Placing Three-Dimensional Models in an Uncalibrated Single Image of an Architectural Scene

Sara Keren
Electrical Engineering Dept.
Technion

Ilan Shimshoni
Industrial Engineering Dept.
Technion

Ayellet Tal
Electrical Engineering Dept.
Technion

ABSTRACT

This paper discusses the problem of inserting three-dimensional models into a single image. The main focus of the paper is on the accurate recovery of the camera's parameters, so that 3D models can be inserted in the "correct" position and orientation. An important aspect of this paper is a theoretical and an experimental analysis of the errors. We also implemented a system which "plants" virtual 3D objects in the image, and tested the system on many indoor augmented reality scenes. Our analysis and experiments have shown that errors in the placement of the objects are un-noticeable.
Keywords: Augmented reality, camera calibration, architectural scenes.

1. INTRODUCTION

A basic problem in many augmented reality systems is how to create new artificial images given real images. A common approach is to create a new image of a scene from several other images of that scene taken from different viewpoints [6, 19]. Another approach is to insert 3D models into video sequences [3, 17, 9, 15]. We deal with a different problem – that of inserting 3D models into a single image.

Solving this problem requires solving several sub-problems. The first problem is how to recover the camera's internal calibration parameters and position within the scene accurately enough so that 3D models can be inserted into the image in the "correct" position [13, 5, 8]. The second problem is how to recover the lighting conditions in the scene [4, 14, 10, 18]. The third is how to deal correctly with occlusions of the new object with the existing objects in the scene [21].

In this paper we concentrate on the first problem. Figure 1 illustrates the intention of our algorithm and the results of running our system on a typical image. This picture was taken by a digital camera whose internal parameters and its location and orientation in the room are unknown. We "placed" on the table artificial objects. The green chime "hanging" from the ceiling is also artificial. We have chosen to place objects which look artificial, to distinguish them from real objects. The emphasis here is on the perfect location, orientation and size of the objects in the image.

Solving the problem of placing artificial objects given a single image without any prior knowledge about the scene is impossible. We therefore chose a large class of images for which there is prior information about the class but not much information has to be supplied with the image in order to perform the task. The class we chose is images of architectural scenes [8, 16, 2, 20], and in particular the interior of buildings. Typically, in these scenes lines appearing in the images are parallel to the three major axes. This information can be used to recover the required parameters.

Still the problem is not trivial. The main question is the following. Can the unknown parameters be recovered accurately enough from noisy measurements from one image such that they can be used to "plant" 3D models in the image in a way that they will look correct to the viewer. In this paper we answer this question in the affirmative.

Basically the problem is to estimate from the image the projection matrix which transforms 3D points to points in the image. Estimating this projection matrix from lines parallel to the major axes was studied in [13, 5, 8]. Usually to evaluate such algorithms, the accuracy of the recovered projection matrix is compared to the ground-truth. However, in our application the most important question is whether the results are accurate enough so that a human viewer will not detect differences between a real object and its 3D model inserted into the scene. We therefore direct our theoretical and experimental analysis to test projection errors.

There are two possible ways to insert 3D models into images. The first option is to reconstruct a three-dimensional scene from the images and let the user "place" the artificial objects in their locations in three dimensions. Then the augmented scene is projected back to image space. The second option is to choose a location in the image for the object to be placed in. Consequentially, the errors measuring the distance between the real location and the calculated location in the image, differ. We take the second avenue and show that the errors are smaller. Intuitively, this is the case since the point where the object is located in the image has no translation error (but might still have orientation errors). Other points on the model may also have translation errors, but obviously, they would be small.

We utilize vanishing points and lines for camera calibration, an approach which is best suited for architectural scenes, due to the orthogonality and parallelism that exist in these scenes. In [11] the problem of reconstructing a textured three dimensional model of a static urban scene viewed by one or several cameras whose relative positions are unknown, is addressed. In [7] the problem of recovering 3D models from two or more uncalibrated images of architectural scenes is addressed and a 3D model is recovered. In [8, 16] methods for creating 3D graphical models of scenes from one or two image are presented. The techniques presented are for metric reconstruction and for correct representation of an-

Figure 1: An augmented room

gles and length ratios. In particular, in [8], it is describes how 3D affine measurements can be computed from a single perspective view of a scene given only a minimal geometric information which can be determined from the image.

Our work has two novel aspects. First, we present a new theoretical error-analysis method. This method can be used for measuring the goodness of various augmented reality algorithms. We also compare the method to the Monte-Carlo estimation technique, in order to demonstrate its quality.

Second, we implemented a system and tested it on many indoor scenes, planting in them various 3D models. The models were placed on various planes in the picture – on furniture, on the floor, hanging from the ceiling and hanging on walls. In all cases, the size, orientation and position of the objects were highly accurate. A system like this has a potential for various commercial applications, for instance for furniture or household companies. The client can bring an image of a room and the system can show the client suggestions for interior design of the room using 3D models of objects from the company's stock.

The paper continues as follows. In Section 2 we describe our method for extracting the projection matrix. The error analysis is described in Section 3. We view this section as the main theoretical contribution of the paper. The results of testing our system on real indoor scenes are presented in Section 4. We conclude in Section 5.

2. THE APPROACH

This section explains the notations and describes the methods used for camera calibration. The goal is to find the parameters of the camera in order to construct a virtual camera with the same parameters as the real camera. This virtual camera projects the virtual objects on the real image.

This section is divided into two parts. In the first part we review the mathematical derivation of the method. In the second part we describe the numerical methods used to extract from the image the information needed for the algorithm.

2.1 Derivation

A camera maps the 3D world to the image plane. Given a point $P = (X, Y, Z)$ in the world its image p in homogeneous coordinates is: $p \cong R(P - C)$, where $\cong$ means equal up to a multiplicative constant, C is the camera center and the orientation of the camera is represented by the rotation matrix R. The camera center and the rotation matrix are the *extrinsic* parameters of the image.

The internal calibration matrix K converts a normalized point p to its pixel coordinates:

$$K = \begin{pmatrix} f & 0 & u_0 \\ 0 & f & v_0 \\ 0 & 0 & 1 \end{pmatrix},$$

where f is the focal length and (u_0, v_0) is the principal point (the projection of points in direction $(0, 0, 1)$). These parameters are known as the *intrinsic* calibration parameters of the camera.

The projection of the point P to the pixel $s = (u, v, 1)$ in the image is:

$$s \cong KR(P - C) = [KR| - KRC] \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix}.$$

We denote by $M = [KR| - KRC]$ the 4×3 projection matrix. Thus when we are given M we can compute the projection of any 3D point P to the image.

Vanishing Points: Given an image containing lines which are parallel in the world, these lines meet at a point in the image termed a vanishing point. Recall that if the direction of the lines in 3D is D, the vanishing point is at the image of the point $(D, 0)$.

Recovery of the Projection Matrix: We assume that o is the image of the origin of the world coordinate system O. This point is selected arbitrarily by the user (e.g. a corner in the room) and the world coordinate system is aligned with the major axes in the room. Three sets of lines parallel to the three major axes are extracted from the image, and their vanishing points $V_x = (u_x, v_x, 1)$, $V_y = (u_y, v_y, 1)$ and $V_z = (u_z, v_z, 1)$ are computed. That is to say, V_x, V_y, V_z and o are the images of the world points $(1,0,0,0)$, $(0,1,0,0)$, $(0,0,1,0)$ and $(0,0,0,1)$ respectively. We denote the columns of M by m_x, m_y, m_z, m_o. Applying M to these four special points we get

$$\lambda_1 V_x = m_x, \quad \lambda_2 V_y = m_y, \quad \lambda_3 V_z = m_z, \quad \lambda_4 o = m_o,$$

where the λ_i's are unknown multiplicative constants.

In order to recover the unknown λ_i's, recall that $[m_x|m_y|m_z] = KR$. Thus $(KR)(KR)^T = KRR^TK^T = KK^T$. Representing KR by the vanishing points we get

$$KR = \begin{pmatrix} \lambda_1 u_x & \lambda_2 u_y & \lambda_3 u_z \\ \lambda_1 v_x & \lambda_2 v_y & \lambda_3 v_z \\ \lambda_1 & \lambda_2 & \lambda_3 \end{pmatrix} = \begin{pmatrix} u_x & u_y & u_z \\ v_x & v_y & v_z \\ 1 & 1 & 1 \end{pmatrix} \begin{pmatrix} \lambda_1 & 0 & 0 \\ 0 & \lambda_2 & 0 \\ 0 & 0 & \lambda_3 \end{pmatrix}.$$

Thus

$$(KR)(KR)^T = \begin{pmatrix} u_x & u_y & u_z \\ v_x & v_y & v_z \\ 1 & 1 & 1 \end{pmatrix} \begin{pmatrix} \lambda_1^2 & 0 & 0 \\ 0 & \lambda_2^2 & 0 \\ 0 & 0 & \lambda_3^2 \end{pmatrix} \begin{pmatrix} u_x & v_x & 1 \\ u_y & v_y & 1 \\ u_z & v_z & 1 \end{pmatrix}$$
$$= \begin{pmatrix} f^2 + u_0^2 & u_0 v_0 & u_0 \\ u_0 v_0 & f^2 + v_0^2 & v_0 \\ u_0 & v_0 & 1 \end{pmatrix} = KK^T.$$

This yields six equations in the six unknowns $f, u_0, v_0, \lambda_1, \lambda_2, \lambda_3$ which can be readily solved yielding a unique solution.

In order to complete the recovery of the projection matrix we still have to recover the last column m_o which is equal to $\lambda_4 o$ for an unknown value of λ_4. λ_4 represents the depth of the world origin from the camera center in the direction of the principal ray. In order to obtain this value we require input from the user as will be explained in Section 4.

Recovery of External Calibration Parameters: The rotation matrix R can be recovered by multiplying K^{-1} by KR which are both known. The location of the camera C is orthogonal to the three rows of M. Using this constraint C can be computed analytically.

Calculating the World Coordinates: Since the projection matrix is not invertible, it is impossible to recover the world coordinates of a point from the image coordinates without extra information. However, if we know that the world point lies on a known plane, there is a one-to-one mapping (a homography) between the point in the image and its world coordinates.

2.2 Numerical Methods

The only information that has to be extracted from the image is the set of vanishing points. Using a standard edge/line detector we extract straight lines from the image. These lines are divided automatically into three sets of lines which intersect approximately at the same point in the image. The question is how to find the most probable vanishing point from a set of lines in the image. In an ideal setting there is only one intersection point that all the lines in the set are incident on. However, inaccurate line detection leads to several intersection points in one set of projected parallel lines. We need to find a point that is the closest to the accurate vanishing point.

Let us consider a set of projected parallel lines $L = \{l_1, \cdots, l_n\}$ where $l_i = (a_i, b_i, c_i)$. Recall that a point $p = (x, y, 1)$ is incident on $l = (a, b, c)$ if $ax + by + c = 0$ or $l \cdot p = 0$. We look for the point $p = (u_p, v_p, 1)$ that is incident on all the lines in L, $Lp = 0$.

In order to find the point closest to all the lines in a least-squares sense , it is customary to solve the least squares problem of $[A|B]p = -C$ where A, B and C are the vectors of the a_i's, b_i's, and c_i's respectively. This solution however, is not the approximate maximum likelihood (AML) estimate of the vanishing point because the above setting is equivalent to assuming that there is an error in the vanishing point and not in the lines l_i. The truth however is the opposite.

We want to find the point p that is the approximate maximum likelihood (AML) estimate of the vanishing point [1]. We call the noise-free set of lines L and the measured set of lines L'. We say that a set of lines supports a point p if all the lines in the set meet at the point p. We want to find a set L'' that supports a point p that is the most likely among all possible sets of lines, given the set of measured lines L'. In other words, we want to find the point p that maximizes the likelihood of L'' being the true set of lines, given the measured set L'. Thus given a candidate point p we have to give an estimate for the probability that it is the vanishing point.

Figure 2 illustrates the geometric solution for finding the set L''. We assume that q_1' and q_2' are the endpoints on a line l' in L'. We want to find a line l'' which is incident on p such that the squared distance from q_1' and q_2' to that line (distances to q_1'' and q_2'') is minimal. In other words, let d_1 and d_2 be the distances between q_1' q_1'', and q_2' and q_2'', respectively. Our goal is to minimize the sum of distances $(d_1^2 + d_2^2)$ from the new endpoints to the old endpoints. The angle θ between the x axis and the line created by the optimal q_1'' and q_2'' is:

$$\theta = \frac{1}{2} atan2(2(x_1' - u)(y_1' - v) + (x_2' - u)(y_2' - v)),$$
$$(x_1' - u)^2 + (x_2' - u)^2 - (y_1' - v)^2 - (y_2' - v)^2)).$$

The distances are:

$$d_1^2 = (-\sin\theta(x_1' - u) + \cos\theta(y_1' - v))^2$$
$$d_2^2 = (-\sin\theta(x_2' - u) + \cos\theta(y_2' - v))^2.$$

The line segments of L' are independent on each other, therefore to maximize the likelihood of the set L'' being the true set, the sum of distances of all lines in $L' = (X_1', Y_1', X_2', Y_2')$ need to be minimized. The function F to be minimized is:

$$F(u, v, X_1', Y_1', X_2', Y_2') = \sum_{i=1}^{n} (d_{1i}^2 + d_{2i}^2) =$$
$$\sum_{i=1}^{n} (-\sin\theta_i(x_{i1}' - u) + \cos\theta_i(y_{i1}' - v))^2 \tag{1}$$
$$+(-\sin\theta_i(x_{i2}' - u) + \cos\theta_i(y_{i2}' - v))^2.$$

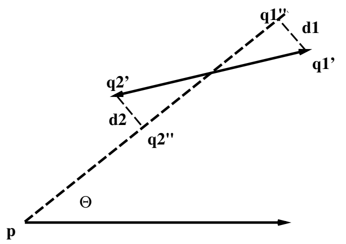

Figure 2: Illustration of the likelihood function.

Finding the point $p = (u, v)$ which minimizes this function gives the best estimate for the vanishing point we seek. Thus we apply a non-linear optimization method to find p.

3. ERROR ESTIMATION AND ANALYSIS

This section focuses on the accuracy of the projection of the 3D virtual objects onto a single 2D image. The quality of the recovered camera calibration determines the accuracy of the parameters of the virtual camera, used to "take a picture" of the virtual objects. Two methods for estimating the errors are described: a novel analytical method and a Monte Carlo method. Both methods compute the covariance matrices of the parameters, which give us the distribution of the errors of the computed values as functions of the errors in the measured values, thus estimating the real errors. We will show that our proposed analytical method is almost as accurate as the Monte Carlo method. Since it is much faster, it can be the method of choice for augmented reality algorithms.

3.1 Analytical Estimation of Covariance

We describe below the analytical method for computing the covariance matrices of several parameters. Our objective is to estimate the dispersion of the vanishing points, the principal point, the focal length, the camera location in the world coordinate system and the projection of points in the world to the image.

We base our estimate on a first order approximation derived in [12], describing how to propagate approximately additive random perturbations through an algorithm step. Covariance propagation allows us to determine the covariance of Θ where $\Theta = G(X)$ from the covariance of X. We assume that G is a non-linear function that is approximately linear in the vicinity of the measured values of X. In this case the covariance matrix of Θ is given by $J\Sigma J^T$, where J is the Jacobian matrix of G evaluated at X.

A related problem is when G is not given explicitly. The ideal input is an $N \times 1$ vector X, the observed perturbed input is an $N \times 1$ vector $\hat{X} = X + \Delta X$, where ΔX is a random vector of noise. The correct solution to the problem is a $K \times 1$ vector Θ. The calculated solution is a randomly perturbed $K \times 1$ parameter vector $\hat{\Theta} = \Theta + \Delta\Theta$, where $\Delta\Theta$ is a random perturbation on Θ induced by the random perturbation ΔX on X. Finally, a function F relates the vectors X and Θ and the vectors $X + \Delta X$ and $\Theta + \Delta\Theta$. F has finite first and second partial derivatives with respect to each component of X and Θ including all the second partial derivatives. The basic problem is thus: given $\hat{X} = X + \Delta X$, determine $\hat{\Theta} = \Theta + \Delta\Theta$ that minimizes $F(\hat{X}, \hat{\Theta})$ assuming that Θ minimizes $F(X, \Theta)$. Let

$$g(\Theta, X) = \frac{\partial F}{\partial \Theta}(\Theta, X).$$

Since $\hat{\Theta}$ minimizes $F(\hat{X}, \hat{\Theta})$ and Θ minimizes $F(X, \Theta)$, both $g(\hat{X}, \hat{\Theta})$ and $g(X, \Theta)$ equal zero. By taking a Taylor series expansion of g around (X, Θ) we obtain a first order approximation:

$$\Delta\Theta = -\left(\frac{\partial g}{\partial \Theta}(\Theta, X)\right)^{-1} \frac{\partial g}{\partial X}(\Theta, X)\Delta X \tag{2}$$

This relation states how the random perturbation ΔX on X propagates to the random perturbation $\Delta\Theta$ on Θ. If $E[\Delta X] = 0$ then from Equation 2, $E[\Delta\Theta] = 0$ to a first order approximation. The covariance matrix of $\hat{\Theta}$ is then:

$$\Sigma_{\hat{\Theta}} = \mathcal{M}\Sigma_{\hat{X}}\mathcal{M}^T, \tag{3}$$

where

$$\Sigma_{\hat{X}} = E[\Delta X \Delta X^T], \qquad \mathcal{M} = \left(\frac{\partial g}{\partial \Theta}\right)^{-1} \frac{\partial g}{\partial X}. \tag{4}$$

We can apply the above theory to our calibration calculations. Recall that our inputs are three sets of end points of image straight lines, in the three principal directions. Let X be the set of accurate end points of the lines in one direction, $X = (x_1, y_1, x_2, y_2, \cdots, x_p, y_p)^T$, and let $\hat{X}$ be the corresponding noisy set measured in the image, $\hat{X} = (\hat{x}_1, \hat{y}_1, \hat{x}_2, \hat{y}_2, \cdots, \hat{x}_p, \hat{y}_p)^T$. The calibration algorithm first finds the principal vanishing points. In this case the function F is the function from Equation 1, $\Theta = (u, v)$ is the accurate vanishing point and $\hat{\Theta} = (\hat{u}, \hat{v})$ is the estimated vanishing point calculated from the noisy inputs. We can calculate the matrix $\mathcal{M}$ in Equation 4 from:

$$\begin{aligned}
g &= \begin{pmatrix} Fu \\ Fv \end{pmatrix} \\
\frac{\partial g}{\partial \Theta} &= \begin{pmatrix} F_{uu} & F_{uv} \\ F_{vu} & F_{vv} \end{pmatrix} \\
\frac{\partial g}{\partial X} &= \begin{pmatrix} F_{u\hat{x}1} & F_{u\hat{y}1} & \cdots & F_{u\hat{y}p} \\ F_{v\hat{x}1} & F_{v\hat{y}1} & \cdots & F_{v\hat{y}p} \end{pmatrix},
\end{aligned} \tag{5}$$

and use it to compute $\Sigma_{\hat{\Theta}vp}$ using Equation 3.

The next steps in the calibration process are calculations of the various parameters using deterministic non-linear functions of the vanishing points. These are intrinsic parameters of the camera(i.e., the principal point and the focal length). In these cases the covariance matrices are obtained by:

$$\Sigma_{\hat{P}} = J\Sigma_{\hat{\Theta}}J^T, \tag{6}$$

where $\hat{P}$ is the estimation of the required parameter and J is the Jacobian matrix of the function evaluated at the correct vanishing points.

Let $(u_x, v_x), (u_y, v_y)$ and (u_z, v_z) be the accurate vanishing points and $(\hat{u}_x, \hat{v}_x), (\hat{u}_y, \hat{v}_y)$ and $(\hat{u}_z, \hat{v}_z)$ be the corresponding estimated vanishing points in directions X, Y and Z respectively. Let (u_0, v_0) be the accurate principal point and $(\hat{u}_0, \hat{v}_0)$ be the estimated principal point. We can use the covariance matrix estimated for the vanishing points and this function to estimate the covariance matrix of the principal point and the covariance matrices of the scale factors λ_1, λ_2 and λ_3. The latter can then be used to estimate the covariance matrix of the focal length f.

The next step in the calibration phase is to find the location of the camera in the world coordinate system. The elements of the projection matrix are functions of the principal vanishing points and of the scale factors. They also depend on a couple of parameters given by the user: the global scale factor λ_4 and the image coordinates of the world origin $(o_x, o_y)^T$. Using Equation 6 the covariance of the estimated location of the camera can be calculated as well.

After computing the camera's intrinsic and extrinsic parameters we can calculate the projection of points in 3D. The system computes the world coordinates of a point marked by the point in the image where a virtual object should be placed. Estimating the error of the projection of the virtual object is more difficult than the previous estimations because it consists of two phases. We first estimate the covariance matrices of the components of the projection matrix as functions of the principal vanishing points, and then use the recovered covariance matrices to estimate the covariance matrices of the projected points to the image. This is done as follows:

$$u = \frac{M^{1T}(X,Y,Z,1)^T}{M^{3T}(X,Y,Z,1)^T}, \qquad v = \frac{M^{2T}(X,Y,Z,1)^T}{M^{3T}(X,Y,Z,1)^T}, \tag{7}$$

where M^{1T}, M^{2T} and M^{3T} are the first, second and third rows of the projection matrix M respectively, $(X, Y, Z, 1)^T$ is the world coordinates of a vertex of the virtual object and $(u, v)^T$ is its image coordinates. Let M be the accurate projection matrix and $\hat{M}$ be the estimated projection matrix. According to Equation 6 we can calculate the covariance of the estimated projection matrix as:

$$\Sigma_{\hat{M}} = J_M \Sigma_{\hat{\Theta}vp} J_M^T, \tag{8}$$

where J_M is the Jacobian matrix of the projection matrix evaluated at the correct vanishing points and $\Sigma_{\hat{\Theta}vp}$ is the covariance matrix of the vanishing points. We can now calculate the covariance of the estimated image coordinate as:

$$\Sigma_{(\hat{u},\hat{v})} = J_{uv} \Sigma_{\hat{M}} J_{uv}^T, \tag{9}$$

where J_{uv} is the Jacobian matrix of the image coordinates evaluated at the correct projection matrix.

3.2 Monte Carlo Estimation of Covariance

The Monte Carlo method estimates the covariance by exhaustive simulation [13]. It can be used to validate the results of the analytical method. This method is also preferable in cases where the first order approximation of the covariance is not accurate enough and thus the analytical estimation may be incorrect. The goal is to statistically compute the covariance matrices of the estimated parameters, having noisy input.

We created a synthetic 3D scene which was rendered from five different view points creating five different images, as shown in Figure 3. The location of the camera between each pair of the first three images is translated while the orientation of the camera remains the same. In the last three images the orientation of the camera changes while the location of the camera remains the same.

The inputs to each test are the following: 1) The intrinsic parameters of the camera. 2) Three sets of end points of eight straight lines, each belonging to one set of parallel lines in the world coordinate system: X, Y and Z. The end points are given as 3D points and are projected to the image by the computed projection matrix. 3) The location of the world origin in the image. 4) One additional point in the image and its 3D distance from the origin of the world coordinate system. This information is used for calculating the global scale factor. 5) Eight 3D points that are used as test points for testing the accuracy of the virtual object registration. These points are translated to 2D image points by the correct projection matrix once in the beginning of the test. Each point is accompanied by information regarding the plane the point lies on and its distance from the principal plane which is parallel to it.

First, the correct parameters are computed using the pure inputs. Afterwards, random noise is added to each end point of the input straight lines. The noise is Gaussian distributed with zero mean and standard deviation values $\sigma = 0.2, 0.4, 0.6, 0.8, 1$ (in pixels). We conducted 1000 experiments with each level of σ.

For each set of noisy inputs we compute the principal vanishing points, the intrinsic parameters of the camera, the projection matrix and the rotation and the translation of the camera. We also calculate the virtual object registration in the image, by computing the projection of 3D points onto the 2D image, using the computed projection matrix. In our setup, the user marks a point in the image where the virtual object should be placed. This 2D point is first transformed to a 3D point using the computed projection matrix, and the virtual object is placed in that coordinate in 3D. The object's vertices are then projected to the image according to the computed projection matrix.

In our experiments we created a test virtual object with 18 vertices, where the distances of these vertices from the center of the virtual object are 30cm, 60cm and 90cm in the directions X, Y and Z in the world coordinate system. These 18 test vectors are projected to the image using the computed projection matrix.

We compare the resulting parameters to the pure parameters and calculate the errors as follows: 1) The distance between the correct vanishing points to the perturbed vanishing points in pixels. 2) The distance from the pure principal point to the noisy principal point in pixels in each axis, divided by the size of the image in each axis. 3) The relative error in the estimated focal length. 4) In order to calculate the error in the rotation of the camera, the computed rotation matrix is multiplied by the inverse of the correct rotation matrix. The result is the rotation between the pure orientation of the camera and the perturbed orientation of the camera. We compute the quaternion of this rotation matrix to get its rotation angle and divide it by the field of view to get the relative error. 5) The error in the position of the camera center in centimeters. 6) The error in the image of the projected 3D points, represented by the distance and the angle between the correct projected vector and its corresponding noisy vector.

After performing 1000 simulations at a specific noise level we calculate the mean and the standard deviation of each kind of error. The standard deviation is estimated as the $2/3 * 1000 = 666$ element in the sorted vector of a specific error (which is a robust estimate for the standard deviation).

3.3 Error Estimation Results

In this section we present the results of the analytical error estimation as well as the results of the Monte Carlo error estimation, and compare the two for the test images in Figure 3.

	Test1	Test2	Test3	Test4	Test5
Image					
Rotation Angles(∘)	(-6.5,0,8.0)	(-6.5,0,8.0)	(-6.5,0,8.0)	(-6.5,0,15.0)	(-6.5,0,30.0)
Camera Center(cm)	(-200,-410,105)	(-250,-430,105)	(-300,-460,105)	(-300,-460,105)	(-300,-460,105)

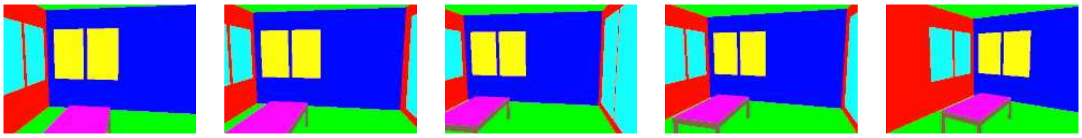

Figure 3: The synthetic test images, the angles between viewing direction and the principal directions and the camera's center in the world.

Table 1 shows the average of the errors of the intrinsic and the extrinsic parameters of the camera for all the test images calculated by the Monte Carlo error estimation method for noise level of $\sigma = 1$. Table 2 shows the standard deviation of the errors of the intrinsic and the extrinsic parameters of the camera for the test images calculated by the Monte Carlo error estimation method. We can see that the errors are negligible relative to their standard deviations. Thus our assumption of zero mean, made in the analytical method, is justified. The results for test4 and test5 are better than the others, due to the fact that in these images the viewing direction of the camera has a large angle with two of the principal directions. We conclude that the best camera orientation is when the world three planes XY, XZ and YZ form an angle close to $\pi/4$ with the image plane. In this case the vanishing points are closer to the image and the accuracy of their localization in the image increases [5].

Error in	Test1	Test2	Test3	Test4	Test5
Focal length % of focal length	0.066	0.051	0.088	0.033	0.002
u0 % of X dim of image	0.016	0.023	-0.020	-0.002	0.034
v0 % of Y dim of image	-0.007	-0.016	-0.006	-0.006	-0.011
Camera center X coord. in cm	-0.596	-0.836	0.451	-1.502	-0.29
Camera center Y coord. in cm	-0.233	0.085	1.281	-0.849	0.55
Camera center Z coord. in cm	0.465	0.391	0.619	0.447	0.547

Table 1: The average errors using the Monte Carlo error estimation method for noise level of $\sigma = 1$.

Error in	Test1	Test2	Test3	Test4	Test5
Focal length	4.4306	4.1744	4.6565	2.4134	1.0077
u0	1.0195	1.0161	1.1205	1.0926	1.1864
v0	1.365	1.2159	1.3124	0.721	0.5257
Rotation angle	0.4175	0.3937	0.4277	0.3839	0.3978
Camera center	19.084	18.801	22.521	12.498	6.5957

Table 2: The standard deviation of the errors for noise level $\sigma = 1$.

Figure 4 shows the results of the estimation of the errors of the intrinsic parameters of the camera and the translation of the camera using both methods. The graphs show that both methods give similar results, justifying the assumption made in the analytical method, that the behavior is linear.

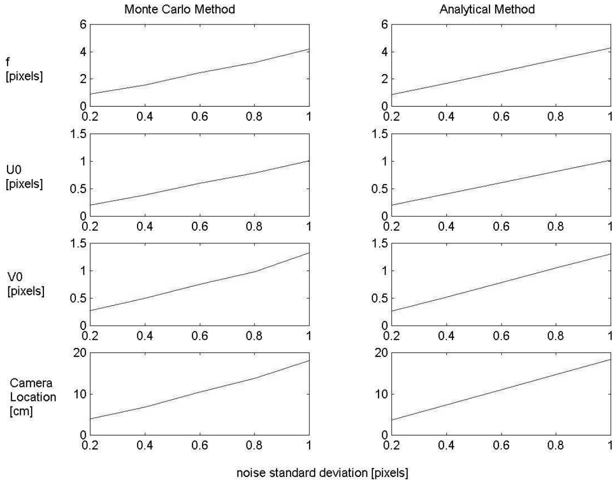

Figure 4: The standard deviation of the errors in test1 as a function of the input noise level. Left: Monte Carlo. Right: analytical.

Table 3 summarizes the results of the simulations of the test vectors. As described in the previous sub-section we chose 8 points in the synthetic scene whose locations are shown in Figure 5. We calculated the projection of 18 vectors originating in these test points, and representing the virtual object, as described in the previous sub-section.

The perturbed results are compared to the pure results in the following way. The angle between each of the pure 18 vectors originating from each test point and the perturbed 18 vectors, is calculated. Table 3 shows that the angles are very small, thus the perturbed vectors and the pure vectors are approximately in the same direction in the image.

We also calculated the distance in pixels between each of the pure 18 points (for each test point) and the perturbed 18 points, divided by the length of the pure projection of each vector. We conclude that the distortion of the virtual objects projected to the image is un-noticeable to the viewer, being at most one or two pixels for standard household items.

In conclusion, the error results obtained by both methods are similar except for the estimation of the covariance matrices of the projection of the 3D points, where the differences between the estimates is about $50\% - 65\%$. This is due to the first order approximation made by the analytical method. These estimates have proven that a system built using our approach will yield sufficiently accurate results.

		Test1	Test2	Test3	Test4	Test5
test point 1 (0,-110,50)	dist. from cam. [cm]	364.73	409.79	464.25	464.25	464.25
	error in length %	0.282	0.306	0.358	0.368	0.480
	error in angle [deg]	0.027	0.021	0.018	0.022	0.068
test point 2 (0,-20,100)	dist. from cam. [cm]	438.32	480.23	532.56	532.56	532.56
	error in length %	0.275	0.290	0.329	0.356	0.508
	error in angle [deg]	0.035	0.029	0.025	0.031	0.098
test point 3 (-20,0,180)	dist. from cam. [cm]	454.01	493.38	543.71	543.71	543.71
	error in length %	1.652	1.565	1.739	1.079	0.861
	error in angle [deg]	0.063	0.063	0.060	0.069	0.153
test point 4 (-300,0,10)	dist. from cam. [cm]	432.58	443.20	469.71	469.71	469.71
	error in length %	3.877	3.680	4.149	2.264	1.296
	error in angle [deg]	0.149	0.110	0.099	0.113	0.246
test point 5 (-20,-20,260)	dist. from cam. [cm]	456.65	495	544.08	544.08	544.08
	error in length %	5.474	5.033	5.518	2.928	1.586
	error in angle [deg]	0.092	0.072	0.070	0.063	0.133
test point 6 (-185,-20,0)	dist. from cam. [cm]	404.17	428.19	466.74	466.74	466.74
	error in length %	2.299	2.091	2.227	1.261	0.774
	error in angle [deg]	0.057	0.049	0.051	0.050	0.087
test point 7 (-100,-120,210)	dist. from cam. [cm]	324.23	360.03	408.20	408.20	408.20
	error in length %	0.249	0.232	0.248	0.233	0.270
	error in angle [deg]	0.028	0.028	0.029	0.028	0.042
test point 8 (-180,-210,210)	dist. from cam. [cm]	226.77	253.62	296.52	296.52	296.52
	error in length %	0.255	0.245	0.259	0.242	0.260
	error in angle [deg]	0.123	0.067	0.040	0.056	0.224

Table 3: The standard deviation of the errors of the projected test vectors for images with noise with $\sigma = 1$(pixel). The first row is the distance of the point from the camera. The second row is the average standard deviation of the errors in the lengths of the projected vectors, relative to their lengths. The third row is the average standard deviation of the errors in the angles between the projected and the pure vectors.

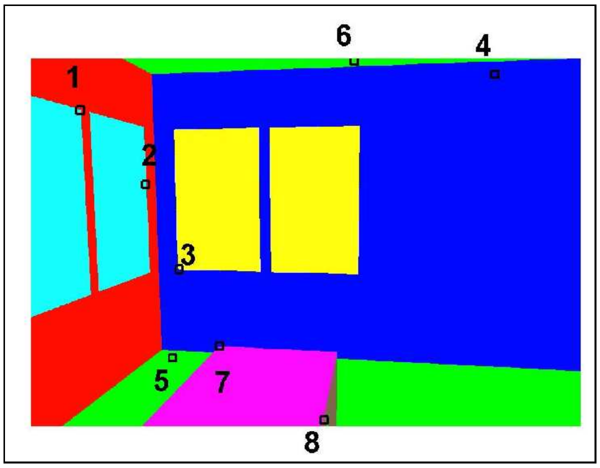

Figure 5: Test1 image with the 8 test points marked on it.

4. IMPLEMENTATION AND RESULTS

We built an interactive system that lets users augment a single indoor image with virtual objects. The system is implemented on a PC with the Windows NT operating system, using OpenGL. The test images were taken by a Canon Powershot S100 digital camera.

An edge detector is applied to the picture to locate straight lines. These lines are then automatically partitioned into four sets of lines: X lines, Y lines, Z lines and nonaligned lines. From the first three sets the vanishing points are estimated. In order to calculate the extrinsic parameters of the camera, the user needs to supply the system with two inputs: the location of the origin of the world in the image and some real-world distance (in cm) between two points in the image.

To insert a virtual object into the image, the user needs to specify its desired location on the image, with a mouse click, and set the distance of the plane on which the object is to be located from the principal plane parallel to it. After adding the object to the image the user can choose to rotate the object around its local rotation axis.

The system computes five transformation matrices and applies them to the object, as follows.

$$Projected\_Object \cong [K][R][T][GR][LR](Virtual\_Object)$$

[K] is the camera's intrinsic parameters matrix. [R] is the camera's rotation matrix. [T] is the translation of the virtual object to its location in the world coordinate system relatively to the location of the camera. [GR] is the global rotation matrix; it is computed according to the surface which the virtual object touches. Finally, [LR] is the local rotation matrix which rotates the virtual object in its local coordinate system.

There is a semi-automatic capability in the system that deals with occlusions of virtual objects by real objects, as illustrated in Figure 9, where the toy train is partially occluded by the couch.

Figures 6–9 illustrate some results obtained by our system. The images were taken from different points of view, and they show three different rooms. Each image is augmented with several virtual objects, some hanging from the ceiling, others are placed on the floor or on the table, and some are hanged on the walls, hence demonstrating the use of the system on various planes.

5. CONCLUSION

This paper has addressed the problem of camera calibration recovery and virtual object registration given a single image of an indoor architectural scene. Our approach has exploited the orthogonality and the parallelism which are prevalent in such scenes. The method is based on finding the principal vanishing points and using them to construct the projection matrix, which encapsulates both the intrinsic and the extrinsic parameters of the camera. With these parameters, we can construct a virtual camera which has the same parameters as the real camera. The virtual camera lets us augment the image by overlaying virtual objects upon the real image in the correct position and orientation.

A major contribution of this paper is the error analysis of the camera calibration and the object registration method. We have described an analytical method for computing the covariance matrices of the camera's parameters and of the virtual object registration in the real image. We have also described a Monte Carlo estimation of the same covariance matrices and compared them to the covariance matrices computed by the analytical method. The analytical method is preferable in estimating the covariance matrices of the camera parameters because it is more efficient computationally. The Monte Carlo method, however, is preferable in estimating the covariance matrices of the object registration errors, because it is more accurate.

We have shown that the standard deviations of the errors of the intrinsic and the extrinsic parameters of the camera are low enough to be useful for augmenting a single image with virtual objects. We have also shown that the distortion of virtual objects is un-noticeable to the viewer. Finally, we have developed a system that implements camera calibration recovery and object registration, while requiring minimal information from the end-user. Placing virtual objects in the scene is done without constructing a full three-dimensional model of the real scene.

Future work includes searching for methods for calibrating images with less than three principal vanishing points, and integration of our system with known methods for handling occlusion, lighting and shading.

6. REFERENCES

[1] A. Adam, E. Rivlin, and I. Shimshoni. Computing the sensory uncertainty field of a vision-based localization sensor. *IEEE Trans. on Robotics and Automation*, 17(3):258–267, June 2001.

[2] M. Antone and S Teller. Automatic recovery of relative camera rotations in urban scenes. In *Proc. IEEE Conf. Comp. Vision Patt. Recog.*, pages II–282–289, 2001.

[3] M.-O. Berger, B. Wrobel-Dautcourt, S. Petitjean, and G. Simon. Mixing synthetic and video images of an outdoor urban environment. *Machine Vision Applications*, 11(3):145–159, 1999.

[4] S. Boivin and A. Gagalowicz. Image-based rendering of diffuse, specular and glossy surfaces from a single image. In *Proc. ACM SIGGRAPH*, pages 107–116, 2001.

[5] B. Caprile and V. Torre. Using vanishing points for camera calibration. *Int. J. of Comp. Vision*, 4(2):127–140, March 1990.

[6] C.F. Chang, G. Bishop, and A. Lastra. Ldi tree: A hierarchical representation for image-based rendering. In *Proc. ACM SIGGRAPH*, pages 291–298, 1999.

[7] R. Cipolla, D.P. Robertson, and E.G. Boyer. Photobuilder - 3D models of architectural scenes from uncalibrated images. In *IEEE International Conference on Multimedia Computing and Systems*, pages 25–31, 1999.

[8] A. Criminisi, I. Reid, and A. Zisserman. Single view metrology. *Int. J. of Comp. Vision*, 40(2):123–148, November 2000.

[9] L. de Agapito, R.I. Hartley, and E. Hayman. Linear calibration of a rotating and zooming camera. In *Proc. IEEE Conf. Comp. Vision Patt. Recog.*, pages I:15–21, 1999.

[10] P.E. Debevec. Rendering synthetic objects into real scenes: Bridging traditional and image-based graphics with global illumination and high dynamic range photography. In *Proc. ACM SIGGRAPH*, pages 189–198, 1998.

[11] O.D. Faugeras, S. Laveau, L. Robert, G. Csurka, and C. Zeller. 3-D reconstruction of urban scenes from image sequences. *Comp. Vis. Im. Understanding*, 69(3):292–309, March 1998.

[12] R.M. Haralick. Propogating covariance in computer vision. *Int. J. of Patt. Recog. and Art. Intell.*, 10:561–572, 1996.

[13] R. Hartley and A. Zisserman. *Multiple Views Geoemtry in Computer Vision*. Cambridge University Press, 2000.

[14] M.L. Koudelka, P.N. Belhumeur, S. Magda, and D.J. Kriegman. Image-based modeling and rendering of surfaces with arbitrary brdfs. In *Proc. IEEE Conf. Comp. Vision Patt. Recog.*, pages I:568–575, 2001.

[15] K.N. Kutulakos and J.R. Vallino. Calibration-free augmented reality. *IEEE Transactions on Visualization and Computer Graphics*, 4(1):1–20, 1998.

[16] D. Liebowitz, A. Criminisi, and A. Zisserman. Creating architectural models from images. In *Proc. Eurographics*, pages 39–50, 1999.

[17] J.P. Mellor. Real-time camera calibration for enhanced reality visualization. In *Proc. Computer Vision, Virtual Reality and Robotics in Medicine*, pages 471–475, 1995.

[18] R. Ramamoorthi and P. Hanrahan. A signal-processing framework for inverse rendering. In *Proc. ACM SIGGRAPH*, pages 117–128, 2001.

[19] J. Shade, S.J. Gortler, L. He, and R. Szeliski. Layered depth images. In *Proc. ACM SIGGRAPH*, pages 231–242, 1998.

[20] S Teller, M. Antone, Z. Bodnar, M. Bosse, S. Coorg, M. Jethwa, and N. Master. Calibrated, registered images of an extended urban area. In *Proc. IEEE Conf. Comp. Vision Patt. Recog.*, 2001.

[21] M. Wloka and B. Anderson. Resolving occlusion in augmented reality. In *Symp. on Interactive 3D Graphics*, pages 5–12, 1995.

Figure 6: An augmented family room: the wine glasses, vases, candle holders, letter holder, are all artifical. Note that the two vases have different sizes due to perspective.

Figure 7: An augmented study: The letter holder, the phone (both hung on the wall, partially occluded by the computer), the green book, the tea cups, the toy train and the vase, are all artificial.

Figure 8: The study from a different point of view: The artificial objects are placed in different locations.

Figure 9: An augmented living room (excuse the virtual mess): A shoe and a toy train are placed on the floor, both partially occluded. All the objects on the table are artificial as well.