2D/3D Augmented Reality

Kamran Ali (2013-10-0174), Syed Bilal Ali (2013-10-0028)

December 10, 2012

Abstract

The goal of this project is to overlay 2D/3D objects onto real world video sequences. User can define the placement of the object which is then done according to the perspective of the real scene. The augmentation is done using both manual and self calibration techniques (without any information about camera parameters, world co-ordinates or reconstructed scene geometry). The key points i.e. edges and corners are tracked between the sequence of images using SIFT[1] for continuous augmentation in the video scenes. After tracking the features of the scenes, different techniques are implemented to calculate the 2D and 3D homographies between the scenes. These homographies are then used to continuously transform the 2D and 3D models respectively which are then augmented in the scenes. In the end, ***photometric*** *changes are done in the texture according to the conditions in the environment to improve the realistic look.*

1 Introduction

Augmented reality technology combines virtual reality with the real world. We find its applications in sightseeing and tourism industries, enhanced GPS systems, Heads-Up Displays (HUD's) in military for aiding both fighter pilots and ground troops, MRI and X-ray systems in medical applications, gaming industry, education, factories to help workers for their job via head-mounted displays, etc.

1.1 Problem Statement

In order to place 2D and 3D artificial objects into video sequence, the questions to be addressed are where and how the object should be placed. In this implementation, the former can be decided on by the user. But in order to actually place the object we must know the perspective in the real world scene. For this we have to find the *homographies* that exist between the artificial model of the object we have and the original scene. The real challenge is to find homography between the the real world coordinates and the virtual 3D objects in the context where don't know about either the camera positions and parameters or the geometry of the real scene.

Another major problem is of continuous *tracking* of some features of the image sequences as the computation of homography needs **corresponding points** in two images. The objects have to placed in the image sequence after learning the homography between the points or the features being tracked.

Figure 1: Pattern used for 2D and 3D augmentation

1.2 Our Goal

Once the user chooses the location of the virtual 2D or 3D object in the scene, we use **SIFT** algorithm to continuously track useful features of the image sequence. The provision for the user and the implementation of a technique for general videos was incorporated at the end of the project. We started off with a video which contained image sequences of a pattern shown in *Fig.1*. Then based on techniques which will be explained later, we approximated the *Homographies* which will be a *3 × 4* **camera matrix** and a *3 × 3* **projective matrix** in case of 3D and 2D augmentations respectively. After the augmentation of the respective 2D or 3D object we made **Photometric changes** in the texture according to the conditions in the environment in order to improve the realistic look[2].

2 Camera Model

We worked with the *perspective* or *pinhole* camera model (*Fig.2*), which is the most common model for video cameras. If we have a point **P=** [X, Y, Z] in 3D, then following *Perspective Transfrom* or *Projection* equation gives us the 2D projection of **P**:

$$\begin{pmatrix} hx \\ hy \\ h \end{pmatrix} = \begin{pmatrix} m_x & 0 & p_x \\ 0 & m_y & p_y \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} r_{11} & r_{12} & r_{13} & t_1 \\ r_{21} & r_{22} & r_{23} & t_2 \\ r_{31} & r_{32} & r_{33} & t_3 \end{pmatrix} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix}$$

The first two matrices on the right hand side can be written in short like this:

$$K[R \mid T]$$

where "K" is the 3x3 matrix of *Intrinsic* camera parameters which defines defines the transformation between camera space and image space, "R" is rotation needed to align camera to world axes and "T" is Translation needed to bring camera to world origin. $[R \mid T]$, a 3x4 matrix is called the matrix of *External* camera parameters which defines the transformation between world coordinates and camera coordinates.

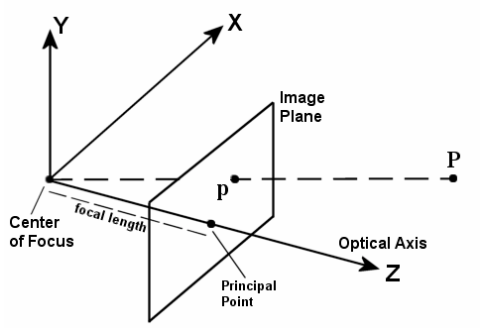

Figure 2: Perspective Camera Model[3]

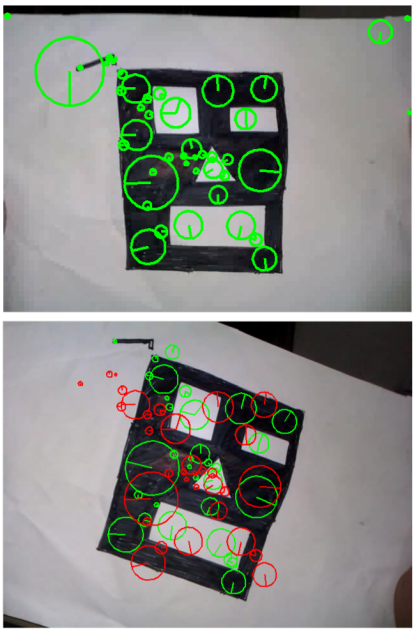

Figure 3: Tracking features of the pattern

3 Features Tracking using SIFT [1]

For tracking of key points on the planar pattern, a very robust technique called SIFT (*Scale Invariant Feature Tranform*) by David G. Lowe [1] was used which estracts distinctive invariant features from images that can be used to perform reliable matching between different views of an object or scene. "*The features are invariant to image scale and rotation, and provide robust matching across a substantial range of affine distortion, change in 3D viewpoint, addition of noise, and change in illumination.*" The results of tracking are shown in Fig.3, where the top image shows the initial image with it's features (green). The lower image shows only the matched features of both images; the features marked in *red* and *green* are the features of 1^{st} and 2^{nd} image respectively.

4 2D/3D Augmentation Approach

As we are doing pattern based augmented reality, our planar pattern defines a world co-ordinate frame into which virtual objects

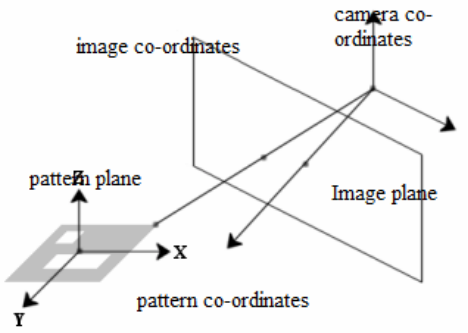

Figure 4: World coordinate system induced by a planar pattern[3]

are placed. It would be very convenient if this planar pattern is used itself for determining the camera matrix that which is needed to augment a 3D virtual object into images. So, there will be no need for a separate calibration to be done, simplifying the overall system for the user.

4.1 2D Augmentations

We know that two images of a plane from pinhole camera are related by *projective* transformation. Making the assumption the our planar pattern defines the $Z = 0$ of the world co-ordinate frame[3], the camera matrix simplifies the following way:

$$\begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} p_{11} & p_{12} & p_{13} & p_{14} \\ p_{21} & p_{12} & p_{13} & p_{24} \\ p_{31} & p_{32} & p_{33} & p_{34} \end{pmatrix} \begin{pmatrix} X \\ Y \\ 0 \\ 1 \end{pmatrix}$$

$$= \begin{pmatrix} p_{11} & p_{12} & p_{14} \\ p_{21} & p_{12} & p_{24} \\ p_{31} & p_{32} & p_{34} \end{pmatrix} \begin{pmatrix} X \\ Y \\ 1 \end{pmatrix} = H \begin{pmatrix} X \\ Y \\ 1 \end{pmatrix}$$

where p_{ij} are the elements of the perspective projection matrix of a pinhole camera and H is a 3×3 2D-2D homography. x_3 is just a scaling factor such that the image co-ordinated are:

$$\begin{pmatrix} x_i \\ y_i \\ 1 \end{pmatrix} = \lambda H \begin{pmatrix} X \\ Y \\ 1 \end{pmatrix}$$

4.1.1 *Solving for H*

H has apparently 9 unknowns. But it can be noted that H can be changed by multiplying by an arbitrary non-zero constant without altering the projective transformation. Thus H is considered a homogeneous matrix and only has 8 degrees of freedom even though it contains 9 elements. This means there are 8 unknowns that need to be solved for. We used The Direct Linear Transform (DLT) algorithm as described in [4] to solve for the homography matrix H given a sufficient set of point correspondences.

If $\mathbf{h} = [h_{11}h_{12}h_{13}h_{21}h_{22}h_{23}h_{31}h_{32}h_{33}]^T$ is a 9-element vector containing the elements of H. For atleast four collinear point correpondences we can solve for elements of H as follows:

$$\begin{pmatrix} x_1 & y_1 & 1 & 0 & 0 & 0 & -x_1'x_1 & -x_1'y_1 & -x_1' \\ 0 & 0 & 0 & x_1 & y_1 & 1 & -y_1'x_1 & -y_1'y_1 & -y_1' \\ x_2 & y_2 & 1 & 0 & 0 & 0 & -x_2'x_2 & -x_2'y_2 & -x_2' \\ 0 & 0 & 0 & x_2 & y_2 & 1 & -y_2'x_2 & -y_2'y_2 & -y_2' \\ x_3 & y_3 & 1 & 0 & 0 & 0 & -x_3'x_3 & -x_3'y_3 & -x_3' \\ 0 & 0 & 0 & x_3 & y_3 & 1 & -y_3'x_3 & -y_3'y_3 & -y_3' \\ x_4 & y_4 & 1 & 0 & 0 & 0 & -x_4'x_4 & -x_4'y_4 & -x_4' \\ 0 & 0 & 0 & x_4 & y_4 & 1 & -y_4'x_4 & -y_4'y_4 & -y_4' \end{pmatrix} \mathbf{h} = 0$$

where x_i''s are the points of the previous image and x_i's are corresponding points of present image. The solution **h** is thus the null-space of this 8x9 matrix, which can be solved using known methods such as singular value decomposition.

Once this homography is computed, we get the mapping required to augment a given 2D virtual object or a texture-mapped 3D polygon defined relative to the plane, with Z=0 for all vertices. So, "Virtual video screens, 2D board games, and textual overlays can be achieved automatically without the need for a manual calibration procedure".[3]

4.2 3D Augmentations

4.2.1 Initial simpler approach

One way of augmenting a 3D point cloud into a 2D image is by projecting 3 orthogonal vectors in the 3D world on to the 2D image and track these vectors across the image sequence of the video. This can be done by marking 4 points that correspond to the 3 orthogonal axes X, Y, Z and tracking these points using SIFT. Now each point P in the 3D world represents a vector that is a linear sum of the 3 orthogonal vectors:

$$P_{3\times1} = aX + bY + cZ$$

and therefore the corresponding 2D point P' on the image will be the linear sum of the projections of the 3 orthogonal vectors on the 2D image x', y', z' which are being tracked on the images. Now to find the 2D points for all the 3D points in the 3D point cloud we can simply represent each point as

$$P'_{2\times1} = ax'_{2\times1} + by'_{2\times1} + cz'_{2\times1}$$

The 3D object can be augmented on to the 2D image by mapping all the 3D points using the above equation. This approach does not require the computation of a Camera matrix and hence is simpler. Although it better than the orthographic approach in the sense that it takes care of the scaling of the object as it moves closer or further from the camera, it is not as accurate as the perspective model because it assumes that all the vectors that are parallel to X, Y, Z in 3D will remain parallel to the projections of X, Y, Z in 2D. Another limitation is that it requires that the projection of 3 orthogonal vectors will always be available on the 2D image. Results are shown in Figures 5 and 6 where we augment a house using lines and a teapot using a point cloud. Figures 7 to 9 show another result.

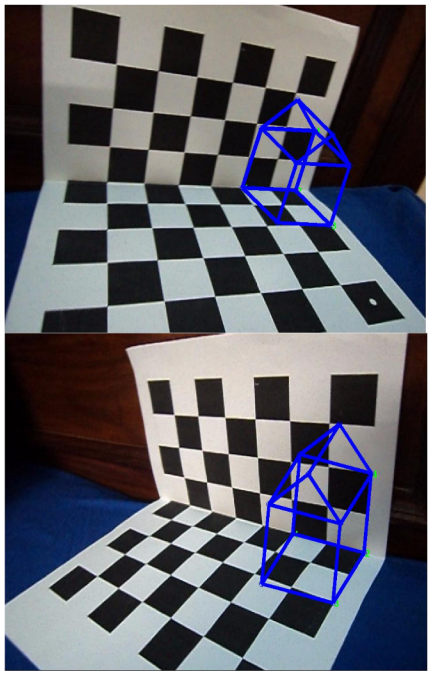

Figure 5: Result(1) of 3D augmentation (Method: 1)

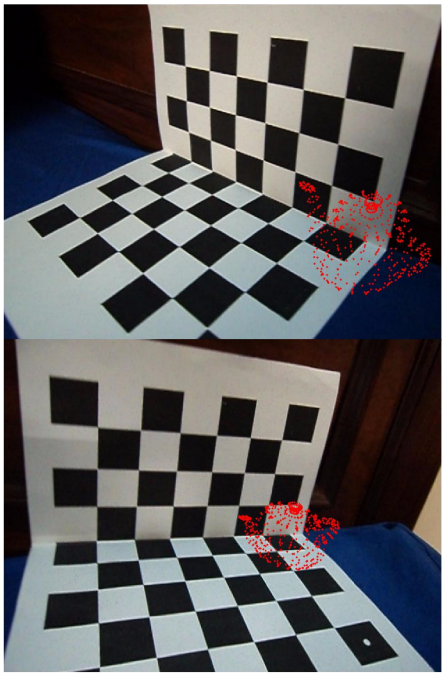

Figure 6: Result(2) of 3D augmentation (Method: 1)

Figure 7: Result(3) of 3D augmentation (Method: 1)

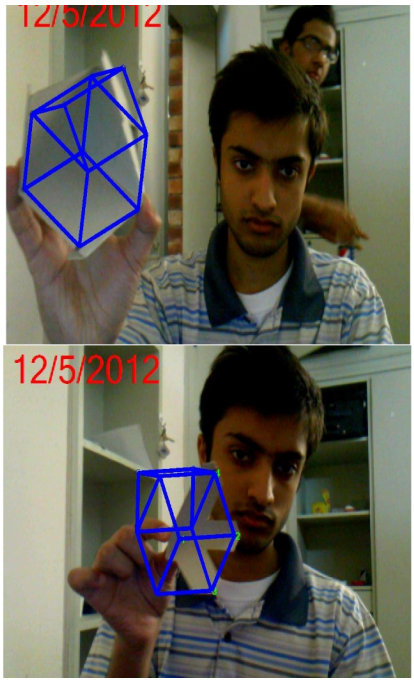

Figure 8: Result(3) of 3D augmentation (Method: 1)

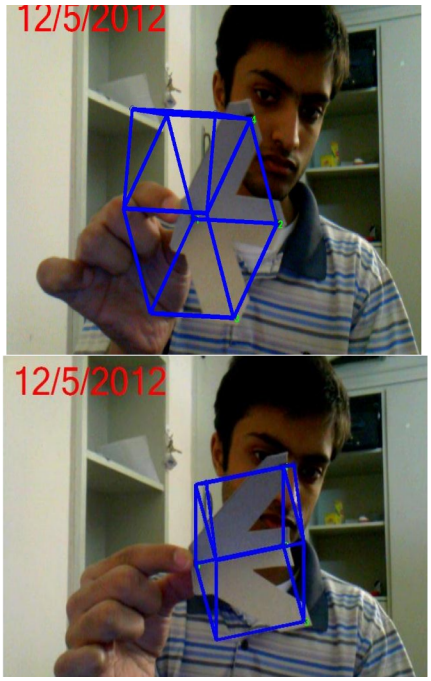

Figure 9: Result(3) of 3D augmentation (Method: 1)

4.2.2 *Self Calibration Approach[3]*

Coming back to the planar pattern, we have assumed that the pattern is at Z=0 plane in the real world co-ordinates. We get a homography between two images of this pattern. But the homography H can not be directly used to augment a 3D object in the video since we always assume the Z component of the pattern plane to be zero. However, we can use the work by Zhengyou Zhang[5] to calculate missing parameters of the $[R \mid T]$. Take $m_x f = f_x$ and $m_y f = f_y$, then H can be written as,

$$\begin{pmatrix} f_x r_{11} & f_x r_{12} & f_x t_1 \\ f_y r_{21} & f_y r_{22} & f_y t_2 \\ r_{31} & r_{32} & t_3 \end{pmatrix}$$

Since R is a rotation matrix, its orthogonality properties give,

$$r_{11}^2 + r_{21}^2 + r_{31}^2 = 1$$

$$r_{12}^2 + r_{22}^2 + r_{32}^2 = 1$$

$$r_{11} r_{12} + r_{21} r_{22} + r_{31} r_{32} = 0$$

Combining above equations with elements $\mathbf{h} = [h_{11} h_{12} h_{13} h_{21} h_{22} h_{23} h_{31} h_{32} h_{33}]^T$ of H, we get following solutions for f_x, f_y and λ,

$$f_x = \sqrt{\frac{h_{11}h_{12}(h_{21}^2 - h_{22}^2) - h_{21}h_{22}(h_{11}^2 - h_{12}^2)}{-h_{31}h_{32}(h_{21}^2 - h_{22}^2) - h_{21}h_{22}(h_{31}^2 - h_{32}^2)}}$$

$$f_y = \sqrt{\frac{h_{11}h_{12}(h_{11}^2 - h_{22}^2) - h_{21}h_{22}(h_{11}^2 - h_{12}^2)}{-h_{31}h_{32}(h_{11}^2 - h_{12}^2) - h_{11}h_{12}(h_{31}^2 - h_{32}^2)}}$$

and once the *intrinsic* parameters have been computed, λ is com-

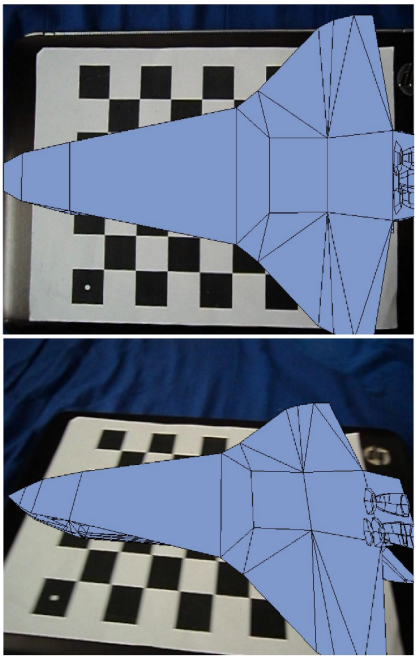

Figure 10: 3D augmented shuttle (Method: 3)

puted to be,

$$\lambda = \frac{1}{\sqrt{h_{11}^2/f_x^2 + h_{21}^2/f_y^2 + h_{31}^2}}$$

The *extrinsic* can now be computed as follows,

$$\begin{pmatrix} r_{11} & r_{12} & r_{13} & t_1 \\ r_{21} & r_{12} & r_{13} & t_2 \\ r_{31} & r_{32} & r_{33} & t_3 \end{pmatrix} =$$

$$\begin{pmatrix} \lambda h_{11}/f_x & \lambda h_{12}/f_x & r_{21}r_{32} - r_{31}r_{22} & \lambda h_{13}/f_x \\ \lambda h_{21}/f_y & \lambda h_{22}/f_y & r_{31}r_{12} - r_{11}r_{32} & \lambda h_{23}/f_y \\ \lambda h_{31} & \lambda h_{32} & r_{11}r_{22} - r_{21}r_{12} & \lambda h_{33} \end{pmatrix}$$

r_{13}, r_{23}, and r_{33} are found by using the fact that the Z axis of the rotation matrix is orthogonal to the other two. So, now we have both K and $[R \mid T]$. $K[R \mid T]$ gives us the required *Perspective camera matrix*.

The process goes like this, 1) user places an object in the video relative to the pattern being tracked, 2) as the *Homography H* is being computed at every step (SIFT used for the tracking of corresponding features continuously), 3) using the homography, the camera matrix is approximated with the help of above procedure, 4) The camera matrix is applied to the 3D co-ordinates of the virtual object aligned according to the tracked pattern, 5) The 2D co-ordinates we get are then augmented into the present image before moving to the next image in the sequence. The results of augmented shuttle are shown in Fig

References

[1] David G. Lowe: *Distinctive Image Features from Scale-Invariant Keypoints*

[2] Lukas Hohl, Till Quack: *Markerless 3D Augmented Reality*

[3] Shahzad Malik: *Robust Registration of Virtual Objects for Real-Time Augmented Reality*

[4] Elan Dubrofsky : *Homography Estimation: Master Essay, Carleton University, 2007*

[5] Zhengyou Zhang: *A Flexible New Technique for Camera Calibration* IEEE Transactions on Pattern Analysis and Machine Intelligence

[6] Yu-Tseh Chi, Jeffrey Ho and Ming-Hsuan Yang: *A Direct Method for Estimating Planar Projective Transform*