Automatic Population of Scenarios with Augmented Virtuality

Oscar Ripolles, Jose Simo and Roberto Vivo

Abstract In this paper we propose a new augmented virtuality framework where the synthetic scenario is populated with the information coming from the real-world. Our proposal is based on a smart camera which processes the images to detect objects. With this information, our framework relies on the collisions of the optical rays with the scenario to locate the detected objects. Our solution is also capable of handling situations where objects are piled on others.

Keywords Augmented virtuality · Surveillance · Computer vision

1 Introduction

Video surveillance is an increasingly active research area, and different researchers have proposed systems for visual surveillance using mixed reality. Most surveillance solutions are based on augmented reality and very few systems offer augmented virtuality applications, where the virtual environments are *augmented* with data and media coming from the real world.

In this paper we present a new augmented virtuality framework using automatic video interpretation. The framework that we propose combines a computer graphics application with a smart camera which is currently being developed under

O. Ripolles (✉) · J. Simo · R. Vivo
Universitat Politècnica de València, Inst. Universitari d'Automàtica i Informàtica Industrial, Camino de Vera s/n, València, Spain
e-mail: oripolles@ai2.upv.es

J. Simo
e-mail: jsimo@ai2.upv.es

R. Vivo
e-mail: rvivo@ai2.upv.es

E. Gelenbe et al. (eds.), *Computer and Information Sciences II*,
DOI: 10.1007/978-1-4471-2155-8_44,

the SENSE project.[1] This smart camera performs image analysis onboard and outputs a stream of XML-coded information containing the details on the detected objects [1]. In this way, our aim is to develop a fully synthetic environment where the augmentation is only used to indicate the characteristics of the detected objects and their location. The graphics application uses the information coming from the smart camera to infer the 3D characteristics of each detected object. Previous solutions have typically used bounding boxes to locate the objects [2, 3], which result in very simple approximations. In our proposal we use the silhouette information to calculate the size of the objects in their three dimensions. Once the 3D scene is populated, the user can vary the view-point as desired to observe the scene from a better perspective.

This paper is structured as follows. The following section introduces previous solutions for the simulation of real scenarios in 3D. Section 3 details the proposed framework and describes the algorithms. Section 4 presents the results that we have obtained and, finally, the conclusions and the main ideas for future work are given in Sect. 5.

2 Previous Work

There are two main lines of research regarding augmented virtuality for surveillance. On the one hand, some authors propose combining 3D environments with real videos. In this sense, 3D scenarios are augmented by locating the real cameras on them and adding a textured quad on this position to render the video sources. These solutions aim at offering situational awareness so that the user has a clearer understanding of where the cameras are located [3, 4].

On the other hand, some authors propose using computer graphics techniques to estimate the position of the objects [2, 3]. This process is depicted in Fig. 1, where the synthetic camera (point A) is located and oriented as the real-world camera, while the image plane is located and oriented so that it matches exactly the perspective view from the synthetic camera. Authors commonly use very rough 3D scenarios and create a simple textured polygon at point C to represent the object with an image from the video source [2, 3, 5]. From a different perspective, other authors propose training the systems to detect a low range of objects. Then, when simulating a scene, the algorithms perform a classification of the objects by finding the combination of type of object, height and orientation that best fits the detected bounding box in the 2D image [6]. With a similar idea, Osawa et al. use the information of previous frames to produce a set of synthetic images by varying the position of the objects in the 3D scenario until the objects detected in the real and the synthetic images are the same [7].

[1] SENSE stands for Smart Embedded Network of Sensing Entities (Project 033279), developed under a grant from European Sixth Framework Programme

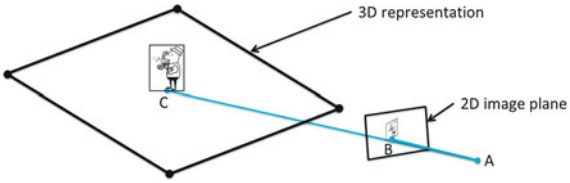

Fig. 1 Example of object location by casting a ray from A (*optical center*) through B (*object location on the image plane*) intersecting the floor in C (*calculated 3D position*)

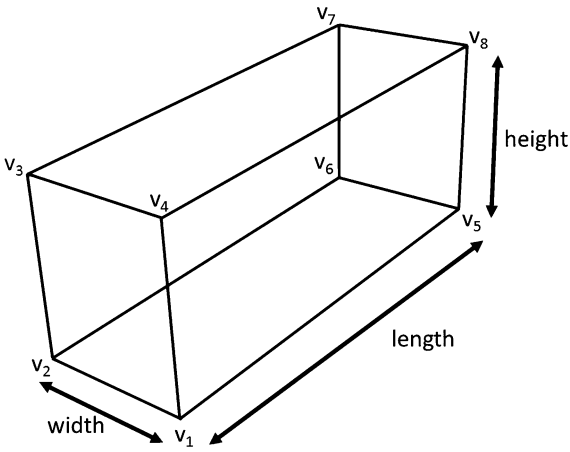

Fig. 2 Cuboid, the geometrical shape used to represented the detected objects

3 Augmenting the Virtual Environment

To convert the 2D information obtained from the camera into accurate 3D data, we follow an approach similar to the one presented in Fig. 1. The main difference with previous approaches is that we use multiple points from the silhouettes to obtain a much better approximation. For the visualization system we have chosen OpenSceneGraph as rendering engine [8]. For the correct performance of the proposed solution, the 3D scenario must be modeled with the actual sizes of the real-world and the camera must be correctly registered. Within the synthetic environment, each detected object is rendered as a cuboid (see Fig. 2).

Depending on the information retrieved from the camera, the system is capable of modeling the objects with more or less detail. For every detected object we are always able to use the information of their bounding box. Nevertheless, if the detected silhouette meets some requirements the framework is also capable of using the silhouette information to model the object with a higher accuracy.

3.1 Using the Bounding Box to Model the Object

The information on the bounding box that the smart camera outputs can be used to approximate the location and shape of the detected object. The algorithm to follow

in this case is simple. Firstly, we start by casting two rays towards the scenario through the two bottom corners of the bounding box. The two collisions give us the location of the object within the scenario and the projected width. Secondly, we create a quad centered between the two collisions and oriented accordingly. The width of the box is slightly increased and the height is given a high value to avoid errors in the following step. The third step consists in casting two more rays through the two top corners of the bounding box. These collisions indicate the height of the detected object, which is calculated as the average of the values obtained from each collision. Finally, we must infer a value for the length of the cuboid, as this third dimension of the object cannot be assessed from the bounding box. The length can be fixed for every object or proportional to its width and/or height.

3.2 Using the Silhouette to Model the Object

Although the approach presented in the previous section was suitable for all cases, the algorithm proposed in this section is only adequate for objects oriented with respect to the camera in such a way that perspective information is available. Using the notation presented in Fig. 2, when the casted ray towards vertex v_1 is parallel to the edge $(v_2,\ v_1)$ and perpendicular to the edge $(v_5,\ v_1)$ (or vice-versa), it is not possible to infer further details on the cuboid as we lack information on the width or the length of the object.

The idea of our second augmentation algorithm is to select the silhouette points that can correspond to the vertices of the cuboid:

- from the points located near the left side of the silhouette, select the one at the bottom to represent v_2
- from the points located near the right side of the silhouette, select the one at the bottom to represent v_5
- from the points located near the bottom side of the silhouette, select the one at the center to represent v_1
- from the points located near the left side of the silhouette, select the one at the top to represent v_3
- from the points located near the right side of the silhouette, select the one at the top to represent v_8

The first three points are projected on the 3D scenario and then used to calculate the position (we consider v_1 to be the location), the width and the length of the cuboid. The only dimension to find out is the height. We create two cuboids, one located at edge $(v_2,\ v_6)$ and the other one on edge $(v_5,\ v_6)$. We project the two latter points (v_3 and v_8) and calculate the height as the mean of the values obtained from these two collisions.

Table 1 Results obtained when varying the orientation or the location

Angle(°)	Real Position (x,y)	Calculated position (x,y)	Width	Height	Length
15	50, 25	50.45, 27.23	2.95	4.22	14.07
30	50, 25	51.22, 26.59	3.39	4.09	13.92
45	50, 25	51.02, 25.97	3.86	4.33	13.52
60	50, 25	51.89, 24.45	3.49	4.10	12.93
75	50, 25	49.03, 26.33	3.75	3.78	11.75
45	50, 25	50.46, 24.77	3.81	4.15	13.84
45	50, 40	50.72, 36.98	3.75	3.75	13.72
45	80, 25	82.86, 23.24	4.54	4.14	12.52
45	80, 40	83.72, 39.15	3.27	4.15	12.72

3.3 Solving Ambiguities

Our framework is capable of working with complex environments with objects piled on others. A problem can appear when parts of an object are hanging off, as we could obtain erroneous values when projecting the corners of the bounding box or the silhouette points. One of the advantages of working with OpenSceneGraph is the fact that the engine is capable of indicating the object that the ray collides to. With this information, it is easy to retrieve the height of the different objects the rays collide and select the highest one (provided the collision happens on top of the objects; if not, errors may be introduced). This height is then used to locate an imaginary plane where to perform the calculations of the location and the shape of the object using the algorithms presented before.

4 Results

In this section we present some results that have been obtained using our augmented virtuality application, which has been developed using OpenSceneGraph, which is written entirely in C++ and OpenGL.

We want to start by testing how the accuracy of the detection depends on the orientation of the object with respect to the camera. Table 1 presents the results obtained when detecting a stapler which measures $3.67 \times 4.55 \times 14.4$ centimeters (width $\times$ height $\times$ length) using the silhouette algorithm. In this table we vary the location and the orientation, being the latter referred to the angle between the ray that goes through v_1 and the edge (v_1, v_5).

Firstly, in the results depicted in the first half of Table 1 we vary the angle, and we can see how the lower the angle is, the higher the accuracy for calculating the length, as we have less perspective information. Similarly, the higher the angle is, the higher the precision for calculating the width. It is worth mentioning that, angles 0° and 90° have not been considered as the silhouette algorithm cannot be applied. Secondly, the results of the second half test the accuracy when the staple is always oriented at approximately 45°. These results prove how the calculations

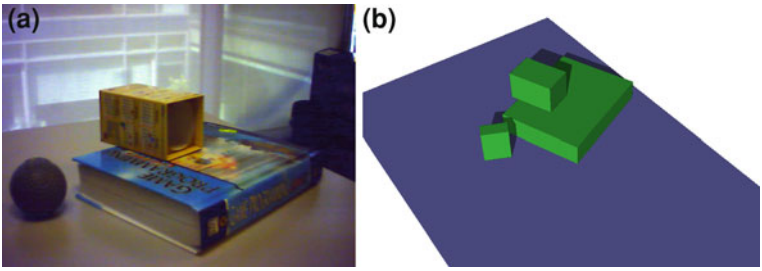

Fig. 3 Synthetic environment obtained from a real scenario. **a** Scenario with three objects **b** Image from a different perspective

are more precise when the object is closer to the camera, as the algorithm has more information on the silhouette.

From a different perspective, Fig. 3 offers a more complex scenario, where the box and the book are modeled with the silhouette algorithm, while the ball is modeled using the bounding box as no perspective information can be obtained from its silhouette. Finally, we must underline that the box has been located in such a way that some points of the silhouette hang off the book. In this case, as some rays collide on the book and some on the floor plane, the algorithm selects the height of the book to locate a plane with which continues the modeling process.

5 Conclusions

In this paper we have presented an augmented virtuality framework based on a smart camera. We have shown how our approach is capable of locating the objects on the scenario with high accuracy. This paper offers important improvements over previous solutions, as it uses the silhouette to better adjust the shape of the objects. Once the synthetic scene is populated, the user can modify the camera position and orientation as desired, obtaining new perspectives that can be useful for surveillance tasks.

As future work, our main line for future work implies extending the vision algorithms of the SENSE camera to offer features like classification, speed or color to improve the quality of the simulation performed in the graphics application. For example, having information on trajectories can enable prediction if there were delays in the visual feedback loop. From a different perspective, we must say that the use of a multi-camera framework would imply altering our algorithms but would definitely improve our simulation.

Acknowledgments This work has been funded by the Spanish Government (TIN2009-14103-C03-03, TSI-020400-2009-0133, DPI2008-06737-C02-02 and DPI2011-28507-C02-02) and by the European Union (ITEA2 IP08009).

References

1. Benet, G., Simo, J.E., Andreu, G., Rosell, J., Sanchez, J.: Embedded low-level video processing for surveillance purposes. In: 3rd International Conference on Human System Interaction (HSI) (2010)
2. Sebe, I., Hu, J., You, S., Neumann, U.: 3D video surveillance with augmented virtual environments. In: ACM Workshop on Video Surveillance, pp. 107–112 (2003)
3. Rieffel, E., Girgensohn, A., Kimber, D., Chen, T., Liu, Q.: Geometric tools for multicamera surveillance systems. In: Conference on Distributed Smart Cameras (2007)
4. de Haan, G., Scheuer, J., de Vries, R., Post, F.: Egocentric navigation for video surveillance in 3D virtual environments. In: IEEE workshop on 3D User Interfaces, pp. 103–110 (2009)
5. Fleck, S., Busch, F., Biber, P., Strasser, W.: 3D surveillance a distributed network of smart cameras for real-time tracking and its visualization in 3D. In: Conference on Computer Vision and Pattern Recognition Workshop (CVPRW06), p. 118 (2006)
6. Ziga, M., Brmond, F., Thonnat, M.: Fast and reliable object classification in video based on a 3D generic model. In: Proceedings of the International Conference on Visual Information Engineering (VIE2006), pp. 26–28 (2006)
7. Osawa, T., Wu, X., Wakabayashi, K., Koike, H.: 3D human tracking for visual monitoring. NTT. Tech. Rev. **5**, 1–8 (2007)
8. Osfield, R., Burns, D.: OpenSceneGraph. http://www.openscenegraph.org (2011)