Augmented Reality using Over-Segmentation

Visesh Chari, Jag Mohan Singh and P. J. Narayanan
Center for Visual Information Technology,
International Institute of Information Technology,
Gachibowli, Hyderabad 500032 Email:{visesh,jagmohan}@research.iiit.ac.in, pjn@iiit.ac.in

***Abstract*—Augmenting virtual objects between real objects of a video poses a challenging problem, primarily because it necessitates accurate knowledge of the scene's depth. In this paper, we propose an approach that generates an approximate depth for every pixel in the vicinity of the virtual object, which we show is enough to decide the ordering of objects in every image. We also draw a similarity between layered segmentation and augmentation. We argue that augmentation only needs to know which layers are in front of the object and which are behind it. Our algorithm makes effective use of object boundaries detected in an image using image segmentation. The assumption that these boundaries correspond to depth discontinuities provides a useful simplification of the problem, sufficient to produce realistic results. We use a combination of segmentation and feature detection for sparse depth assignment. Results on challenging data sets show the effectiveness of our approach.**

I. Introduction

Augmented Reality refers to the class of techniques that augment videos of real scenes with virtual objects. Such solutions find a wide variety of uses [1]. The objective is to place the virtual objects in accordance with the physical attributes of the real world being captured in the video. The two basic principles involved in this process are

- Identifying where the virtual object should be placed in every frame of the video.
- Identifying which object(s) in the real video occlude/are occluded by the virtual object in every frame of the video.

The first part of the problem requires the estimation of 3D camera positions, to decide where the object is placed with respect to each of the cameras. The second part of the problem requires 3D knowledge of the scene to decide how the real object(s) is placed with respect to the virtual object(s). The various scenarios in which augmented reality is needed has given rise to two types of techniques: marker-based and marker-less augmented reality. Our work falls in the later category.

Traditionally, Augmented Reality used markers to augment virtual objects in real scenes. These markers could be easily detected in different images and the camera matrix can be recovered from it. The virtual object could be placed only in front of the real image, but with correct projection. In contrast, marker-less augmented reality advocated careful estimation of parameters like camera motion and scene structure from image features alone. The camera parameters alone will suffice if the virtual objects are to be placed in front of the real ones. Scene structure or depth is required if real objects can obscure parts of the virtual object. However, this requires considerable effort as computing these parameters are complex for a scene [2]. Thus an augmentation like Figure 1, would require accurate estimation of the depths of all the points on the pillar as well as the bikes behind the pillar. However, we would like to argue that augmented reality is a problem similar to layered segmentation. Layered segmentation divides a scene in to layers. Given that in most cases, a discontinuity in depth is accompanied by a change in color, we argue that a layer always consists of pixels belonging to the same object, with similar depth values. Thus for correct augmentation, we only need to know which layers are in front of the virtual object and which are behind it. Thus, we need not compute dense depths, but only an *approximate estimation of layers* in the scene is enough. In this paper, we use a potent combination of image segmentation and SIFT features to estimate the approximate depth of layers. The depth of each segment in the image is the average of all the features in it.

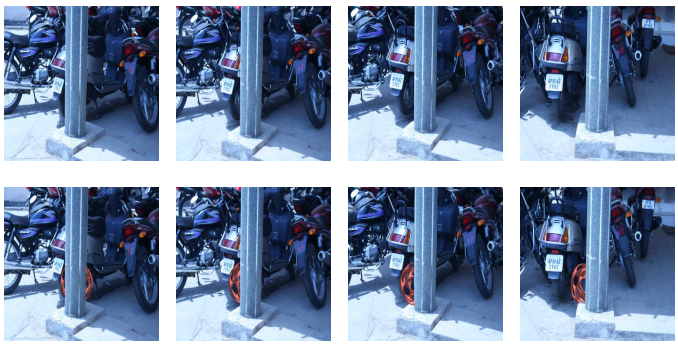

Fig. 1: Top Row: It shows the scene without the virtual object. Bottom Row: It shows the scene with the virtual object lying in-between the pillars and bikes, and it moves accurately with the camera

Section II presents the various works that bear significant relation to ours. Section III details our particular solution. Section IV shows results on some sequences, highlighting the challenging contributions. Finally, Section V discusses future extensions, and presents some concluding remarks.

II. Related Work

Traditional Augmented Reality techniques use markers and place virtual objects in front of real ones using them. The camera needs to be recovered so that the virtual object moves correctly as the camera moves. This is done by using markers

or tie points, which can be identified both in the reference image and the new images. The positions of these point are easily predicted in the new image [3]. However, virtual object can be placed only with respect to the markers in front [4]. Markerless Augmented Reality with real-time affine region tracker, divides the image into locally affine invariant regions and finds correspondences between the views [5]. It then places animated textures in the desired region. In order to place the virtual object anywhere in the environment, one needs to recover structure of the scene. This approach is however very expensive as in order to compute structure of the scene, camera needs to be calibrated and structure needs to recovered. This is done by using feature tracker to establish correspondences followed by self-calibration of the camera. A bundle adjustment step which refines the estimated camera and structure is performed normally. Plausible Physics in Augmented Images [6] first computes SIFT features in the unordered images. This is followed by bundle adjustment to get $3D$ positions and camera parameters. Then the augmented object can be placed anywhere in the scene. This approach requires scenes which have enough features, since the correspondences computed are normally not dense. The depth map computed from this approach is sparse and is interpolated using thin-plate splines. Kanade et al. computed the depth map at video rate using stereo machine [7] and showed z-keying which was same as merging of virtual and real world in real time. One can use a model based region-tracker and estimate structure for that model then the problem can be reduced to that of pose computation of the object in the scene [8]. This approach can place the virtual object both in front or back of tracked region in the real scene, however it is not possible to define models for all the objects in the scene.

Over-segmentation algorithms [9] have recently gained popularity in geometry based works as an important pre-processing step. Automatic Photo popup [10] uses over-segmentation for single view reconstruction. The main idea is to first cluster an image into different "super-pixels" and then to determine the "label" for each super-pixel, which represents its geometry. Another area where over-segmentation has been extensively used is stereo [11]. In case of stereo algorithms, over-segmentation is first used to obtain a coarse disparity map / depth map for a scene, which may be further refined to get pixel level correspondences. Other areas where over-segmentation finds use is image based rendering [12], structure from motion [13] and layer segmentation based stereo [14]. Normalized cut treats image segmentation as a graph partitioning problem and uses the normalized cut for segmenting the graph. The normalized cut measure increases total dissimilarity between different groups along with total similarity within the groups [15].

III. Approach

The outline of our approach is illustrated in Figure 2. We use over-segmentation [9] to obtain segments that typically belong to one object, the main assumption being that depth variation is insignificant across a segmented patch. Estimates of camera motion and feature correspondences are then used to determine the depth of every over-segmented patch. Finally, the synthetic object is rendered with the help of these estimated depths.

a) **Camera estimation::** Any Augmented Reality system requires accurate calibration of the camera, else the error would show up as a "jitter" in the augmented object. For camera motion estimation, we use the method by Pollefeys et al. [16], because of its robustness to feature correspondence errors and the two-view intiialization to the reconstruction process.

Firstly, the essential matrix is computed using feature correspondences and a RANSAC based algorithm by solving the following equations

$$x^\top \mathcal{F} x = 0 \quad (1)$$

$$\mathcal{E} = K^\top \mathcal{F} K \quad (2)$$

where $\mathcal{F}$ is the Fundamental matrix, $\mathcal{E}$ is the Essential matrix and K is the internal camera calibration. Next, we compute the camera matrix by decomposing the Essential matrix, using `svd` [2]. Once parameters of all key frames are computed, optimization over the camera parameters for all views is performed.

$$\mathcal{E} = R \times t \quad (3)$$

$$P = \begin{bmatrix} R & t \end{bmatrix} \quad (4)$$

$$\begin{bmatrix} P & X \end{bmatrix} = \arg\min_P (PX - x)^2 \quad (5)$$

where x and X are the 2D image feature and its corresponding 3D point, respectively. Finally, we only compute camera matrices for keyframes and then interpolate to obtain the intermediate frames.

$$R = R_1 * \alpha + R_2 * (1 - \alpha)$$

$$t = t_1 * \alpha + t_2 * (1 - \alpha)$$

where R is not represented as a matrix, but as a 3-vector using the exponential mapping.

b) **Region selection::** Correct augmentation only requires the depths only in the region around the object. The region can be identified by it's bounding box which is by projecting the virtual object's 3D model onto the image after recovering the cameras. A region of interest around the object is identified for each image, and segments overlapping this area are considered for depth recovery and the final rendering of the object. Subsequent steps are done only in the region of interest.

c) **Over-segmentation::** Over-segmentation algorithms produce a segmentation that obeys the color boundaries of the scene. This means that each segment belongs to one object in the scene, although many segments might belong to the same object. We use the algorithm of [9], where a graph based approach is proposed. The nodes represent the various layers in the image and edges represent spatial neighborhood information. The weights of each edge represent the similarity of color / texture / edge information between the two nodes

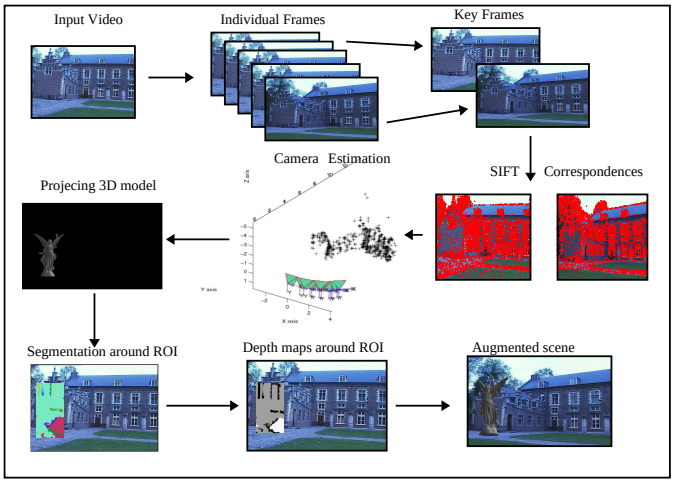

Fig. 2: Overall algorithm. Input images were taken from the Leuven Castle dataset (http://www.cs.unc.edu/∼marc)

connected by an edge. Initially, each pixel corresponds to a node. Edges are collapsed when their weights are low (below a threshold), merging the clusters connected by them into one. Since the threshold for segmentation is a parameter to the algorithm, different segmentations result from different values of this parameter.The over-segmentation needed by our algorithm will vary from scene to scene. In scenes with less texture, it is desirable to have large segments since finding point correspondences between segments recovered from two images becomes a harder problem. On the other hand, its damaging to have the segmentation cross object boundaries. In scenes with sufficient texture information, small segments are preferable. Setting the threshold to such a low value might, however, cause noise in the image to be highlighted. Thus, although we choose the threshold value manually, automatically determining its value is of importance.

d) **Growing Correspondences::** Feature correspondence algorithms like SIFT [17], give a sparse set of accurate correspondences even across widely separated views. Such features are thus extremely useful for robust camera motion estimation. However, the sparseness of these correspondences makes it likely for a segment to not have any feature point within it. Since depths need to be assigned to every segment, at least one feature correspondence needs to be computed per segment. We follow a slightly modified version of the approach presented in [18], and "grow" correspondence around SIFT features. Initially, Harris points are detected in the vicinity of every SIFT feature. These points are then "transferred" to every other image, in a manner that preserves their distance from the corresponding SIFT feature in every image. Once transferred, a correspondence for every Harris point is searched in the surrounding texture patch by maximizing the normalized cross correlation function 6.

Since we detect Harris points and use normalized cross correlation for matching, spurious matches may be detected at edge boundaries with insufficient texture information due to the aperture problem. Also, such correspondences are much less accurate than their SIFT counterparts. The correspondences are refined by fitting a quantity like the Fundamental Matrix, after removing outliers [19]. Figure 3 shows depth maps computed with and without growing correspondences.

(a) SIFT (b) SIFT+Harris

Fig. 3: (a) Correspondence estimation with SIFT [17] (red points). (b) After growing correspondences using Harris points [18] (blue points). Notice how the textured part of the scene has been nicely sampled.

$$\gamma(u,v) = \frac{\Sigma_{x,y}\left[I_1(x,y)-\bar{I_1}\right]\left[I_2(x-u,y-v)-\bar{I_2}\right]}{\sqrt{\left\{\Sigma_{x,y}\left[I_1(x,y)-\bar{I_1}\right]^2\Sigma_{x,y}\left[I_2(x-u,y-v)-\bar{I_2}\right]^2\right\}}}$$

e) **Assigning Depths::** Since the algorithm for growing correspondences works well for textured scenes, areas in an image without texture are problematic. At this point, one can choose to incorporate algorithms from the stereo literature [14], [11], in order to assign depths to such pixels. We need to know depth labeling only in region which lie inside the bounding box of the virtual object. This label indicates whether the segment is in front or behind the virtual object. In order to assign these labels we need depth of these segments, if they already have depth values then it is used. If the segment does not have depth value we assign it a depth value by interpolating it from it's neighbours and then assign a depth label to it. The depth labeling for segments lying outside the bounding box of the virtual object is done in the following manner it is maximum for regions lying beyond the virtual object and minimum for regions lying nearer.

f) **Rendering::** We have depth labeling for each segment inside the bounding box of the virtual object at a finer level and coarse depth labeling for other regions. We copy the image and depth to the frame buffer first. Then render the virtual object with lighting.

IV. Results

The virtual object is interactively placed in the real sequence. This is done for the first frame of the video. The bounding box of the virtual object is computed and used for post-processing. Alternatively, the user can specify the coordinates of the bounding box of the object which is used for augmentation.

The first scenario is the famous "Leuven castle" sequence. It consists of 12 cameras. The structure of the same is recovered and there are no occluding parts in the same. Figure 4 shows the scene with and without the virtual object. The computation

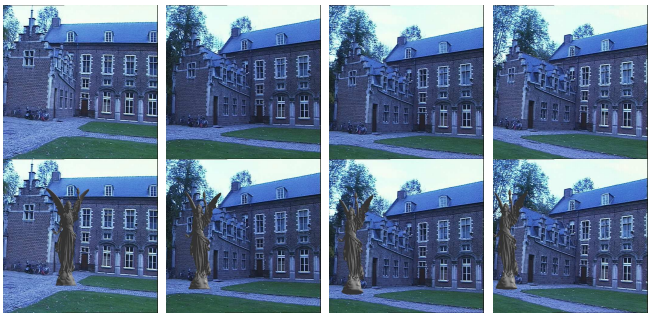

Fig. 4: Top Row: It shows the Leueven Castle sequence without the virtual object. Bottom Row: It shows the scene with the virtual object placed in the scene in front of the castle and moves accurately with the camera

Fig. 5: Top Row: It shows the sequence from Stanford Light field dataset without the virtual object. Bottom Row: It shows the virtual object placed in between the plants and other objects. Note the accurate placement of the virtual object between the CD and the poster and it is occluded by leaves in the front.

time for the given sequence is around 50 minutes for camera-generation and segmentation and depth-labeling takes around 10 minutes.

The second scenario shows results on a sequence taken from Stanford Light Field Database and shows several occluding objects. The sequence consists of 104 scenes and the structure for same is recovered nicely. Figure 5 shows the virtual object is placed between the real objects. The computation time for this sequence is less as we have the camera-matrices for the same and it is 2 minutes for depth labeling generation given the segmented images. The time for segmentation is around 5 minutes.

The third scenario shows results on a sequence taken from an outdoor environment and shows several cluttered objects with occlusions. A virtual object is placed hugging the pillar in the front (Figure 1). This sequence now consists of three layers, the object's exterior, the pillar and the bikes placed behind them. The figures show correct placement of the object. The time for 20 minutes for camera-generation and 5 minutes for segmentation and depth labeling generation.

V. Discussion

In this paper, we presented an approach to augmented reality that relies on over-segmentation of images. The main insight is that augmented reality requires only nearly accurate depths and that over-segmentation algorithms produce segments that obey object boundaries and so are suitable for reconstruction. Results show that our approach can handle complex scenarios with visually acceptable results. Extensions to this approach would include a closer inspection at the process of segmentation and correspondence estimation for texture-less areas, and algorithms to automatically determine the optimal segmentation of a scene.

References

[1] O. Bimber and R. Raskar, *Spatial Augmented Reality, Merging Real and Virtual Worlds*. A K Peters LTD, 2005.

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

[3] K. N. Kutulakos and J. R. Vallino, "Calibration-free augmented reality," *IEEE Transactions on Visualization and Computer Graphics*, vol. 4, no. 1, pp. 1–20, 1998.

[4] M. Billinghurst, H. Kato, and I. Poupyrev, "The magicbook: A transitional ar interface," *Computers and Graphics*, pp. 745–753, November 2001.

[5] V. Ferrari, T. Tuytelaars, and L. V. Gool, "Markerless augmented reality with a real-time affine region tracker," *Procs. of the IEEE and ACM Intl. Symposium on Augmented Reality*, vol. I, pp. 87–96, October 2001.

[6] M. Leotta and K. Boyle, "Plausible physics in augmented images," *SIGGRAPH '05 : Poster*, 2005.

[7] T. Kanade, A. Yoshida, K. Oda, H. Kano, and M. Tanaka, "A stereo machine for video-rate dense depth mapping and its new applications," *Proceedings of the 15th Computer Vision and Pattern Recognition Conference (CVPR '96)*, pp. 196–202, June 1996.

[8] A. I. Comport, E. Marchand, and F. Chaumette, "A real-time tracker for markerless augmented reality," *ISMAR '03: Proceedings of the The 2nd IEEE and ACM International Symposium on Mixed and Augmented Reality*, p. 36, 2003.

[9] P. F. Felzenszwalb and D. P. Huttenlocher, "Efficient graph-based image segmentation," *International Journal of Computer Vision*, vol. 59, no. 2, September 2004.

[10] D. Hoiem, A. A. Efros, and M. Hebert, "Geometric context from a single image," *International Conference of Computer Vision (ICCV)*, vol. 1, pp. 654 – 661, October 2005.

[11] M. Bleyer and M. Gelautz, "Graph-cut-based stereo matching using image segmentation with symmetrical treatment of occlusions," *Image Commun.*, vol. 22, no. 2, pp. 127–143, 2007.

[12] C. L. Zitnick and S. B. Kang, "Stereo for image-based rendering using image over-segmentation," *International Journal of Computer Vision*, 2007.

[13] F. Ernst, P. Wilinski, and K. van Overveld, "Dense structure from motion: An approach based on segment matching," *ECCV '02. 2002 European Conference on Computer Vision*, pp. 552–554, 2002.

[14] M. Bleyer and M. Gelautz, "A layered stereo algorithm using image segmentation and global visibility constraints," *ICIP '04. 2004 International Conference on Image Processing*, vol. 5, pp. 2997–3000, 2004.

[15] J. Shi and J. Malik, "Normalized cuts and image segmentation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 22, no. 8, pp. 888–905, 2000.

[16] M. Pollefeys, L. V. Gool, M. Vergauwen, F. Verbiest, K. Cornelis, J. Tops, and R. Koch, "Visual modeling with a hand-held camera," *International Journal on Computer Vision*, pp. 207–232, 2004.

[17] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *International Journal of Computer Vision*, vol. 60, no. 2, pp. 91–110, 2004.

[18] M. Lhuillier, "A quasi-dense approach to surface reconstruction from uncalibrated images," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 27, no. 3, pp. 418–433, 2005.

[19] P. H. S. Torr, A. W. Fitzgibbon, and A. Zisserman, "Robust computation and parametrization of multiview relations," *ICCV '98. 1998 International Conference on Computer Vision*, pp. 485–491, 1998.