Hindawi
Journal of Sensors
Volume 2021, Article ID 5526931, 8 pages
https://doi.org/10.1155/2021/5526931

Research Article

3D Distance Measurement from a Camera to a Mobile Vehicle, Using Monocular Vision

Saúl Martínez-Díaz

División de Estudios de Posgrado e Investigación, Tecnológico Nacional de México-Instituto Tecnológico de La Paz, La Paz 23080, Mexico

Correspondence should be addressed to Saúl Martínez-Díaz; smdiaz06@hotmail.com

Received 16 January 2021; Revised 16 March 2021; Accepted 5 April 2021; Published 16 April 2021

Academic Editor: José A. Padilla-Medina

Estimation of distance from objects in real-world scenes is an important topic in several applications such as navigation of autonomous robots, simultaneous localization and mapping (SLAM), and augmented reality (AR). Even though there is a technology for this purpose, in some cases, this technology has some disadvantages. For example, GPS systems are susceptible to interference, especially in places surrounded by buildings, under bridges or indoors; alternatively, RGBD sensors can be used, but they are expensive, and their operational range is limited. Monocular vision is a low-cost suitable alternative that can be used indoor and outdoor. However, monocular odometry is challenging because the object location can be known up a scale factor. Moreover, when objects are moving, it is necessary to estimate the location from consecutive images accumulating error. This paper introduces a new method to compute the distance from a single image of the desired object, with known dimensions, captured with a monocular calibrated vision system. This method is less restrictive than other proposals in the state-of-the-art literature. For the detection of interest points, a Region-based Convolutional Neural Network combined with a corner detector were used. The proposed method was tested on a standard dataset and images acquired by a low-cost and low-resolution webcam, under noncontrolled conditions. The system was tested and compared with a calibrated stereo vision system. Results showed the similar performance of both systems, but the monocular system accomplished the task in less time.

1. Introduction

In recent years, due to the increase in the processing capacity of computers, it has been possible to process digital images in real time. With this technological improvement, it also grew the interest of the scientific community in developing systems for pose and location estimation based on artificial vision. One reason is because there are many applications in which this technology can be used. These applications include the navigation of autonomous robots [1, 2], simultaneous localization and mapping (SLAM) in unknown places [3, 4], augmented reality (AR) [5], and inspection of industrial systems [6, 7]. Many of these applications require information of object location in three-dimensional (3D) real world coordinates.

GPS systems are normally used for 3D real world locations, but this technology is susceptible to interference, especially in places surrounded by buildings, under bridges or indoors. Furthermore, they have large error margins, up to several decimetres. On the other hand, RGBD cameras can be used too; however, in addition to their high cost, they use infrared sensors to determine the distance from the camera to object (depth). This hinders and even prevents its application in some places illuminated with natural light.

In some cases, vision systems offer some advantages over the dominant technologies currently used at a low cost. A vision system can use a single camera, called a monocular system, or two (or more) cameras, called a stereo system. In stereo systems, usually, it is necessary to perform a stereoscopic calibration to know the rotation and translation of one camera with respect to the other. Generally, the first camera position serves as a coordinate reference system. The stereo calibration parameters must be kept fixed during all estimation process to reach good results. To calculate the location of a point in three-dimensional space, each camera must capture an image containing that point; then, it is necessary to match the corresponding points and identify its 2D coordinates within the two images; finally, triangulation is performed to obtain their 3D coordinates. However, there are some practical disadvantages in this type of systems based on two (or more) cameras. The main of these problems are the difference in the response of each camera to the colour and luminance of the input signal makes difficult the matching of corresponding points; these systems require more physical space and consume more energy, and the computational cost is higher because it needs to process two images on each occasion; it is possible that the cameras lose calibration due to movements or vibrations, mainly when cameras are attached on a mobile vehicle. In addition, when distant points on two cameras are observed, the system degenerates and tends to behave like a monocular system. All above puts monocular systems as a good alternative. Moreover, monocular vision systems can be designed with low-cost hardware. Besides, it can be used for indoor and outdoor applications with low error rates.

To estimate the pose in monocular systems, since just one camera is used, it is necessary to move the same camera and capture images in different positions. Because this movement is unknown, each new relative pose must be estimated. In the literature, two approaches have proven successful for monocular pose estimation: filtering methods [8–11] and keyframe-based methods [12–15]. Pioneering work of Davison et al. [8] recovered trajectories from a monocular camera by detecting natural landmarks using the Shi and Tomasi [16] operator; they made a probabilistic estimation of the state of the moving camera with an Extended Kalman Filter (EKF). In that approach, every frame is processed by the filter to jointly estimate the map feature locations and the camera pose. However, Strasdat et al. [17] showed that keyframe-based techniques are more accurate than filtering for the same computational cost. This method needs to keep the information of captured views from each image. Besides, optimization methods are required to reduce the estimation errors. Usually, the optimization is computationally expensive because it involves computing the reprojection error from all accumulated views. Despite this, with time, the error increases considerably.

Other important problem in monocular systems is that the location of a point in 3D space can only be known up to a certain scale factor. This scale factor must be obtained by using any bootstrap method. In the literature, several approaches have been proposed to estimate this scale factor: in reference [18], the authors use a camera as the main sensor and an inertial measurement unit (IMU) to determine the scale. In [19], the depth is estimated by using a convolutional neural network, this estimation is refined, and the error is reduced by training the network with consecutive images. In [20], the authors used a pattern of three concentric circles of known diameter in a plane; the camera must be perpendicular to the circles plane, to calculate the initial depth. In [21], it is assumed that the field of vision of one camera mounted on an airship is always perpendicular to the earth. Subsequently, the camera is placed at a known distance from the face of a person (centered in the image). Then, pixels of the detected face are counted; finally, with these data, a relationship is established to calculate the depth when detecting the same face in future images from different distances from the camera.

In this paper, we introduce a new technique to calculate the 3D location of a moving vehicle. Depth calculation is based on a group of three points from the vehicle, using a calibrated monocular system. Only the distance between each pair of reference points must be known for depth computation. The points can be in any position and the camera is not required to be perpendicular to the plane formed by the three points. Using this technique, it is possible to calculate the pose (rotation and translation) of the camera from each image, avoiding error accumulation. To detect the reference object, we use transfer learning with a pretrained Convolutional Neural Network. Computer simulations show a good performance of the proposed method in terms of precision and speed of processing.

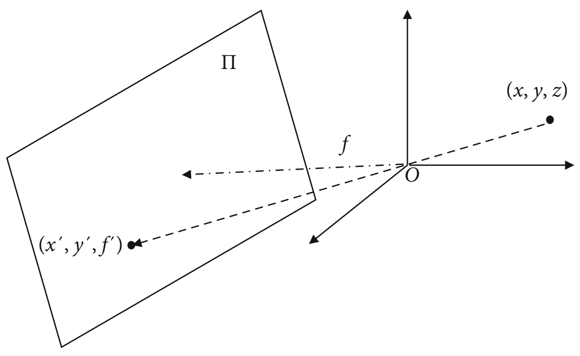


Figure 1: Pinhole model. The three dimensional point (x, y, z) is projected through the pinhole to the two dimensional plane Π.

2. Materials and Methods

Most vision techniques rely on the camera model and calibration. Camera model is a geometrical approximation of how light travels through the camera lens and forms images. Camera calibration is required to correct the main deviations due to the model used. Besides, camera calibration can be used to relate camera measurements in pixels, to the real three-dimensional world. In this section, we review some basic principles of pinhole camera model, calibration, and triangulation methods.

2.1. Pinhole Model. Due to its simplicity, the pinhole model is widely used to represent the formation of images in a camera. In this model, it is supposed a single ray of light entering to the camera and projected onto an imaging plane (projective plane). Figure 1 illustrates the principle on which this model is based on. As can be seen, each point **P** with coordinates (x, y, z) in 3D space is projected through the pinhole (which is taken as the origin of the coordinate system) to point $\mathbf{P}'$ with coordinates (x', y', f) of the plane Π into the camera. Here, f is the focal length of the camera. From Figure 1, by the similarity of triangles, it can be established that [22]

$$\frac{x'}{x} = \frac{y'}{y} = \frac{f}{z} = \lambda, \tag{1}$$

where λ is a scale factor. If the focal length and λ are known, it is possible to calculate the coordinates of the 3D point **P** from the 2D coordinates of the projected point onto the image plane and the focal length. Usually, the focal length together with the intrinsic and extrinsic parameters can be obtained from the camera calibration process.

2.2. Camera Calibration. The basic pinhole model does not include any kind of distortions, which usually occur in real world cameras. Camera calibration gives a model of the camera's geometry and distortions caused by lenses. This information can be used to define the intrinsic and extrinsic parameters of the camera.

Let $\mathbf{P}'$ be the projected point of a 3D point **P** in the camera plane (as shown in Figure 1). By using homogeneous coordinates, we define $\mathbf{P} = [X\,Y\,Z\,1]^T$ and $\mathbf{P}' = [x\,y\,1]^T$. Then, we can express the mapping from **P** to $\mathbf{P}'$ in terms of matrix multiplication as [23]

$$\lambda\mathbf{P}' = \mathbf{A}[\mathbf{R}\,\mathbf{t}]\mathbf{P}, \tag{2}$$

where **P** is a point of the object in 3D, in homogeneous coordinates; $\mathbf{P}'$ is the same point of the object in homogeneous 2D coordinates; $[\mathbf{R}\,\mathbf{t}]$ is a matrix of extrinsic parameters (rotation and translation); λ is an arbitrary scale factor, and **A** is a matrix of intrinsic parameters. The information of the intrinsic parameter matrix is defined by

$$\mathbf{A} = \begin{bmatrix} f_x & s & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix}. \tag{3}$$

Here, f_x, f_y provide information (depending on the size of the pixel) of the focal distance in the direction of x and y , respectively; c_x and c_y are the coordinates of the main point of the image; s is known as skew and represents the angle of inclination of the pixel.

The pose of an object, relative to the camera coordinate system, could be described in terms of the rotation matrix **R** and the translation vector **t**. Rotation around x, y, and z axes can be represented by rotation matrices $\mathbf{R}_x$, $\mathbf{R}_y$, and $\mathbf{R}_z$, respectively:

$$\mathbf{R}_x = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\alpha) & \sin(\alpha) \\ 0 & -\sin(\alpha) & \cos(\alpha) \end{bmatrix},$$
$$\mathbf{R}_y = \begin{bmatrix} \cos(\beta) & 0 & -\sin(\beta) \\ 0 & 1 & 0 \\ \sin(\beta) & 0 & \cos(\beta) \end{bmatrix}, \tag{4}$$
$$\mathbf{R}_z = \begin{bmatrix} \cos(\theta) & -\sin(\theta) & 0 \\ \sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix}.$$

Here, α, β, and θ are rotation angles around x, y, and z axes, respectively. Finally, the rotation matrix **R** can be composed by the multiplication of the three rotation matrices.

2.3. Triangulation. The process of determining the 3D coordinates of a point in space, given its projection onto two or more images, is called triangulation. First, for triangulation, it is required for the detection and extraction of interest points, such as corners or salient features in images. Then, it is necessary to match the corresponding points in at least two images (find the same points in them). After that, selecting the strongest match with respect to a stablished threshold is needed (keeping index of the strongest matching points in the input set). Next, the triangulation of the matched features is achieved as follows: suppose we have a set of corresponding points $x_i \leftrightarrow x_i'$ in two images, and there exist some camera matrices $\mathbf{P}$, $\mathbf{P}'$, and a set of 3D points X_i that give rise to these image correspondences in the sense that $\mathbf{P}X_i = x_i$ and $\mathbf{P}'X_i = x_i'$. These equations can be combined into the form $\mathbf{AX} = \mathbf{b}$, which is an equation linear in **X**, with

$$\mathbf{A} = \begin{bmatrix} x\boldsymbol{p}^{3T} - \boldsymbol{p}^{1T} \\ y\boldsymbol{p}^{3T} - \boldsymbol{p}^{2T} \\ x'\boldsymbol{p}'^{3T} - \boldsymbol{p}'^{1T} \\ y'\boldsymbol{p}'^{3T} - \boldsymbol{p}'2T \end{bmatrix}, \tag{5}$$

where $\mathbf{p}^{iT}$ are the rows of **P**. Solution of this redundant set of equations gives us the triangulation result. The system can be solved by finding the unit singular vector corresponding to the smallest singular value of **A**.

2.4. Proposed Method. The main steps of the method in pseudocode are as follows:

```
(1) Capture image
(2) Search the desired object using a Region-based Convolutional Neural Network (R-CNN)
(3) If the vehicle is detected, find corners inside the region containing it
(4) Compute the depth (z coordinate in 3D world) from the camera with interest corner
(5) Estimate the distance from the camera to the centroid of the corners
```

The next subsections explain each step.

2.5. Convolutional Neural Network. Unlike traditional machine learning methods for classification, in which features must be chosen manually and extracted with specialized algorithms, deep learning networks automatically discover relevant features from data. CNNs are composed of an input layer, an output layer, and many hidden layers in between [24]:

Convolutional Layer. Units in a convolutional layer are organized in feature maps. Within them, each unit is connected to local patches in the feature maps of the previous layer through a set of weights, called a filter bank. Each filter activates certain features from the images.

Pooling Layer. This layer is used to merge semantically similar features into one, to simplify the output, by performing nonlinear downsampling, which reduces the number of parameters that the network needs to learn. The pooling layer takes a pool size as a hyperparameter, usually 2 by 2. It then processes its input image in the following way: divide the image in a grid of 2 by 2 areas and take from each four-pixel a representative value (normally the maximum value is used).

Rectified Linear Unit (ReLU). The result of the convolution is then passed through a nonlinearity called ReLU, which allows faster and more effective training by mapping negative values to zero and maintaining positive values.

These three operations are repeated over tens or hundreds of layers, with each layer learning to detect different features. After feature detection, the architecture of a CNN shifts to classification. The next-to-last layer is a fully connected layer that outputs a vector of K dimensions, where K is the number of classes that the network will be able to predict. This vector contains the probabilities for each class of any image being classified. The final layer of the CNN architecture uses a softmax function to provide the classification output.

2.6. Region-Based CNN. Contrasting with classification, detection requires the accurate localization of objects (probably many) in images. For that purpose, Region-based CNN (R-CNN) were proposed. In this case, the network finds regions of interest (ROIs) where the object probably can be found. Compared to image classification, object detection is a more challenging task that requires more complex methods to solve it. Complexity arises because detection requires the accurate localization and refinement of many objects. Several algorithms have been proposed for training R-CNNs, some methods use multistage pipelines, but they are slow. Other methods use a sliding window technique to generate region proposals.

For this task, region-based methods have shown better performance than the other methods. In this case, the first step is proposing several candidate object localizations; then, the proposals must be refined to achieve precise localization. Each region must be evaluated, and its membership to any class of object vs. background is scored. From the latter, the most popular algorithms are regions with CNN (R-CNN) [25], Fast Region-based Convolutional Network (Fast R-CNN) [26], and Faster Region-based Convolutional Network (Faster R-CNN) [27].

2.7. Transfer Learning. The success of CNNs depends on the number of images used to train them. Usually, thousands or millions of images are required to achieve good results. Training a CNN from scratch may require weeks of computation in a high-performance computer. Besides, preparing manual images for training consumes so much time. A better option is to use a pretrained network and fine-tune it with a few images to make it work in our desired task. This method is called transfer learning.

Transfer learning refers to the situation where what has been learned in one setting is exploited to improve generalization in another setting [28]. For example, one CNN is trained to recognize one set of visual categories, such as cars, then is fine-tuned to learn about a different set of visual categories, such as trucks. Here, a key point about image data is that the extracted features from a data set are highly reusable across other data sources. Usually, only the deeper layers are fine-tuned, and the weights of the early layers are fixed. The reason for training only the deeper layers, while keeping the early layers fixed, is that the earlier layers capture only primitive features like edges, whereas the deeper layers capture more complex features. The primitive features do not change too much with the application at hand, whereas the deeper features might be sensitive to the application at hand. Some popular pretrained CNNs available for transfer learning are AlexNet [29], ResNet [30], and GoogLeNet [31].

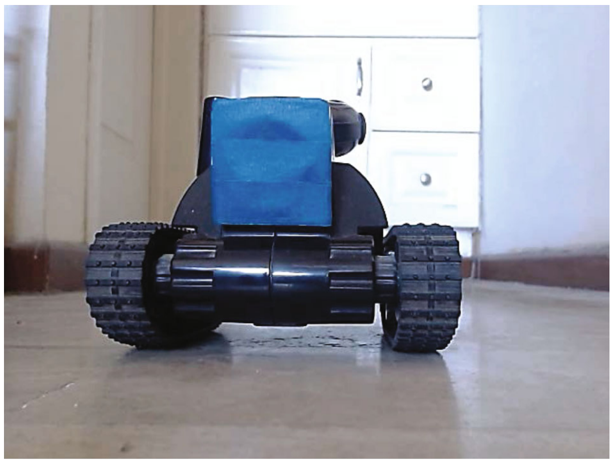


FIGURE 2: Miniature mobile vehicle used in experiments.

2.8. Corners Detection. Corners are features invariant to translation, rotation, and illumination. As well, there exist robust algorithms to detect them. The main idea for corner detection is searching for strong derivatives in two orthogonal directions of image I. This can be done by applying a matrix of second-order derivatives (Hessian matrix **Hs**) to the image intensities [32]:

$$\mathbf{Hs}(p) = \begin{bmatrix} \dfrac{\partial^2 I}{\partial x^2} & \dfrac{\partial^2 I}{\partial x \partial y} \\ \dfrac{\partial^2 I}{\partial y \partial x} & \dfrac{\partial^2 I}{\partial y^2} \end{bmatrix}. \tag{6}$$

In practice, the most used approach is by applying the autocorrelation matrix **M** of the second derivative image over a small window W around each point:

$$\mathbf{M}(x, y) = \begin{bmatrix} \sum_{x,y\in W} I_x^2(x, y) & \sum_{x,y\in W} I_x(x, y) I_y(x, y) \\ \sum_{x,y\in W} I_x(x, y) I_y(x, y) & \sum_{x,y\in W} I_y^2(x, y) \end{bmatrix}. \tag{7}$$

2.9. Depth Computation. In this section, we describe the proposed depth calculation method. From Equation (1), the following relationships can be obtained:

$$x = \frac{x'}{f} z, \qquad y = \frac{y'}{f} z. \tag{8}$$

Now, suppose we have at least three points $\mathbf{P}_1$, $\mathbf{P}_2$, and $\mathbf{P}_3$ in 3D space with coordinates (x_1, y_1, z_1), (x_2, y_2, z_2), and (x_3, y_3, z_3), respectively. Suppose also that the distances d_1, d_2, and d_3 between each pair of points are known. Using the relationships in (8), the absolute distance between each pair of points can be calculated by:

$$\begin{aligned} d_1 &= \left| \frac{x'_1}{f} z_1 - \frac{x'_2}{f} z_2 \right| + \left| \frac{y'_1}{f} z_1 - \frac{y'_2}{f} z_2 \right| + |z_1 - z_2|, \\ d_2 &= \left| \frac{x'_1}{f} z_1 - \frac{x'_3}{f} z_3 \right| + \left| \frac{y'_1}{f} z_1 - \frac{y'_3}{f} z_3 \right| + |z_1 - z_3|, \\ d_3 &= \left| \frac{x'_3}{f} z_3 - \frac{x'_2}{f} z_2 \right| + \left| \frac{y'_3}{f} z_3 - \frac{y'_2}{f} z_2 \right| + |z_3 - z_2|. \end{aligned} \tag{9}$$

In this system of nonlinear Equation (9), z_i represents the depth at which each point is located with respect to the camera. Note that each absolute value term $|A - B|$ has two possible solutions: $A - B$ if $A > B$ or $B - A$ if $A \le B$. Then, by combining all solutions of the nine absolute value terms, we can obtain 512 linear equation systems. For example, one of these systems is

$$\begin{aligned} d_1 &= \left(\frac{x'_1}{f} z_1 - \frac{x'_2}{f} z_2 \right) + \left(\frac{y'_1}{f} z_1 - \frac{y'_2}{f} z_2 \right) + (z_1 - z_2), \\ d_2 &= \left(\frac{x'_1}{f} z_1 - \frac{x'_3}{f} z_3 \right) + \left(\frac{y'_1}{f} z_1 - \frac{y'_3}{f} z_3 \right) + (z_1 - z_3), \\ d_3 &= \left(\frac{x'_3}{f} z_3 - \frac{x'_2}{f} z_2 \right) + \left(\frac{y'_3}{f} z_3 - \frac{y'_2}{f} z_2 \right) + (z_3 - z_2). \end{aligned} \tag{10}$$

The system in (10) can be rearranged and solved for z_i as

$$\begin{aligned} d_1 &= \left(\frac{x'_1}{f} + \frac{y'_1}{f} + 1 \right) z_1 - \left(\frac{x'_2}{f} + \frac{y'_2}{f} + 1 \right) z_2, \\ d_2 &= \left(\frac{x'_1}{f} + \frac{y'_1}{f} + 1 \right) z_1 - \left(\frac{x'_3}{f} + \frac{y'_3}{f} + 1 \right) z_3, \\ d_3 &= \left(\frac{x'_3}{f} + \frac{y'_3}{f} + 1 \right) z_3 - \left(\frac{x'_2}{f} + \frac{y'_2}{f} + 1 \right) z_2. \end{aligned} \tag{11}$$

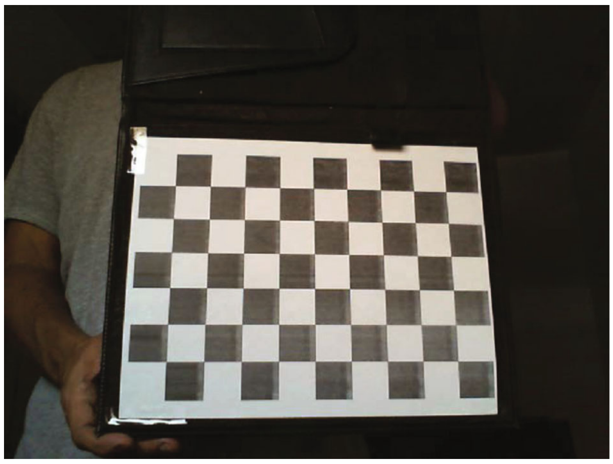
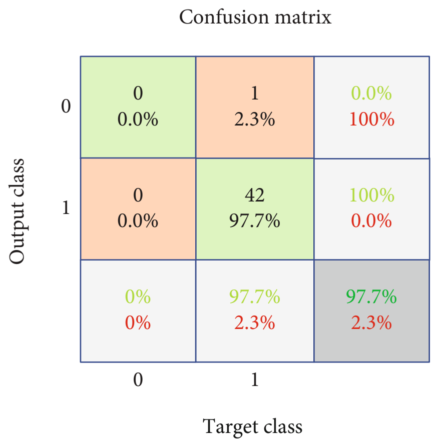


Figure 3: Pattern used to calibrate cameras.


Confusion matrix

Output class \ Target class	0	1	
0	0 0.0%	1 2.3%	0.0% 100%
1	0 0.0%	42 97.7%	100% 0.0%
	0% 0%	97.7% 2.3%	97.7% 2.3%

Figure 4: Confusion matrix resulting from Fast R-CNN training.

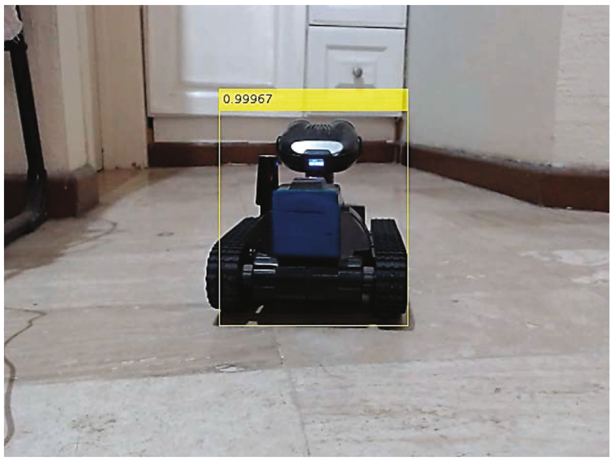


Figure 5: Object detected with Fast R-CNN.

To find the best solution, all 512 sets of equations must be computed. In (11), by changing sign to each term, there are eight combinations of values inside each parenthesis pair. Besides, the content inside each parenthesis pair is repeated in two different equations. Then, to reduce computations,

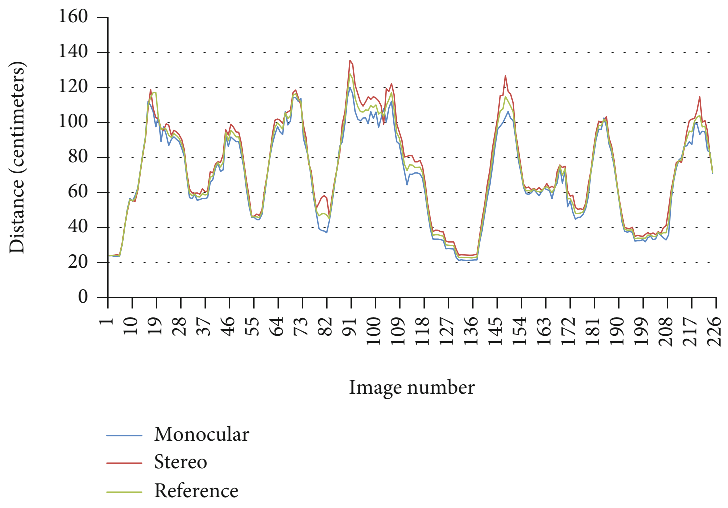


Figure 6: Comparison of distance measurement among monocular (proposed) system, stereo system, and manual reference.

all the 24 values can be calculated in advance. Next, these values are combined properly to compute the 512 solutions.

Subsequently, each solution can be tested in the original equations system (9) and keep the solutions that fulfill the restriction that all z_i are positive (the camera cannot capture points behind it). Finally, the solution with the least error is selected. Substituting these results in Equation (8) can also be obtained the real-world coordinates (x_i, y_i, z_i) of the three points. Relative poses (translation and rotation) of the camera respect to the vehicle can be determined too with such coordinates.

3. Results and Discussion

3.1. Experimental Configuration. In this section, by means of computer simulations, we illustrate performance of the proposed distance computation method. Due to stereo vision has proven good results computing distance from objects, we validated the results of the proposed method with the results obtained from a calibrated stereo vision system. Both, monocular and stereo system were compared with respect to manually measured distances. RGB images were obtained with two low-cost webcams at a resolution of $640 \times 480 \times 3$ pixels. Both cameras were fixed and calibrated for stereo operation. We use a set of 234 images containing the mobile vehicle shown in Figure 2. Vehicle dimensions are $21 \times 16 \times 15.5$ centimeters. The blue rectangle at the back of the vehicle is used as the reference to compute distance. The rectangle dimensions are 4.7×5 centimeters.

We use noncontrolled natural illumination. Calibration of the cameras was carried out with the technique proposed by Zhang [33]. With this calibration technique, it is only required that the camera observe a flat pattern taken from different orientations (as shown in Figure 3). The pattern or camera can be moved freely, and it is not necessary to know the movement made. This calibration allows to obtain simultaneously the intrinsic and extrinsic parameters of the camera. Each square is 27×27 millimeters in length.

3.2. Object Detector. For object detection, we tested three region-based algorithms: R-CNN [25], Fast R-CNN [26], and Faster R-CNN [27]. The best results were achieved with Fast R-CNN algorithm. The network takes as input an entire image and a set of object proposals. Then, the whole image with several convolutional and max pooling layers is processed to produce a map feature. Then, for each object proposed, a pooling layer extracts a fixed-length feature vector from the map. Each feature vector is fed into a sequence of fully connected layers that end in two output layers: one that produces softmax probability estimates over all object classes and a background.

For our implementation, we selected a pretrained net: AlexNet [29]. AlexNet was the winner of the 2012 ILSVRC competition and has been trained on over a million images and can classify images into 1000 object categories. The network has learned rich feature representations for a wide range of images. It takes an image as input and outputs the probabilities for each of the object categories. The first layer (input layer) requires input images of size 227-by-227-by-3, where 3 is the number of color channels; then, each input image must be resized to such size. The last three layers of the pretrained network net are configured for 1000 classes. These three layers must be fine-tuned for the new classification problem. To retrain the selected net, we replaced the last three layers of the network. The new added layers were a fully connected layer, a softmax layer, and a classification layer. The final fully connected layer was set to have the same size as the number of classes in the new data set (one class: vehicle). To learn faster in the new layers than in the transferred layers, we increased the learning rate factors of the fully connected layer.

To train the network, we used 105 images containing the desired object. Each target was manually enclosed into a box, and the coordinates of each box were used as the ground truth. 60% of the images were used for training and 40% for testing. The network was trained using a low-performance GPU (GeForce GTX 960 M) with an Intel Pentium Core i7 processor with 8 Mb of RAM. The main parameters selected for training were gradient method L2-norm, epoch 50, minibatch size 8, momentum 0.9000, and initial learning rate 1.0000e-03. Figure 4 shows the confusion matrix of the results, where class one is for the handguns and class zero is for the background. As can be seen, the system misclassifies only 2.3% of the images (one false positive).

Figure 5 shows an example of performance of the trained network. As can be seen, the net recognizes the vehicle with 0.9997 of confidence, even with the low-quality images. Next, inside the ROI, the Harris detector is applied to identify corners of the blue rectangle (back of the vehicle).

3.3. Distance Calculation. For distance calculation, the vehicle was moved remotely, and images were captured in pairs (using the stereo system), from several poses. Monocular system computes distance as described previously. For the stereo system, the detected corners on the left and right cameras are matched, and its centroid is calculated. With such centroids, by means of a triangulation also described previously, the distance is computed. Figure 6 shows a comparative graph. The percentage of mean quadratic error was 0.27% for stereo system and 0.28% for monocular system respect to manual reference. This means that the proposed method performs similarly to a stereo system.

Since computational cost is an important issue, we compared the processing time of both systems. To guarantee statistically correct results, 30 tests were performed and averaged on each system. The stereo system got an average time of 0.01945 seconds; on the other hand, the monocular system got an average time of 0.01485 seconds. This means that the algorithm for monocular system is faster than the algorithm for stereo system.

4. Conclusions

In this paper, a new method to calculate the distance from a camera to a moving object in three-dimensional space, using a monocular vision system, was introduced. This information can serve for applications of industrial autonomous robot navigation, SLAM, augmented reality, or control systems, by using a single camera. For the calculation, it is only necessary to know three points and the pairwise distance between them. To detect objects and interest points, a R-CNN combined with a corner detector were used. Unlike other methods such as Mono-SLAM, this method avoids error accumulation since it computes the distance from each image and does not require optimization methods, such as bundle adjustment.

The experimental results shown a good performance of the distance computation algorithm in images taken with a low-cost camera, under uncontrolled conditions of illumination. The maximum error obtained was less than 0.9%, compared to a calibrated stereo vision system. Moreover, the system runs faster than the stereo system.

A drawback of this method is that it highly depends on a good detection and matching of interest points. Although modern Convolutional Neural Networks perform well in detecting objects, the computation cost is high yet.

Data Availability

Data can be requested by e-mail to smdiaz06@hotmail.com.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

References

[1] K. Wang, Y. Liu, and L. Li, "Vision-based tracking control of underactuated water surface robots without direct position measurement," *IEEE Transactions on Control Systems Technology*, vol. 23, no. 6, pp. 2391–2399, 2015.

[2] M. Knudson and K. Tumer, "Adaptive navigation for autonomous robots," *Robotics and Autonomous Systems*, vol. 59, no. 6, pp. 410–420, 2011.

[3] R. Mur-Artal and J. D. Tardos, "ORB-SLAM2: an open-source SLAM system for monocular, stereo and RGB-D cameras," *IEEE Transactions on Robotics*, vol. 33, no. 5, pp. 1255–1262, 2017.

[4] R. Mur-Artal and J. D. Tardos, "Visual-inertial monocular slam with map reuse," *IEEE Robotics and Automation Letters*, vol. 2, no. 2, pp. 796–803, 2017.

[5] D. Chatzopoulos, C. Bermejo, Z. Huang, and P. Hui, "Mobile augmented reality survey: from where we are to where we go," *IEEE Access*, vol. 5, pp. 6917–6950, 2017.

[6] L. Xu, X. He, X. Li, and M. Pan, "A machine-vision inspection system for conveying attitudes of columnar objects in packing processes," *Measurement*, vol. 87, pp. 255–273, 2016.

[7] R. Shanmugamani, M. Sadique, and B. Ramamoorthy, "Detection and classification of surface defects of gun barrels using computer vision and machine learning," *Measurement*, vol. 60, pp. 222–230, 2014.

[8] A. J. Davison, I. D. Reid, N. D. Molton, and O. Stasse, "MonoSLAM: real-time single camera SLAM," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 29, no. 6, pp. 1052–1067, 2007.

[9] J. Civera, A. J. Davison, and M. M. Montiel, "Inverse depth parametrization for monocular SLAM," *IEEE Transactions on Robotics*, vol. 24, no. 5, pp. 932–945, 2008.

[10] A. Chiuso, P. Favaro, H. Jin, and S. Soatto, "Structure from motion causally integrated over time," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 24, no. 4, pp. 523–535, 2002.

[11] E. Eade and T. Drummond, "Scalable monocular SLAM," in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, vol. 1, pp. 469–476, New York City, 2006.

[12] E. Mouragnon, M. Lhuillier, M. Dhome, F. Dekeyser, and P. Sayd, "Real time localization and 3d reconstruction," in *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, pp. 363–370, New York, NY, USA, 2006.

[13] G. Klein and D. Murray, "Parallel tracking and mapping for small AR workspaces," in *IEEE and ACM International Symposium on Mixed and Augmented Reality (ISMAR)*, pp. 225–234, Nara, Japan, November 2007.

[14] R. Mur-Artal, J. M. M. Montiel, and J. D. Tardos, "ORB-SLAM: a versatile and accurate monocular SLAM system," *IEEE Transactions on Robotics*, vol. 31, no. 5, pp. 1147–1163, 2015.

[15] S. Leutenegger, S. Lynen, M. Bosse, R. Siegwart, and P. Furgale, "Keyframe-based visual–inertial odometry using nonlinear optimization," *International Journal of Robotics Research*, vol. 34, no. 3, pp. 314–334, 2015.

[16] J. Shi and C. Tomasi, "Good features to track," in *Proceedings IEEE Conf. Computer Vision and Pattern Recognition*, pp. 593–600, Seattle, USA, June 1994.

[17] H. Strasdat, J. M. M. Montiel, and A. J. Davison, "Visual SLAM: why filter?," *Image and Vision Computing*, vol. 30, no. 2, pp. 65–77, 2012.

[18] J. J. Tarrio and S. Pedre, "Realtime edge based visual inertial odometry for MAV teleoperation in indoor environments," *Journal of Intelligent and Robotic Systems*, vol. 90, no. 1-2, pp. 235–252, 2018.

[19] X. Yin, X. Wang, X. Du, and Q. Chen, "Scale recovery for monocular visual odometry using depth estimated with deep convolutional neural fields," in *2017 IEEE International Conference on Computer Vision*, pp. 5871–5879, Venice, Italy, October 2017.

[20] Z. Said, K. Sundaraj, and M. N. Wahab, "Depth estimation for a mobile platform using monocular vision," in *International Symposium on Robotics and Intelligent Sensors (IRIS 2012)*, vol. 41, pp. 945–950, Kuching, Sarawak, Malaysia, 2012.

[21] N. Yao, E. Anaya, Q. Tao, S. Cho, H. Zheng, and F. Zhang, "Monocular vision-based human following on miniature robotic blimp," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 3244–3249, Singapore, June 2017.

[22] A. I. Barranco-Gutiérrez, S. Martínez-Díaz, and J. L. Gómez-Torres, *Visión Estereoscópica con MATLAB y OPENCV (in Spanish)*, Pearson, México, 2018.

[23] R. Hartley and A. Zisserman, *Multiple view geometry in computer vision*, Cambridge University, 2004.

[24] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[25] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, vol. 2014, pp. 1–21, 2014.

[26] R. Girshick, "Fast R-CNN," in *IEEE International Conference on Computer Vision (ICCV), 2015*, pp. 1–9, Santiago, Chile, 2015.

[27] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: towards real-time object detection with region proposal networks," *Neural Information Processing Systems (NIPS)*, vol. 2015, pp. 1–9, 2015.

[28] I. Goodfellow, Y. Bengio, and A. Courville, *Deep learning*, MIT Press, 2016.

[29] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," *Communications of the ACM*, vol. 60, no. 6, pp. 84–90, 2017.

[30] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 770–778, Las Vegas, NV, USA, 2016.

[31] C. Szegedy, W. Liu, Y. Jia et al., "Going deeper with convolutions," in *IEEE, 2015 Conference on Computer Vision and Pattern Recognition (CVPR)*, vol. 4, pp. 10059–10066, Boston, MA, USA, 2016.

[32] C. Harris and M. Stephens, "A combined corner and edge detector," in *Proceedings of the 4th Alvey Vision Conference*, pp. 147–151, Manchester, UK, September 1988.

[33] 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.