Synthesizing Reality for Realistic Physical Behavior of Virtual Objects in Augmented Reality Applications for Smart-Phones

Nir Amar, Meir Raviv, Barak David, Oleg Chernoguz, and Jihad El-Sana

Computer Science, Ben-Gurion University of the Negev, Isreal

ABSTRACT

This paper presents a framework for augmented reality applications that runs on smart mobile phones and enables realistic physical behavior of the virtual objects in the real-world. The used mobile phone is equipped with two cameras and provides stereo images and live video. The two images are used to reconstruct a 3D representation of the real world, which is good enough to enable physical interaction between the virtual and the real-world objects, but not fine enough to synthesize the real-world view for back projection. The visual synthesis of the real-world view is done through the video stream. The constructed 3D representation is registered in the real-world view and used to place the virtual objects, determine their physical behavior, and detect collision with objects in the real-world view. The 3D reconstruction is not performed at each frame, but applied when necessary based on the position of the dynamic objects. Pose estimation is determined based on the movement of the mobile phone (gyroscope and accelerometer) and the viewed images. A physics engine, which utilizes the gravity vector obtained from the accelerometer sensor of the mobile device, is integrated into our framework. The physics engine equips virtual objects with realistic physical behavior.

Index Terms: I.2.10 [Computing Methodologies]: Vision and Scene Understanding—3D/stereo scene analysis

1 INTRODUCTION

Augmented Reality technologies enhance user perception by supplementing the real world with virtual content. This is usually performed by generating a virtual view that includes real-word's objects and virtual objects. Generating such a view usually involves synthesizing the real world, which is done by reconstructing the 3D representation of the real world or using the video stream directly without estimating the scene structure, and embedding the virtual objects into the synthesized scene. Generating a full 3D representation of the real word simplifies inserting virtual objects into the scene. However, reconstructing a full 3D model from 2D information at interactive rates is a challenging task. Avoid estimating scene structure complicates the placement of virtual objects in the scene and requires dense correspondences between views, which are expensive to compute. Inserting a virtual object into a video stream is a fundamental task in augmented reality applications. It requires knowledge about the structure of the real-world view and the layout of its objects. The problem becomes more sever when the virtual objects are dynamic; i.e, they move and interact with the real world objects.

Many algorithms have been developed to reconstruct a 3D structure from stereo images. These algorithms usually establish correspondence between points in the two images, and then match these points to estimate their depth [5]. In this work we utilize this knowledge and optimize it toward running on mobile phones. Recently, approaches try to extract features from the viewed scene for registration and pose estimation [7, 4, 10]. We employ the extracted features for 3D reconstruction and pose estimation.

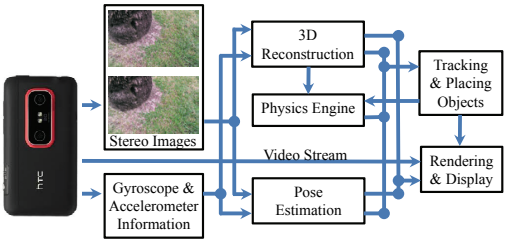

Figure 1: The components of the framework and its flow control

The introduction of commodity depth cameras, such as the Microsoft Kinect, enable acquiring depth images at interactive frame rates and real-time 3D reconstruction of the viewed scene [8]. However, most of these technologies are not yet available on mobile devices and/or limited for indoor usage. Several approaches use depth sensing to place objects in tabletop [12] or into one's palm [11]. Depth maps are also used for automatic object placement, based on detecting collisions between real and virtual objects [3]. In this work we perform the entire flow on a mobile phone in an indoor as well as outdoor environments.

This paper presents a framework for augmenting a real world environment with dynamic virtual objects on mobile phones. The framework is based on computing a 3D representation of the real world, detailed enough for placing the dynamic objects on the real world, determining collision between virtual objects and real-world objects, and applying physical interaction. The computed 3D representation and the virtual objects are registered on the video stream, which is used to synthesize the real-world. The 3D reconstruction is not performed at each frame, but applied when necessary based on the position of the dynamic objects. We integrate a physics engine into our framework and utilize the gravity vector obtained from the accelerometer sensor of the mobile device. Figure 1 presents the various components of the framework and the control flow.

Camera pose estimation obtains the relative camera-to-object position and orientation. Typical augmented reality applications estimate camera pose using marker-based and motion-based approaches. Marker-based approaches utilize a simple black and white marker for easy detection and tracking of the marker area [7]. Motion-based approaches track many feature points and recover camera motion parameters from the point correspondences [9]. We compute camera pose estimation using the device sensors (gyroscope and, accelerometer) and feature correspondence between consecutive images, without using model geometry.

2 3D RECONSTRUCTION

To construct the 3D representation of the viewed scene from stereo images, we extract sparse feature points from the two images, match up corresponding feature points in both images, and apply triangulation to determine the depth of the corresponding points.

We adopt SURF [2] to extract sparse feature points and its association method to assign scores to pairs of points according to the Euclidean squared distance between the descriptor vectors. The sign of the Laplacian is used for fast indexing and the horizontal alignment of the two images is taken into account to reduces the search space.

We assume the parameters of the camera and their configuration does not change over a work-session. We calibrate the stereo cameras, and compute the principal point and focal length. We then estimate the transformation from the left to the right camera, and use triangle similarity to compute the depth of the feature points. The obtained set of points forms a coarse sampling of the viewed scene. The number of points depends on the resolution of the acquired images and the parameters of the SURF algorithm.

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

Figure 2: 3D reconstruction: (a) left image, (b) right image, (c) computed points, and (d) final points.

Filtering Points: The accuracy and robustness of the obtained points vary and depend on their neighborhood and the intensiveness of the search algorithm. To build a more accurate and reliable model of the real world, we filter unreliable points using vertical disparity. The validity of a disparity value is estimated based on the distribution and disparity of neighboring points. We rely on the SURF [2] separation radius, R_{surf}, to determine the size of the neighborhood of a point. In our current implementation we use a neighborhood of size $4R_{surf}$. A point, which has less than two adjacent points within its neighborhood, is considered an outlier and filtered out. Filtering out unreliable points downsizes half of the point set.

Point Refinement: The remaining points are distributed irregularly over the viewed scene and their number is too small to produce an acceptable mesh. To overcome this limitation we refine the set of points to increase their number and simplify the generation of the mesh. The refinement is performed by imposing a uniform grid $m \times n$ on the xy-plane and assign a vertex to the center of each cell. The depth of each vertex is computed by bilinear interpolation of its adjacent points (input points only). Figure 2 illustrates the stages of the 3D reconstruction. In our current implementation we have found that a grid of size 20×20 leads to the generation of acceptable representation at reasonable time. The refinement procedure produces regularly distributed vertices, which simplifies the mesh generation. The generated mesh is used to place and align virtual objects, and detect collision between the virtual objects and the real word.

3 OBJECT PLACEMENT

Many augmented reality applications aim to integrate the virtual objects within the real world in a seamless manner. These applications are required to address the visual and behavior challenges of the objects on the viewed scene. The advances in graphics hardware have almost overcome the rendering limitation. The basic behaviors, include physical behavior and interaction, are still challenging tasks. This work aims to explore the interaction between the virtual objects and the real-world objects. Reconstructing a digital representation of the real-world brings its objects into the virtual world, where it is possible to compute and regulate these interactions. In our framework the video stream is used to view the real world and the generated mesh, which is aligned (registered) with the video stream, is used for collision detection and physical behavior. For that purpose, we have integrated a physics engine, JBullet [1], into our framework. The physics engine utilizes the gravity vector obtained from the accelerometer sensor of the mobile device to enable accurate placement of the virtual objects and allow them to move and fall in a realistic manner, similar to [6]. The physics engine regulates the interaction between the real-word (represented by the generated mesh) and the virtual objects. In each iteration the physical simulation computes the new position of the dynamic virtual objects and detects collisions. The new positions are used to draw the virtual objects at their new locations, see Figure 1. Parameters from camera calibration phase, such as focal length and aperture, and the device position and orientation are used to determine the view parameters for rendering the virtual objects.

4 EXPERIMENTAL RESULTS

We benchmarked and tested the framework on an HTC EVO 3D mobile phone with dual-core 1.2 GHz CPU and Adreno 220 GPU. It has 1 GB RAM local memory and runs Android OS. We tested our framework on various scenes – natural and artificial – with different complexities and measured the time required for each phase. We have found that the detection, description, and matching phases consume 40%, 20%, 20% of the construction time, and the size of the obtained point set depends on to scene complexity. We designed a simple game to experimented with our framework and demonstrate interactive interactions with the real-world in a simple and intuitive manner. The game simulates a motion of a vehicle on different real world views, and runs at a 20-40 frames per second.

5 CONCLUSION

We have presented a framework for realistic physical behavior of the virtual objects in the real-world in augmented reality applications. It runs on mobile phones equipped with two cameras and provides stereo images and live video. A 3D representation of the real world is reconstructed and used to guide the physical interaction between the virtual and the real-world objects.

REFERENCES

[1] Jbullet. http://jbullet.advel.cz/, 2012.

[2] H. Bay, A. Ess, T. Tuytelaars, and L. Van Gool. Speeded-up robust features. *Comput. Vis. Image Underst.*, 110(3):346–359, June 2008.

[3] D. Breen, R. Whitaker, E. Rose, and M. Tuceryan. Interactive occlusion and automatic object placement for augmented reality. *Computer Graphics Forum*, 15(3):11–22, 1996.

[4] N. Hagbi, O. Bergig, J. El-Sana, and M. Billinghurst. Shape recognition and pose estimation for mobile augmented reality. *IEEE Trans. on Vis. and Comp. Graphics*, 17(10):1369–1379, 2011.

[5] R. I. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, 2nd edition, 2004.

[6] J. Laviole, M. Hachet, and C. Schlick. Tabletop games using real environment and physical simulation. In *The Foundations of Digital Games*, Bordeaux, France, 2011.

[7] V. Lepetit and P. Fua. Monocular model-based 3d tracking of rigid objects. *Found. Trends. Comput. Graph. Vis.*, 1(1):1–89, Jan. 2005.

[8] R. A. Newcombe, S. Izadi, O. Hilliges, D. Molyneaux, D. Kim, A. J. Davison, P. Kohli, J. Shotton, S. Hodges, and A. Fitzgibbon. Kinectfusion: Real-time dense surface mapping and tracking. In *ISMAR '11*, pages 127–136, 2011.

[9] J. S. Park, M. Sung, and S. R. Noh. Virtual object placement in video for augmented reality. In *Advances in Multimedia Information Processing*, volume 3767, pages 13–24. Springer, 2005.

[10] H. Uchiyama and E. Marchand. Toward augmenting everything: Detecting and tracking geometrical features on planar objects. In *ISMAR '11*, pages 17–25, 2011.

[11] A. Wilson and H. Benko. Combining multiple depth cameras and projectors for interactions on, above and between surfaces. In *symposium on user interface software and technology*, pages 273–282, 2010.

[12] A. D. Wilson. Depth-sensing video cameras for 3d tangible tabletop interaction. In *Tabletop'07*, pages 201–204, 2007.