(||u(l_p) - u(l_q)||, \mu_0\right) + \lambda_1 \min\left(||s(l_p) - s(l_q)||, \mu_1\right). \quad (9)$$

As can be seen, these terms penalize labelings that assign discontinuous displacements or discontinuous scale factors to neighboring pixels (note that, due to the truncation by μ_0 and μ_1, some discontinuities are still allowed to exist, e.g., at the boundary between 2 objects). Minimizing the resulting MRF energy is in general NP-hard. Yet, there exist optimization techniques, which can provably guarantee approximately optimal solutions [21, 22, 24]. Foe example, Figure 5 contains the resulting optical flow for images $\mathcal{I}_k$, $\mathcal{I}_{k+1}$ of Figure 4. For comparison, we also show the corresponding optical flow when no search across scale takes place, that is $\mathcal{S} = \{1\}$. As expected, in this case, the optical flow vectors are quite noisy in regions undergoing a large change of scale due to the camera motion. When using the fast optimization algorithms from [21], convergence is very fast (only a few iterations are needed) and the average running time per iteration is on the order of a few seconds.

5.2. Geometric morphing

After estimating function $u_{k\to k+1}$, we may apply (6) to all points in region Ψ. This allows us to copy values from the geometric maps of model $\mathcal{L}_{k+1}$ in order to fill-in region Ψ of the destination geometric maps $\mathcal{X}_{\text{dst}}$, $\mathcal{Y}_{\text{dst}}$, $\mathcal{Z}_{\text{dst}}$. For notational convenience, the resulting partially complete geometric maps, that is with known values only in region Ψ, will be denoted by $\widehat{\mathcal{X}}_{\text{dst}}$, $\widehat{\mathcal{Y}}_{\text{dst}}$, $\widehat{\mathcal{Z}}_{\text{dst}}$ (e.g., see the middle image in Figure 6). To completely specify morphing, it only remains to fill-in the destination geometric maps over the rest of their domain, that is, over region $\overline{\Psi} = \mathcal{R}_k - \Psi$. For this, we need to specify the corresponding destination 3D vertices for all points of $\mathcal{L}_k$ in $\overline{\Psi}$. Since these points do not have a physically corresponding point in $\mathcal{L}_{k+1}$, we cannot apply (6) to get a destination 3D coordinate from model $\mathcal{L}_{k+1}$. A naive solution to deal with this would be to simply apply

no geometric morphing to these points, that is to allow the destination 3D vertices of these points to coincide with their source vertices in $\mathcal{L}_k$. In that case, we would end up with destination geometric maps:

(i) whose values in Ψ would have been copied from model $\mathcal{L}_{k+1}$,

(ii) but whose values in $\overline{\Psi}$ would have been copied from model $\mathcal{L}_k$.

In this manner, however, the resulting destination maps $\mathcal{X}_{\text{dst}}$, $\mathcal{Y}_{\text{dst}}$, $\mathcal{Z}_{\text{dst}}$ would contain discontinuities along the common boundary (say $\partial\overline{\Psi}$) of regions Ψ and $\overline{\Psi}$ (e.g., see the left image in Figure 6), thus leading to very annoying discontinuity artifacts (holes) during the morphing procedure (e.g., see the left image in Figure 7). This would indeed be the case, since the geometries of both $\mathcal{L}_k$ and $\mathcal{L}_{k+1}$ (as well as their relative poses) have been estimated only approximately and thus these models may not match perfectly to each other.

Instead, the proper way to fill-in the destination vertices at region $\overline{\Psi}$ is based on the observation that a physically valid destination 3D model should satisfy the following 2 conditions.

(1) Along the boundary of $\overline{\Psi}$, no discontinuity in its 3D structure should exist, that is the values of $\mathcal{X}_{\text{dst}}$, $\mathcal{Y}_{\text{dst}}$, $\mathcal{Z}_{\text{dst}}$ at $\partial\overline{\Psi}$ should coincide with the already filled-in values of $\widehat{\mathcal{X}}_{\text{dst}}$, $\widehat{\mathcal{Y}}_{\text{dst}}$, $\widehat{\mathcal{Z}}_{\text{dst}}$ at $\partial\overline{\Psi}$.

(2) In the interior of $\overline{\Psi}$, the relative 3D structure of the destination model should coincide with the relative 3D structure of the source model $\mathcal{L}_k$ (since these two models are assumed to be related by a rigid transformation). Put otherwise, the relative 3D structure of $\mathcal{L}_k$ should be preserved during morphing, or, at least, it should not be distorted too much.

In mathematical terms, the first condition obviously translates to

$$\begin{aligned} \mathcal{X}_{\text{dst}|\partial\overline{\Psi}} &= \widehat{\mathcal{X}}_{\text{dst}|\partial\overline{\Psi}}, \\ \mathcal{Y}_{\text{dst}|\partial\overline{\Psi}} &= \widehat{\mathcal{Y}}_{\text{dst}|\partial\overline{\Psi}}, \\ \mathcal{Z}_{\text{dst}|\partial\overline{\Psi}} &= \widehat{\mathcal{Z}}_{\text{dst}|\partial\overline{\Psi}}, \end{aligned} \tag{10}$$

whereas the second condition, that is that of preserving the relative 3D structure of $\mathcal{L}_k$ during morphing, implies

$$\begin{bmatrix} \mathcal{X}_{\text{dst}}(p) - \mathcal{X}_{\text{dst}}(q) \\ \mathcal{Y}_{\text{dst}}(p) - \mathcal{Y}_{\text{dst}}(q) \\ \mathcal{Z}_{\text{dst}}(p) - \mathcal{Z}_{\text{dst}}(q) \end{bmatrix} = \begin{bmatrix} \mathcal{X}_k(p) - \mathcal{X}_k(q) \\ \mathcal{Y}_k(p) - \mathcal{Y}_k(q) \\ \mathcal{Z}_k(p) - \mathcal{Z}_k(q) \end{bmatrix}, \quad \forall p, q \in \overline{\Psi} \tag{11}$$

which is easily seen to be equivalent to

$$\begin{aligned} &[\nabla\mathcal{X}_{\text{dst}}(p) \;\; \nabla\mathcal{Y}_{\text{dst}}(p) \;\; \nabla\mathcal{Z}_{\text{dst}}(p)] \\ &\quad = [\nabla\mathcal{X}_k(p) \;\; \nabla\mathcal{Y}_k(p) \;\; \nabla\mathcal{Z}_k(p)], \quad \forall p \in \overline{\Psi}. \end{aligned} \tag{12}$$

Therefore, based on the above conditions, the destination vertices can be extracted by independently solving 3 similar optimization problems (one problem per 3D coordinate). For instance, the missing values of $\mathcal{Z}_{\text{dst}}$ can be estimated by solving the following optimization problem:

$$\min_{\mathcal{Z}_{\text{dst}}} \iint_{\overline{\Psi}} ||\nabla\mathcal{Z}_{\text{dst}}(p) - \nabla\mathcal{Z}_k(p)||^2 dp, \quad \text{s.t. } \mathcal{Z}_{\text{dst}|\partial\overline{\Psi}} = \widehat{\mathcal{Z}}_{\text{dst}|\partial\overline{\Psi}}. \tag{13}$$

The finite-difference discretization of (13) (using the underlying pixel grid) yields a quadratic optimization problem, which can be reduced to a sparse (banded) system of linear equations, that can then be solved easily. Alternatively, one needs to observe that a function minimizing (13) is also a solution to the following Poisson equation with Dirichlet boundary conditions [25, 26]:

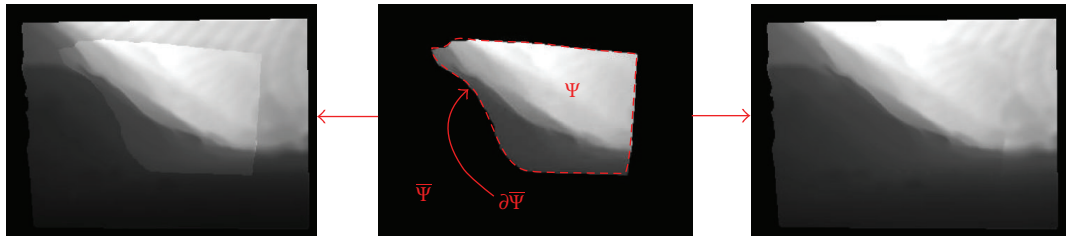
$$\Delta\mathcal{Z}_{\text{dst}} = \text{div}(\nabla\mathcal{Z}_k), \quad \text{subject to } \mathcal{Z}_{\text{dst}|\partial\overline{\Psi}} = \widehat{\mathcal{Z}}_{\text{dst}|\partial\overline{\Psi}}. \tag{14}$$

Therefore, in this case, the missing values of the destination geometric maps can be filled by independently solving 3 Poisson equations of the above type (note that these equations can be solved very efficiently, e.g., the needed time is at most only a few seconds). See the right image in Figure 6 for a destination depth map produced with this method. Also, the middle image of Figure 7 contains a corresponding rendering.

6. RENDERING PIPELINE

An important advantage of our framework is that, regardless of the scene's size, only one "morphable 3D-model" $\mathcal{L}_{\text{morph}}$ needs to be displayed at any time during rendering, that is, the rendering pipeline has to execute the geometric and photometric morphings for only one local model $\mathcal{L}_k$ (as described in Section 5). This makes our system extremely scalable to large-scale scenes. In addition to that, by utilizing the enhanced capabilities of modern 3D graphics hardware, both types of morphing can admit a GPU implementation, thus making our system ideal for 3D acceleration and capable of achieving very high frame rates during rendering.

More specifically, for implementing the photometric morphing of model $\mathcal{L}_k$, *multitexturing* needs to be employed as a first step. To this end, both images $\mathcal{I}_k$, $\mathcal{I}_{k+1}$ will be used as textures and each 3D vertex whose corresponding 2D point $p \in \mathcal{I}_k$ is located inside region Ψ will be assigned 2 pairs of texture coordinates: the first pair will coincide with the image coordinates of point $p \in \mathcal{I}_k$ while the second one will be equal to the image coordinates of the corresponding point $u_{k\to k+1}(p) \in \mathcal{I}_{k+1}$ (see (6)). Then, given these texture coordinates, a so-called *pixel-shader* (along with its associated *vertex-shader*) [27] can simply blend the two textures in order to implement (on the GPU) the photometric morphing defined by (5). Pixel and vertex shaders are user defined scripts that are executed by the GPU for each incoming 3D vertex and output pixel, respectively. One possible implementation of such scripts, for the case of photometric morphing, is shown on the left side of Figure 8 where, for this specific example, the OpenGL Shading Language (GLSL) [27] has been used for describing the shaders. As for the 3D vertices which are associated to points located inside region $\overline{\Psi}$, the situation is even simpler



Full depth map that has been filled-in without using geometric morphing

Incomplete depth map (only region Ψ has been filled)

Full depth map that has been filled-in using geometric morphing

FIGURE 6: In the middle, we show an incomplete destination depth map $\hat{\mathcal{Z}}_{\text{dst}}$, whose region Ψ has been filled-in by using the optical flow from Figure 5 and applying (6). On the left, we show the full destination depth map that has been produced by extending $\hat{\mathcal{Z}}_{\text{dst}}$ to region $\overline{\Psi} = \mathcal{R}_k - \Psi$ without using geometric morphing. Notice the discontinuities existing along the boundary $\partial\overline{\Psi}$. On the right, we show the full destination depth map that has been produced when geometric morphing has been enabled. As can be seen, no discontinuities exist along the boundary $\partial\overline{\Psi}$ in this case.

Morphable model without geometric morphing

(a)

Morphable model with geometric morphing enabled

(b)

Zoomed version of the image in the middle

(c)

FIGURE 7: We show rendered views of a morphable 3D-model during the transition from one key-position (corresponding to image $\mathcal{I}_k$ in Figure 4) to another key-position (corresponding to image $\mathcal{I}_{k+1}$ in Figure 4). On (a) we show the result produced when no geometric morphing is APPLIED to points in $\overline{\Psi}$ (notice the resulting discontinuity, i.e., hole). On (b) we render the same view, but this time we also apply geometric morphing to points in $\overline{\Psi}$ (no discontinuities/holes exist in this case). A zoomed version of this view is also shown (c). Although, as mentioned, there is no geometric discontinuity, yet there is a difference in texture resolution between the left part of the image (points in $\overline{\Psi}$) and the right part (points in Ψ), since only points of the latter part are morphed photometrically.

since no actual photometric morphing takes place in there (see (3)) and so only image $\mathcal{I}_k$ needs to be texture-mapped onto these vertices.

On the other hand, for implementing the geometric morphing, the following procedure is used: two 2D triangulations of regions Ψ, $\overline{\Psi}$ are first generated resulting into two 2D triangle meshes TRI_{Ψ}, $\text{TRI}_{\overline{\Psi}}$. Based on these triangulations and the underlying geometric maps of $\mathcal{L}_k$, two 3D triangle meshes MESH_{Ψ}^{k}, $\text{MESH}_{\overline{\Psi}}^{k}$ are constructed. Similarly, using TRI_{Ψ}, $\text{TRI}_{\overline{\Psi}}$ and the destination geometric maps $\mathcal{X}_{\text{dst}}$, $\mathcal{Y}_{\text{dst}}$, $\mathcal{Z}_{\text{dst}}$, two more 3D triangle meshes $\text{MESH}_{\Psi}^{\text{dst}}$, $\text{MESH}_{\overline{\Psi}}^{\text{dst}}$ are constructed as well. It is then obvious that geometric morphing (as defined by (5)) amounts to a simple *vertex blending* operation, that is, meshes MESH_{Ψ}^{k}, $\text{MESH}_{\overline{\Psi}}^{k}$ are weighted by $1-\alpha$, meshes $\text{MESH}_{\Psi}^{\text{dst}}$, $\text{MESH}_{\overline{\Psi}}^{\text{dst}}$ are weighted by α, and the resulting weighted vertices are then added together. Vertex blending, however, is an operation that is directly supported by all modern GPUs and, as an example, Figure 8 (right box) contains skeleton code in C showing how one can implement vertex blending using the OpenGL standard.

Therefore, based on the above observations, rendering a morphable model simply amounts to feeding into the GPU just 4 textured triangle meshes. This is, however, a rendering path which is highly optimized in all modern GPUs and, therefore, a considerable amount of 3D acceleration can be achieved in this manner during the rendering process.

6.1. Decimation of local 3D models

Up to now, we have assumed that a full local 3D model is constructed each time, that is, all points of the image grid are included as vertices in the 2D triangulations TRI_{Ψ}, $\text{TRI}_{\overline{\Psi}}$. However, we can also use simplified versions of these 2D triangle meshes, provided, of course, that these simplified meshes approximate well the underlying geometric maps [28]. In fact, due to our framework's structure, a great amount of simplification can be achieved and the reason is that a simplified model $\mathcal{L}_k'$ has to be a good approximation to the full local model $\mathcal{L}_k$ only in the vicinity of pos_k (remember that model $\mathcal{L}_k$ is being used only in a local region

```
// Vertex shader
void main()
{
  // set texture coordinates for multitexturing
  gl_TexCoord[0] = gl_MultiTexCoord0;
  gl_TexCoord[1] = gl_MultiTexCoord1;

  gl_Position = ftransform();
}

// Pixel shader
uniform float α; // the amount of morphing
uniform sampler2D tex0; //texture of image I_k
uniform smpaler2D tex1; //texture of image I_{k+1}

void main()
{
  vec2 st0 = texture2D(tex0,gl_TexCoord[0].st);
  vec2 st1 = texture2D(tex1,gl_TexCoord[1].st);
  gl_FragColor = (1-α)*st0+α*st1;
}
```

(a)

```
...
// enable vertex blending with 2 weights
glEnable(GL_VERTEX_BLEND_ARB);
glVertexBlendARB(2);
...

// set 1st blending weight for MESH_Ψ^k, MESH_Ψ^k
glWeightfvARB (1 − α);

// you can now render MESH_Ψ^k, MESH_Ψ^k
glMatrixMode (GL_MODELVIEW0_ARB);
...

// set 2nd blending weight for MESH_Ψ^dst, MESH_Ψ^dst
glWeightfvARB (α);

// you can now render MESH_Ψ^dst, MESH_Ψ^dst
glMatrixMode (GL_MODELVIEW1_ARB);
...
```

(b)

Figure 8: (a): Pixel shader code (and the associated vertex shader code), written in GLSL (OpenGL Shading Language), for implementing the photometric morphing. (b): Skeleton code in C for applying vertex blending in OpenGL.

around pos_k). Based on this observation, the following iterative procedure is being used for the simplification of the 2D meshes: at the start of each iteration, there exists a current 2D Delaunay triangulation TRI_i which has as vertices only a subset of the points on the image grid. Based on TRI_i, an error function $e(p)$ is defined over the image grid which is measuring how well the current MESH_i approximates the underlying geometric maps (here MESH_i denotes the 3D surface defined by TRI_i). To each triangle, say T, of TRI_i we then associate the following two quantities: $e(T) = \max_{p\in T} e(p)$ (i.e., the maximum geometric error across T) and $p(T) = \arg\max_{p\in T} e(p)$ (i.e., the interior point of T achieving this maximum error). At each iteration the triangle $T_{\max} = \arg\max_{T\in \text{TRI}_i} e(T)$ of maximum error is selected and its point $p(T_{\max})$ is added as a new vertex in the triangulation. This way a new Delaunay triangulation TRI_{i+1} is given as input to the next iteration of the algorithm and the process repeats until the maximum error $\max_{T\in \text{TRI}_i} e(T)$ falls below a user specified threshold $e_{\max}$, which basically controls the total amount of simplification to be applied to the local model. Our algorithm is initialized with a sparse Delaunay triangulation TRI_0 and the only restriction imposed on TRI_0 is that it should contain the edges along the boundary between regions Ψ and $\overline{\Psi}$ (i.e., a constrained Delaunay triangulation has to be used) so that there are no cracks at the boundary of the corresponding meshes.

For completely specifying the decimation process, all that remains to be defined is the error function $e(p)$. One option would be to set $e(p) = \|\text{DEV}(p)\|$, where $\text{DEV}(p) = \|\text{MESH}_i(p) - [\mathcal{X}_k(p)\ \ \mathcal{Y}_k(p)\ \ \mathcal{Z}_k(p)\,]\|$ denotes the geometric deviation at p between MESH_i and the underlying geometric maps ($\text{MESH}_i(p)$ is the 3D point defined by MESH_i at p). This, however, would not result in the greatest possible geometric decimation. The key observation for managing to increase the geometric simplification of MESH_i, relies on the fact that MESH_i needs to provide a good approximation of $\mathcal{L}_k$ only in a local region between positions pos_k and pos_{k+1} of the path. As a result, we can choose to relate $e(p)$ to the maximum projection error only at these locations. More specifically, we set

$$e(p) = \max\left(\text{PROJ\_ERR}_{\text{pos}_k}, \text{PROJ\_ERR}_{\text{pos}_{k+1}}\right), \tag{15}$$

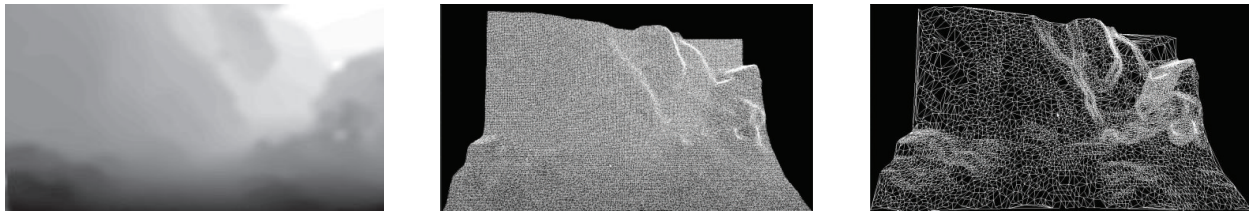
where $\text{PROJ\_ERR}_{\text{pos}_k}$ and $\text{PROJ\_ERR}_{\text{pos}_{k+1}}$ denote the maximum projection error at positions pos_k and pos_{k+1} respectively, that is,

$$\begin{aligned} \text{PROJ\_ERR}_{\text{pos}_k} &= \max_{p\in I_k} \left\|\text{PROJ}_{\text{pos}_k}(\text{DEV}(p))\right\|, \\ \text{PROJ\_ERR}_{\text{pos}_{k+1}} &= \max_{p\in I_{k+1}} \left\|\text{PROJ}_{\text{pos}_{k+1}}(\text{DEV}(p))\right\|. \end{aligned} \tag{16}$$

In practice, this definition of the error $e(p)$ has managed to achieve much larger reductions in the geometric complexity of the local 3D models, without sacrificing the visual quality of the resulting renderings, thus giving excellent results. Furthermore, the user-defined threshold $e_{\max}$ can now be expressed in pixel units and can thus be set in a far more intuitive way by the user. An example of a simplified local model that has been produced in this manner is shown in Figure 9, in which case $e_{\max}$ has been set equal to 0.5 pixels. We should finally note that by using the simplified local 3D models one can reduce the rendering time even further, thus achieving even higher frame rates, for example, over 50 fps.

7. 3D-MOSAICS CONSTRUCTION

For simplicity, up to this point we have been assuming that, during the image acquisition process, only one stereoscopic



(a) (b) (c)

Figure 9: (a) Estimated disparity field corresponding to a local 3D model $\mathcal{L}_k$. (b) Resulting full 3D model produced when a nondecimated 2D triangulation of the geometric maps has been used. (c) Simplified 3D model of $\mathcal{L}_k$ produced using a decimated 2D triangulation where the $e_{\max}$ threshold has been set equal to 0.5 pixels.

image has been captured per key-position along the path. Our framework, however, can be readily extended to the case where multiple stereoscopic views (and hence multiple local models) exist per key-position, related to each other by a pure rotation of the stereoscopic camera. Such a scenario can be useful, for example, when an extended field of view is required during the virtual walkthrough (as in VR installations with large screens). To reduce this case to the one already examined, it suffices that a single local model (called *3D-mosaic* hereafter) is constructed at each key-position. This can be done by assembling all individual local models at that position, which, as we will see, is an easy task, since these models differ only by a rotation. Then, at any time during the rendering process, instead of morphing the individual local models, a morphing between successive 3D-mosaics should take place, as before. For this reason, the term *morphable 3D-mosaic* is used in this case.

Intuitively, each 3D-mosaic should correspond to a local model produced from a stereoscopic camera with a wider field of view. Technically, let $\mathcal{L}_k^i = \{\mathcal{X}_k^i, \mathcal{Y}_k^i, \mathcal{Z}_k^i, \mathcal{I}_k^i, \mathcal{R}_k^i\}$, with $i \in \{1, \ldots, n\}$, be the individual local models at the kth key-position. Then, constructing a 3D mosaic $\mathcal{L}_{\text{mos}} = \{\mathcal{X}_{\text{mos}}, \mathcal{Y}_{\text{mos}}, \mathcal{Z}_{\text{mos}}, \mathcal{I}_{\text{mos}}, \mathcal{R}_{\text{mos}}\}$ corresponds to filling-in the geometric and photometric maps of $\mathcal{L}_{\text{mos}}$ based on all maps from $\{\mathcal{L}_k^i\}_i$. To this end, the rotation R_{ij} between each pair of local models $\mathcal{L}_k^i$, $\mathcal{L}_k^j$ needs to be estimated first. This can be done very accurately, since no depth information is required for this task. Essentially, estimating R_{ij} is equivalent to estimating a 2D projective transformation H_{ij} (the infinite homography induced by the plane at infinity [3]) relating pixels in photometric images $\mathcal{I}_k^i$, $\mathcal{I}_k^j$ (and for this, just a sparse set of pixel correspondences between $\mathcal{I}_k^i$, $\mathcal{I}_k^j$ needs to be established).

The estimated transformations H_{ij} (for all i, j) can then be used for aligning the maps of all local models $\{\mathcal{L}_k^i\}_i$ into a common image plane. Furthermore, these aligned maps can be fused to produce the maps of model $\mathcal{L}_{\text{mos}}$. Although this suffices for the construction of the photometric map $\mathcal{I}_{\text{mos}}$, it turns out that, before fusing the geometric maps, for example, $\mathcal{Z}_k^i$ to produce $\mathcal{Z}_{\text{mos}}$, an additional rectification step needs to be applied to all these maps $\mathcal{Z}_k^i$, so that they become geometrically consistent with each other (recall that the geometry of each $\mathcal{L}_k^i$ has been estimated only approximately). To this end, an operator RECTIFY($\mathcal{Z}_k^i \rightarrow \mathcal{Z}_k^j$) is defined, whose role is to modify $\mathcal{Z}_k^i$ so that it coincides with $\mathcal{Z}_k^j$ at the intersection $\mathcal{R}_k^i \cap \mathcal{R}_k^j$ (this ensures geometric consistency between $\mathcal{Z}_k^i$, $\mathcal{Z}_k^j$), but without the relative 3D structure of $\mathcal{Z}_k^i$ in region $\mathcal{R}_k^i - \mathcal{R}_k^j$ being distorted too much. This operator RECTIFY($\mathcal{Z}_k^i \rightarrow \mathcal{Z}_k^j$) can be easily seen to reduce to solving the following optimization problem:

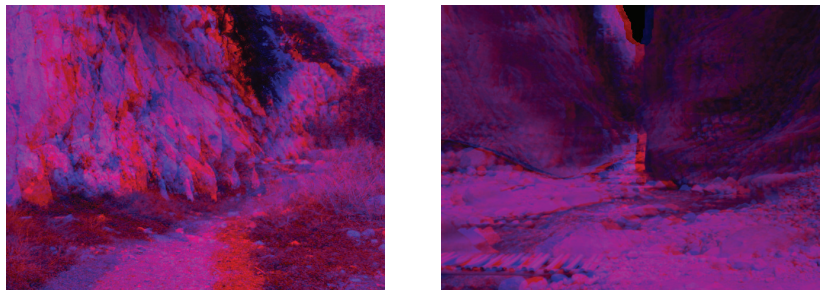
$$\mathcal{Z}_k^i = \arg\min_{\mathcal{Z}} \iint_{\mathcal{R}_k^i} \left\| \nabla \mathcal{Z}(p) - \nabla \mathcal{Z}_k^i(p) \right\|^2 dp, \quad \text{s.t. } \mathcal{Z}_{|\mathcal{R}_k^i \cap \mathcal{R}_k^j} = \mathcal{Z}_{k|\mathcal{R}_k^i \cap \mathcal{R}_k^j}^j. \tag{17}$$

This problem is similar to the problem (13) studied in the previous section, and can thus be reduced to a Poisson partial differential equation. Obviously, geometric consistency is ensured simply by applying operator RECTIFY($\cdot$) to all pairs $\mathcal{Z}_k^i$, $\mathcal{Z}_k^j$ with $i < j$, that is, by solving a set of Poisson PDEs. See Figure 10 for some results produced with this method.

As already mentioned, an important property of our geometric rectification algorithm is that the true relative 3D structure is preserved in the resulting 3D mosaic. This is in deep contrast to other methods for fusing geometric maps, such as feathering. This is illustrated with the following toy example. Let us assume that we have 2 local models, $\mathcal{L}_{\text{left}}$, $\mathcal{L}_{\text{right}}$, that must be fused to produce a 3D mosaic. The true geometry of the 3D mosaic should consist of a planar object at a constant depth $\mathcal{Z}_{\text{true}}$. Due to noisy observations, however, the left half of this planar object (corresponding to local model $\mathcal{L}_{\text{left}}$) was estimated to have depth $\mathcal{Z}_{\text{left}} = \mathcal{Z}_{\text{true}} - \epsilon_0$, while the right half of the object (corresponding to local model $\mathcal{L}_{\text{right}}$) was estimated to have depth $\mathcal{Z}_{\text{right}} = \mathcal{Z}_{\text{true}} + \epsilon_1$. It is easy to verify that, no matter what the values of ϵ_0, ϵ_1 are, the 3D mosaic produced by our fusion algorithm will always be a planar object with constant depth. On the other hand, when using a feathering-like method, that is, when a weighted linear combination of $\mathcal{Z}_{\text{left}}$, $\mathcal{Z}_{\text{right}}$ is used for creating the mosaic's depth map, this depth map will exhibit a high distortion at a zone around the middle of the object, that is, around the common boundary of $\mathcal{L}_{\text{left}}$ and $\mathcal{L}_{\text{right}}$. Inside this zone, the resulting 3D mosaic will have depths that vary from $\mathcal{Z}_{\text{true}} - \epsilon_0$ to $\mathcal{Z}_{\text{true}} + \epsilon_1$, and will thus appear highly distorted to the human eye, when visualized. Not only that, but the higher the error (i.e., the values of ϵ_0, ϵ_1), the higher the distortion will be. In fact, since this error is typically proportional to how far the object is from the

(a) (b)

Figure 10: On (a) we show a portion of a 3D-mosaic constructed by combining two local models. We show the result both when no geometric rectification is applied to the local models and also when we do apply a geometric rectification. In the former case, notice the geometric inconsistency at the boundary between the two local models. On (b) we show another 3D-mosaic that has been constructed by combining three local models.



(a) (b)

Figure 11: Two stereoscopic views as would be rendered by the VR system (for illustration purposes these are shown in the form of red-blue images).

camera, this value can be really high for large natural scenes, as is the distortion in this case.

8. VISUAL RECONSTRUCTION OF THE SAMARIA GORGE

The "morphable 3D-mosaics" framework has been successfully applied to the visual 3D reconstruction of the well-known Samaria Gorge in Crete (a gorge which was recently awarded by the Council of Europe with a Diploma First Class, as being one of Europe's most beautiful spots). Based on this 3D reconstruction, and by also using a 3D virtual reality installation, the ultimate goal of that work has been to provide a lifelike virtual tour of the Samaria Gorge to all visitors of the National History Museum of Crete, located in the city of Heraklion. To this end, the most beautiful spots along the gorge have been selected and for each such spot a predefined path, that was over 100 meters long, was chosen as well. About 15 key-positions have been selected along each path and approximately 45 stereoscopic views have been acquired at these positions, with 3 stereoscopic views per position (in this manner, a field of view, that was approximately 120^{o} wide, has been covered). Using the reconstructed morphable 3D-mosaics, a photorealistic walkthrough of the Samaria Gorge has been obtained, which was visualized at interactive frame rates by means of a virtual reality system. The hardware equipment that has been used for the virtual reality system was a common PC (with a Pentium 4 2,4 GHz CPU on it), which was connected to a single-channel stereoscopic projection system from Barco. The projection system was consisting of a pair of circular polarized LCD projectors (Barco Gemini), an active-to-passive stereo converter, as well as a projection screen, while the rendering was done just on a single GeForce 6800 3D graphics card (installed on the PC). For the stereoscopic effect to take place, 2 views (corresponding to the left and right eyes) were rendered by the graphics card at any time (e.g., see Figure 11) and so the museum visitors were able to participate in the virtual tour simply by wearing stereo glasses that were matched to the circular polarization of the projectors.

Despite the fact that a common graphics card has been used, very high frame rates, of about 25 fps in stereo mode (i.e., 50 fps in monomode), were obtained. On the one hand, this is due to the fact that, regardless of the actual size of the scene, only one morphable 3D-model needs to be displayed at any time during rendering. This is an important advantage, which makes our framework extremely scalable to large-scale scenes. On the other hand, the achieved frame rates are also due to the highly optimized rendering pipeline of our system. The reason for this, is that both the photometric as well as the geometric morphings

can be implemented directly on the GPU. For example, the photometric morphing operation (associated with, say, morphable model $\mathcal{L}_{k\to k+1}$) reduces to defining a simple so-called pixel shader. This is a user-defined script that is automatically executed by the GPU for each output pixel. In this case, that script should simply be responsible for blending corresponding pixels (as specified by $u_{k\to k+1}$) from photometric images $\mathcal{I}_k$ and $\mathcal{I}_{k+1}$. Similarly, geometric morphing amounts to a simple vertex blending operation, that is, an operation that is directly supported by all modern GPUs. Hence, rendering a morphable model essentially reduces to feeding a textured triangle mesh into the GPU (with pixel and vertex shaders being applied to this mesh). This is, of course, a highly optimized rendering path in all modern GPUs, and thus a considerable amount of acceleration can be attained in this manner.

Sample results from the visual reconstruction of the Samaria gorge are shown in Figure 12. That figure contains some rendered views that were generated in real time as the virtual camera traversed a path through the so-called *Iron Gates* area, which is one of the most famous parts of the gorge. A corresponding video, with just a short clip from a virtual tour into the Samaria Gorge, is also available at the URL in [29]. Also, Figure 13 contains some rendered views of the gorge, where a synthetically generated volumetric fog has been added to the scene to further enhance the visual experience of the virtual tour.

8.1. Extensions

Another difficulty that we had to face, during the visual reconstruction of the Samaria Gorge, was related to the fact that a small river was passing through a certain part of the gorge. This was a problem for the construction of the corresponding local 3D models as our stereo matching algorithm could not possibly extract disparity (i.e., find correspondences) for the points on the water surface. This was the case because the water was not always static and, furthermore, the sun reflections that existed on its surface were violating the Lambertian assumption during stereo matching (see Figure 14(a)). Therefore, the disparity for all pixels lying on the water had to be estimated in a different way. To this end, since the water surface was approximately planar, we made the assumption that a 2D homography (i.e., a 2D projective transformation represented by a 3×3 homogeneous matrix $\mathcal{H}_{\text{water}}$) can directly map the left-image pixels that lie on the water to their corresponding points in the right image. For estimating $\mathcal{H}_{\text{water}}$, we took advantage of the fact that most left-image pixels that are located on the ground next to the river lie approximately at the same plane as the water surface and, in addition to that, stereo matching can extract valid correspondences for these pixels, as they are not on the water. A set $\{g^i_{\text{left}}\}_{i=1}^K$ of such pixels in the left image is thus extracted and their matching points $\{g^i_{\text{right}}\}_{i=1}^K$ in the right image are also computed based on the already estimated disparity maps. Using this sparse set of matches, the matrix $\mathcal{H}_{\text{water}}$ can then be easily recovered via using an RANSAC-based procedure to cope with outliers in the matches $\{g^i_{\text{left}}, g^i_{\text{right}}\}$. Inlier matches found by RANSAC can then be used for refining the estimation of $\mathcal{H}_{\text{water}}$ by minimizing the total reprojection error $E(\mathcal{H}_{\text{water}})$, that is, the sum of distances between $\{\mathcal{H}_{\text{water}} \cdot g^i_{\text{left}}\}_{i=1}^K$ and $\{g^i_{\text{right}}\}_{i=1}^K$ (note that points g^i_{left} and g^i_{right} are represented in homogeneous coordinates):

$$E(\mathcal{H}_{\text{water}}) = \sum_i ||\mathcal{H}_{\text{water}} \cdot g^i_{\text{left}} - g^i_{\text{right}}||. \tag{18}$$

Given the matrix $\mathcal{H}_{\text{water}}$, the disparity of an image point p on the water surface can then be set equal to $\mathcal{H}_{\text{water}} \cdot p - p$. An example of a disparity field that has been estimated with this method can be seen in Figure 14(c). We should note that, by using a similar method, a 2D homography $\mathcal{H}^{k\to k+1}_{\text{water}}$, mapping pixels of $\mathcal{I}_k$ lying on the water to their corresponding pixels in image $\mathcal{I}_{k+1}$, can be computed as well. In this way, we can achieve estimating an optical flow field $u_{k\to k+1}$ for all the pixels of image $\mathcal{I}_k$ that are lying on the water surface.

9. CONCLUSIONS AND DISCUSSION

In conclusion, we have presented a new approach for obtaining photorealistic and interactive walkthroughs of large, outdoor scenes. To this end, a novel hybrid data structure has been proposed, called "morphable 3D-mosaics". The resulting framework offers a series of important advantages. (1) To start with, no global 3D model of the environment needs to be assembled, a process which can be extremely cumbersome and error-prone, especially for large-scale scenes, where many local models need to be registered to each other. (2) Furthermore, rendering such a global model would make a very inefficient operation, with no inherent support for scalability, thus leading to very low frame rates for large-scale scenes. On the contrary, our method is extremely scalable to large-scale environments, as only one morphable 3D-model needs to be displayed at any time. (3) On top of this, it also makes use of a rendering path, which is highly optimized in modern 3D graphics hardware. (4) Not only that, but by utilizing an image-based data representation, our framework is also capable of fully reproducing the photorealistic richness of the scene. (5) As another advantage, the data acquisition process is very easy (e.g., collecting the stereoscopic images for a path over 100 meters long took us only about 20 minutes), while it also requires no special or expensive equipment (but just a pair of digital cameras and a tripod). (6) Finally, our framework makes up an end-to-end system, thus providing an almost automated processing of the input data, which are just a sparse set of stereoscopic images from the scene.

In the future, we intend to eliminate the need for calibrating the stereoscopic camera, as well as to allow the stereo baseline to vary during the acquisition of the stereoscopic views (these enhancements would allow for an even more flexible data acquisition process). Another issue that we wish to investigate is the ability of capturing the dynamic appearance of any moving objects, such as moving water or grass, that are frequently encountered in outdoor scenes (instead of just rendering these objects as static). To

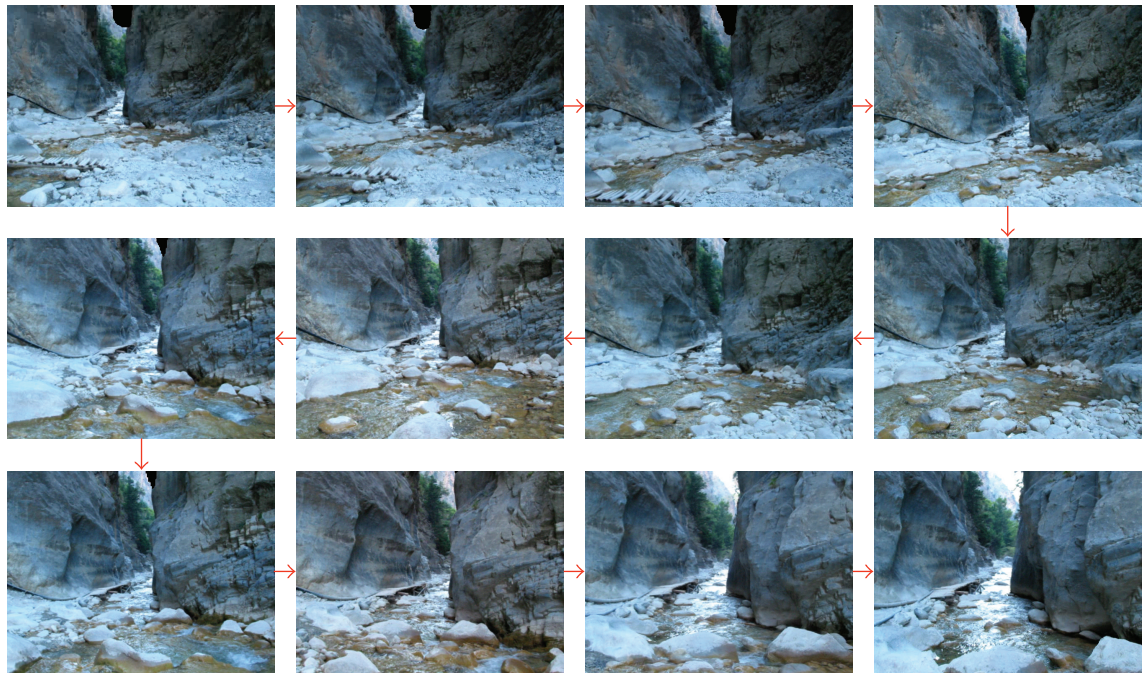
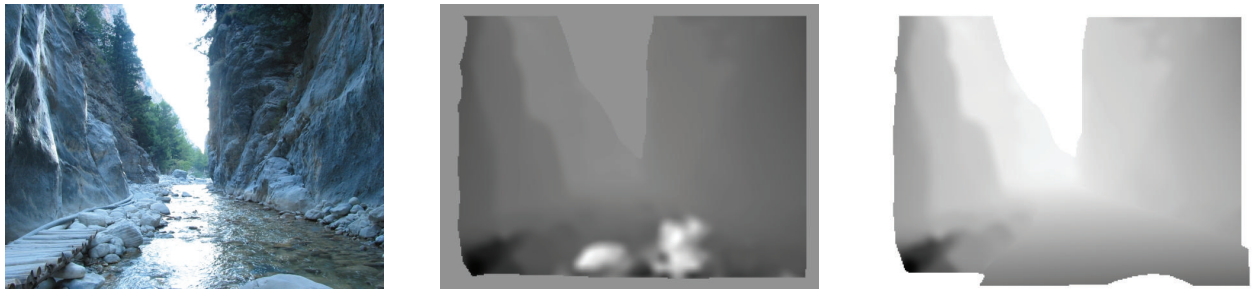


Figure 12: Some rendered views that are produced as the virtual camera traverses a path through the so-called "Iron Gates" area, which is the most famous part of the Samaria Gorge. In this case, the virtual camera passes through multiple morphable 3D models.

(a) (b) (c)

Figure 13: One of the additional benefits of having a virtual 3D reconstruction of the gorge is the ability, for example, to add synthetic visual effects. Here, for instance, a synthetically generated volumetric fog has been added to the scene.



(a) (b) (c)

Figure 14: (a) The left image of a stereoscopic image pair that has been captured at a region passing through a small river. (b) The estimated disparity by using a stereo matching procedure. As expected, the disparity field contains a lot of errors for many of the points on the water surface. This is true especially for those points that lie near the sun reflections on the water. (c) The corresponding disparity when a 2D homography is being used to fill the left-right correspondences for the points on the water. In this case, the water surface is implicitly approximated by a 3D plane.

this end, we plan to enhance our "morphable 3D-mosaic" framework, so that it can also make use of real video textures that have been previously captured inside the scene. Finally, a current limitation of our method is that it assumes that the lighting conditions across the scene are not drastically different (something which is not always true in outdoor environments). One possible approach to deal with this issue is by obtaining the radiometric response function of each photograph. In this case, when constructing the morphable 3D models, one should also use the estimated response functions so as to compensate for the image differences due to the varying lighting conditions.

ACKNOWLEDGMENTS

This paper is part of the 03ED417 research project, implemented within the framework of the Reinforcement Programme of Human Research Manpower (PENED) and cofinanced by National and Community Funds (75% from E.U.-European Social Fund and 25% from the Greek Ministry of Development-General Secretariat of Research and Technology).

REFERENCES

[1] H.-Y. Shum, R. Szeliski, S. Baker, M. Han, and P. Anandan, "Interactive 3D modeling from multiple images using scene regularities," in *Proceedings of the European Workshop on 3D Structure from Multiple Images of Large-Scale Environments (SMILE '98)*, pp. 236–252, Freiburg, Germany, June 1998.

[2] D. Nistér, "Automatic passive recovery of 3D from images and video," in *Proceedings of the 2nd International Symposium on 3D Data Processing, Visualization, and Transmission (3DPVT '04)*, pp. 438–445, Thessaloniki, Greece, September 2004.

[3] R. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision*, Cambridge University Press, Cambridge, UK, 2000.

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

[5] Y. Ma, S. Soatto, J. Kosecka, and S. Sastry, *An Invitation to 3D Vision*, Springer, New York, NY, USA, 2005.

[6] M. Pollefeys, L. Van Gool, M. Vergauwen, et al., "Visual modeling with a hand-held camera," *International Journal of Computer Vision*, vol. 59, no. 3, pp. 207–232, 2004.

[7] P. E. Debevec, C. J. Taylor, and J. Malik, "Modeling and rendering architecture from photographs: a hybrid geometry- and image-based approach," in *Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '96)*, pp. 11–20, New Orleans, La, USA, August 1996.

[8] L. McMillan and G. Bishop, "Plenoptic modeling: an image-based rendering system," in *Proceedings of the 22nd Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '95)*, pp. 39–46, Los Angeles, Calif, USA, August 1995.

[9] C. Zhang and T. Chen, "A survey on image-based rendering: representation, sampling and compression," Tech. Rep. AMP 03-03, Electrical and Computer Engineering, Carnegie Mellon University, Pittsburgh, Pa, USA, 2003.

[10] M. Levoy and P. Hanrahan, "Light field rendering," in *Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '96)*, pp. 31–42, New Orleans, La, USA, August 1996.

[11] S. Gortler, R. Grzeszczuk, R. Szeliski, and M. Cohen, "The lumigraph," in *Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '96)*, pp. 43–54, New Orleans, La, USA, August 1996.

[12] E. H. Adelson and J. R. Bergen, "The plenoptic function and the elements of early vision," in *Computational Models of Visual Processing*, pp. 3–20, MIT Press, Cambridge, Mass, USA, 1991.

[13] N. Komodakis, G. Pagonis, and G. Tziritas, "Interactive walkthroughs using "morphable 3D-mosaics"," in *Proceedings of the 2nd International Symposium on 3D Data Processing, Visualization, and Transmission (3DPVT '04)*, pp. 404–411, Thessaloniki, Greece, September 2004.

[14] N. Komodakis and G. Tziritas, "Morphable 3D-mosaics: a framework for the visual reconstruction of large natural scenes," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR '06)*, New York, NY, USA, June 2006.

[15] C. Buehler, M. Bosse, L. McMillan, S. Gortler, and M. Cohen, "Unstructured lumigraph rendering," in *Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '01)*, pp. 425–432, Los Angeles, Calif , USA, August 2001.

[16] H. Schirmacher, L. Ming, and H.-P. Seidel, "On-the-fly processing of generalized lumigraphs," *Computer Graphics Forum*, vol. 20, no. 3, pp. 165–174, 2001.

[17] M. Uyttendaele, A. Criminisi, S. B. Kang, S. Winder, R. Szeliski, and R. Hartley, "Image-based interactive exploration of real-world environments," *IEEE Computer Graphics and Applications*, vol. 24, no. 3, pp. 52–63, 2004.

[18] D. G. Aliaga, T. Funkhouser, D. Yanovsky, and I. Carlbom, "Sea of images," in *Proceedings of the IEEE Conference on Visualization (VIS '02)*, pp. 331–338, Boston, Mass, USA, October-November 2002.

[19] L. Yang and R. Crawfis, "A practical system for constrained interactive walkthroughs of arbitrarily complex scenes," in *Scientific Visualization: The Visual Extraction of Knowledge from Data*, Springer, New York, NY, USA, 2006.

[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] N. Komodakis, G. Tziritas, and N. Paragios, "Fast, approximately optimal solutions for single and dynamic MRFs," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR '07)*, pp. 1–8, Minneapolis, Minn, USA, June 2007.

[22] N. Komodakis and G. Tziritas, "Approximate labeling via graph cuts based on linear programming," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 29, no. 8, pp. 1436–1453, 2007.

[23] S. Z. Li, *Markov Random Field Modeling in Computer Vision*, Springer, London, UK, 1995.

[24] 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.

[25] D. Zwilliger, *Handbook of Differential Equations*, Academic Press, Boston, Mass, USA, 1997.

[26] P. Pérez, M. Gangnet, and A. Blake, "Poisson image editing," *ACM Transactions on Graphics*, vol. 22, no. 3, pp. 313–318, 2003.

[27] M. Segal and K. Akeley, "The OpenGL Graphics System: A Specification (Version 1.5)," http://www.opengl.org/.

[28] P. S. Heckbert and M. Garland, "Survey of polygonal surface simplification algorithms," in *Proceedings of the 24th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '97)*, Los Angeles, Calif, USA, August 1997.

[29] http://www.csd.uoc.gr/~komod/research/morphable_3d_mosaics/.