A 3D Pattern for Pose Estimation for Object Capture

Lei Wang, Cindy Grimm, and Robert Pless
Department of Computer Science and Engineering
Washington University
One Brookings Drive, St. Louis, MO, 63130
{lw3,cmg,pless}@cse.wustl.edu

Abstract

We describe a new pose estimation approach for a 3D object capture system. This 3D pose estimation approach offers several advantages: increased visibility, robustness to lighting conditions, and improved reliability with evenly distributed errors. The calibration pattern is built using 3D conic features. We use simplex search to find the camera position and orientations that minimizes the error between the projected 3D cone features and the corresponding 2D image features. We demonstrate that our approach is accurate, efficient and robust.

Keywords: CR Category: I.2.10 [Vision and Scene Understanding]: Calibration, Object Capture, 3D Pattern, Pose Estimation, Ellipse Fitting, Conic

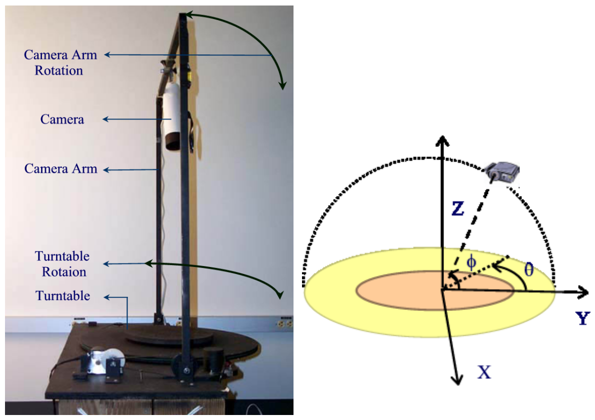

Figure 1: Turntable System.

1 Introduction

Our 3D calibration approach was motivated by the need for robust camera calibration in a turntable-based 3D object capture system. Since the objects we intend to capture are not restricted to be Lambertian, passive image-based modeling techniques are preferred. Camera calibration is one of the most critical aspects of image-based modeling. We desire a highly detailed representation, especially for fuzzy objects where it is important to capture small details. This requires many images taken from different views in order to cover every aspect of the object.

We use a turntable system (see Figure 1) to acquire the raw images of one object from any angle in the upper hemisphere. The system has a turntable, which rotates through 360 degrees, and a camera arm which rotates through 90 degrees. We know the approximate position and orientations of the camera from the motors; we use a calibration pattern to achieve more accurate pose estimation.

Depending on the required detail of the object, we may need anywhere from hundreds to thousands of pictures to reconstruct the virtual model. Because of this, the camera calibration has to be entirely automatic. The calibration pattern must also be visible from the entire hemisphere and robust to different lighting conditions. Furthermore, since each view of the object contributes equally to the virtual shape, we want equal reliability for every view. Finally, the pattern must be easy to build and detect. The 3D calibration pattern we designed is a truncated cone with two ellipses (see Figure 2). Image processing detects the projected 2D ellipses. We perform a non-linear search to find the translation and rotation of the camera so that the projected 3D ellipses match the detected 2D ellipses.

During the raw image capture phase, the camera lens is fixed and not re-focused or zoomed. Therefore, we only need to calculate the intrinsic parameters for the camera once at the start of the capture process. We calculate the extrinsic parameters for every image.

In Section 2, we discuss related work and how we adapted traditional approaches to meet our own needs. In Section 3, we introduce our 3D pattern and how to build it. In Section 4, we focus on feature detection techniques such as color detection, boundary detection, pixel grouping and fitting algorithms. In Section 5, we describe the approach used for solving for the intrinsic parameters. In Section 6, we discuss how to solve for the extrinsic parameters. We close with accuracy tests and results.

2 Previous Work

The pinhole camera model defines the relationship between a 3D point M and its 2D image projection m as:

$$sm = A\ RT\ M \tag{1}$$

where

$$m = [u\ v\ 1]^T \tag{2}$$
$$M = [x\ y\ z\ 1]^T \tag{3}$$

s is a scale factor, and A is the intrinsic camera matrix and RT is the extrinsic camera matrix. The matrix A represents how a point in camera coordinates is projected onto the camera image plane. The matrix RT represents the transformation between world coordinates and camera coordinates. Camera calibration is the process of solving for both A and RT, and pose estimation is the process of solving for RT based on a given A.

The most common method for solving for the intrinsic matrix uses multiple images of a planar calibration pattern consisting of a checkerboard [9]. This calibration method also calculates the pose estimation parameters RT — the relative position and orientation of the camera and the checkerboard pattern. Other work that explicitly uses ellipses as part of planar patterns for calibration and pose estimation include Rothwell [6], who calculates a set of specific points using tangent lines and intersection, Song [7], who uses an iterative approach that requires a corresponding pair of conics, and Ji [3] who considers ellipses as one of many geometric primitives (the others are points and lines) and gives a least squares solution for arbitrary sets of matched primitives.

A planar pattern's 2D features can be difficult to extract from oblique camera angles (below 30 degrees). This means that one 2D planar pattern has approximately 1/8 sphere visibility. Non-planar patterns can be accurately measured from a larger set of viewpoints. For instance, Gortler [2] used three orthogonal planar patterns in the scene. This increased the visibility to 1/4 sphere, but it is still far from our hemisphere requirement. Using more planar patterns also introduces other issues like occlusion, pattern identification, and so on. Another drawback of using a planar pattern is that the accuracy of the feature detection depends upon the viewpoint. Generally, when the camera moves away from the perpendicular position, reliability goes down.

Existing approaches also use specific points as the mapping feature when solving for camera calibration. This requires that these points be accurately detected, which can be difficult under changing lighting conditions. A better approach is to use areas of color-contracting regions [8].

Our approach differs from previous work in that we use a truncated 3D cone as the calibration pattern and two ellipses as the mapping features. There are several reasons choosing this pattern. First, the cone has hemisphere visibility in term of detecting the 2D features. Although the cone might be occluded by the object from some viewpoint, we are able to obtain the entire 2D features by ellipse fitting to the visible part of the ellipses. Second, we use the visible potion of the ellipse shape as our 2D feature which means many pixels contribute to the 2D feature detection. Finally, we adopt the stable color ratio technique to ensure robustness to different lighting conditions.

3 Calibration Pattern

Now let us look at the 3D calibration pattern in detail. The pattern is a truncated cone with two marked ellipses (refer to Figure 2). The parameterized function for the cone with radius r and base angle β is:

$$c(t,\theta) = (rt\sin\theta, rt\cos\theta, r(1-t)\tan\beta) \tag{4}$$
$$t \in [0,1], \theta \in [0,2\pi) \tag{5}$$

The ellipse features are the red-green or green-blue boundaries shown on the cone. The position and orientation of the cone in 3D world coordinates is shown in Figure 2. Recall that a conic is the intersection of the cone and a plane. A general equation for a plane is:

$$ax + by + cz + d = 0 \tag{6}$$

For the red-green boundary plane, we chose the following plane equation:

$$0x + y - 5z\tan^{-1}\beta + 2r = 0 \tag{7}$$

Substituting equation 7 into equation 5 and solving for t:

$$t = \frac{3}{5 + cos\theta} \tag{8}$$

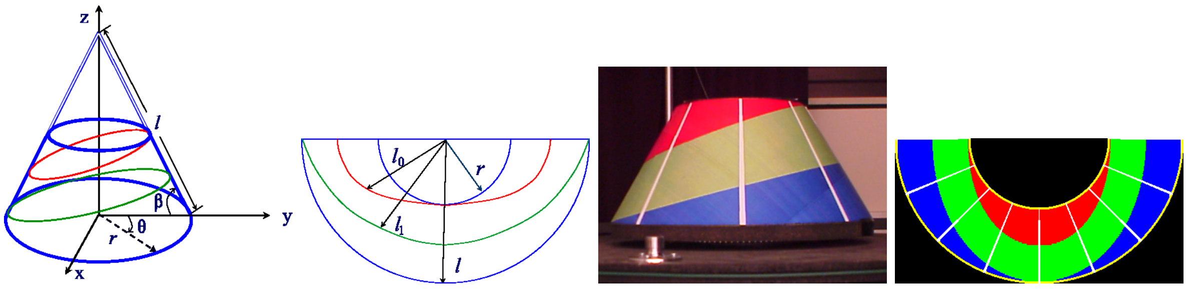

Figure 2: 3D pattern: From left to right: A diagram of cone. A 2D diagram of the "unwrapped" cone. A picture of the actual 3D cone. A picture of the 2D unwrapped cone which we printed out and used to build the 3D cone.

Substituting t into equation 5, we obtain the following 3D ellipse, parameterized by $\theta \in [0, 2\pi]$:

$$e(\theta) = \left(\frac{3r\sin\theta}{5+\cos\theta}, \frac{3r\cos\theta}{5+\cos\theta}, \frac{r\tan\beta(2+\cos\theta)}{5+\cos\theta}\right) \quad (9)$$

We define the green-blue boudary in a similar manner.

How do we build the pattern? If we cut the cone and lay it flat we get a fan which is a 2D planar pattern. We build the 3D pattern by wrapping a 2D planar pattern into the 3D shape. For our choices of β and r the fan occupies 180 degrees of the circle; different choices of β and r will result in wider or narrower angles. The fan can be parameterized by a radius s and an angle ϕ:

$$f(s,\phi) = (s \in [r_i, l], \phi \in [0, 180]) \quad (10)$$

where l is the length of the edge from the apex of the cone to its base and r_i is the inner radius of the fan. r_i is determined by where the cone is truncated.

The unwrapped 3D ellipse can be parameterized by ϕ and a changing radius based on ϕ:

$$e_2(\phi) = (h(\phi), \phi) \quad (11)$$

We first find the radius h in terms of the cone parameter θ. On the cone, the ratio of h's z value over the height of the cone is the same as its ratio to l.

$$h_z = (r\tan\beta - z)/sin\beta \quad (12)$$

We know from the cone equation that

$$z = r(1-t)\tan\beta \quad (13)$$

By substituting in z, we have:

$$h_z(t) = rt\cos^{-1}\beta \quad (14)$$

Substituting equation 8 into equation 14, we obtain:

$$h(\theta) = \frac{3r}{cos\beta(5+cos\theta)} \quad (15)$$

The angle ϕ of the fan is related to the angle θ of the cone by the following equation:

$$\theta = \frac{\phi}{cos\beta} \quad (16)$$

To obtain the explicit 2D function of one unwrapped ellipse we have:

$$h(\phi) = \frac{3r}{\cos\beta(5+\cos(\phi/\cos\beta))} \quad (17)$$

To simplify the pattern we choose β to be 60 degrees. This makes $r_i = r$. Equations for our unwrapped cone boundaries are:

$$\begin{aligned} l_0 &= r & (18)\\ l_1 &= \frac{6r}{5+\cos 2\phi} & (19)\\ l_2 &= \frac{12r}{7+\cos 2\phi} & (20)\\ l_3 &= 2r & (21)\\ \phi &\in [0,\pi) & (22)\end{aligned}$$

Using these equations, we generate the image of the unwrapped cone and print out the pattern on hard paper. We paste it onto a stiff board and wrap it around to obtain the physical 3D truncated cone. To stablize the cone, we paste circular boards at the top and the bottom of the cone.

4 2D feature detection

In this section we describe how to find the 2D ellipses in the images. Recall that the ellipses are defined by

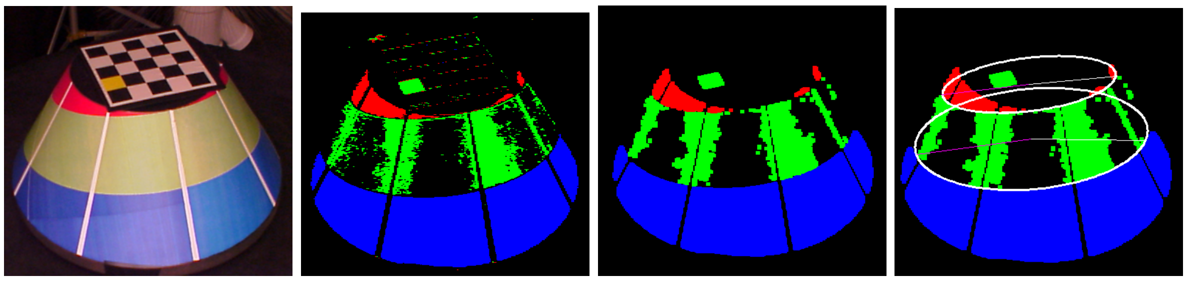

Figure 3: Feature detection. From left to right: Original image. Initial classification. After erosion and dilation. Fitting ellipses.

the color boundary between two colors. To detect the boundary, we need to color classify the pixels, identify the boundary pixels, and then fit them to get the 2D ellipse equation.

Our color classifier uses color ratios which are robust under different lighting conditions [8]. The red ratio value is defined by:

$$r_v = \frac{R}{R+G+B} \tag{23}$$

and similarly for green and blue. If the color ratio value is larger than some threshold, we classify it as that color. Even for images taken under different lighting, the threshold is almost constant (see Figure 3). We use a flood-fill algorithm to create connected regions of similar colors. The boundary pixels are those that lie between two regions.

The simplest method to find the boundary is to look for the pixels that are relatively close to both boundary colors. To speed up this process, we only look at pixels in the connected region.

Some pixels are mis-labeled because of noise or because the object is colored. To eliminate small noisy regions, we perform erosion then dilation. We also use the estimates of the camera's position and orientation to estimate where the object and pattern might be in the image.

We use a linear least squares algorithm to fit a conic to the boundary pixels. A conic is represented by the following implicit function:

$$E(x,y) = Ax^2 + Bxy + Cy^2 + Dx + Ey + F \tag{24}$$

Each pixel results in one linear equation. To prevent the zero solution, we add one additional constraint forcing the sum of the coefficients to be non-zero. We typically have hundreds of pixels. The result of fitting is shown in Figure 4.

Note that we do not require that the fitted conic be an ellipse, although the projection of the 3D ellipse is a 2D ellipse. We restrict our search error function to the visible portion of the ellipse. This means that the conic only needs to fit in the area where we have boundary pixels.

5 Intrinsic Parameters

We use Zhang's [9] algorithm to calculate the intrinsic parameters at the beginning of a capture session. We color one white space yellow in order to uniquely determine the orientation of the pattern.

The checkerboard coordinate system should be the same as the cone's coordinate system. We put the checkerboard on top of the truncated cone and line up the x and y coordinates relative to a physical mark on the cone. We first take one run of the checkerboard and the cone, randomly pick some pictures from the set, and solve for the intrinsic parameters. Keeping exactly the same configuration of the system, we replace the checkerboard with the desired object, and capture the desired image set. We then solve for the extrinsic parameters for each image in the set.

6 Pose Estimation

Using a 3D pattern leads to a nonlinear pose estimation problem. To solve the nonlinear problem, we minimize a cost function representing the accuracy of the current estimation. Because the cost space is not uniform, we use simplex search.

There are six parameters we need to adjust. There are three parameters represent the rotation in the x, y, and z axes. There are three parameters represent the translation in the x, y, and z axes.

To calculate our cost function we begin by taking 100 points evenly distributed along each of the two 3D ellipses. We project these points to the image plane using our current camera pose estimate:

$$V(P) \quad = \quad (u/w, v/w) \tag{25}$$

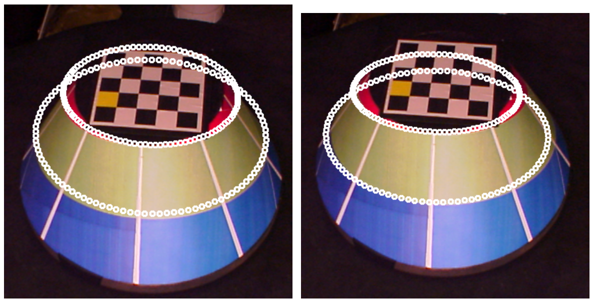

Figure 4: Pose estimation. Left: Original guess. Right: Final fit.

where $m = [u\ v\ w]^T$ comes from equation 1. We then eliminate any points that do not lie near our boundary pixels. Our cost function is then:

$$Err(T,R) = \sum_i E_{rg}(V(P_i))^2 + \qquad (26)$$

$$\sum_j E_{rb}(V(P_j))^2 \qquad (27)$$

where E_{rg} and E_{rb} are the two ellipses found in the 2D feature detection stage.

We obtain initial estimates of the camera's position and orientation from the turntable angle θ and the camera arm angle ϕ (see Figure 1). The camera is assumed start at $(0,0,0)$, oriented down the $z-$ axis. We first translate the camera to the end of the arm $(0,0,-z)$, rotate it around the $y-$ axis by ϕ, then around the $z-$ axis by θ.

The initial guess is usually very close to the correct answer (see Figure 4). We run the simplex search until the cost is below some threshold (10^{-6}). The search converges in ten to twenty steps.

7 Results and conclusions

The metric we used to check the accuracy of the calibration is the geometric pixel error of both the checkerboard corners and the ellipses. For the checkerboard, the distance is the Euclidean distance between the projected corners and the detected corners. For the ellipse, the distance is the average geometric distance from the projected one hundred boundary points to the detected ellipse. We compute the geometry distance from a point to an ellipse based on the approach introduced in "3D Game Engine Design" [1].

We show our approach is efficient and accurate by testing it on a virtual turntable system where both the camera and the pattern are known. Table 1 presents the performance of the simplex search.

	arm angle		
turntable ang.	0–30	30–50	50–80
$0--90$	2.6	2.3	2.8
$90--180$	2.8	2.5	2.8
$180--270$	2.4	2.0	3.0
$270--360$	2.4	2.5	2.6

Table 1: Virtual pixel accuracy by arm and turntable angle

The average error is the geometric distance from the known projected points to the detected ellipse. The table is divided by arm angle (across) and turntable angle (down). This shows that with an ideally accurate and noise-free virtual setup we can use our pattern and approach to yield a reasonably accurate calibration result.

To determine the accuracy in the real world situation, we compare the results of the traditional checkerboard approach and our approach. In our test, the simplex search converged in ten to twenty steps.

Table 2 compares the checkerboard calibration and our calibration from different camera arm angles. The left table shows results for the checkerboard calibration and the right table shows results for our 3D pattern calibration. We show the accuracy of both the checkerboard distance and the ellipse distance.

Table 3 compares the checkerboard calibration and our calibration from different turntable angles. The left table shows results for the checkerboard calibration and the right table shows results of our 3D pattern calibration. We show the accuracy of both the checkerboard distance and the ellipse distance.

From the comparison, we observe that although our approach is based on the ellipse pattern, not only is the ellipse distance minimized but also the checkerboard distance is small. The traditional checkerboard approach might yield a smaller checkerboard distance, but the ellipse distance is worse. Also, we notice that with the traditional approach, there is no calibration data available when the arm angle is less than 30 degrees, while with our approach, we maintain a similar accuracy level at all arm angles and turn table angles.

In conclusion, we have shown that using this 3D calibration pattern efficiently achieves reliable post estimation for all view points from the hemisphere.

arm angle	0–30	30–50	50–80
checkerboard	—	5.3	3.3
ellipses	3.0	2.3	3.1

arm angle	0–30	30–50	50–80
checkerboard	—	3.0	1.2
ellipses	—	2.9	3.4

Table 2: Results of the real-world test sorted by arm angle. Geometric error on checkerboard corners and ellipses Top: based on the checkerboard approach. Bottom: based on our approach.

turntable ang.	0–90	90–180	180–270	270–360
checkerboard	5.1	4.6	4.8	4.4
ellipses	2.9	2.3	3.1	2.7

turntable ang.	0–90	90–180	180–270	270–360
checkerboard	2.1	4.6	1.1	1.0
ellipses	3.4	3.7	4.1	3.7

Table 3: Results of the real-world test sorted by turntable angle. Geometric error on checkerboard corners and ellipses Top: based on the checkerboard approach. Bottom: based on our approach.

References

[1] David Eberly. *3D Game Engine Design.* Morgan Kaufmann Publishers, 2001.

[2] Steven J. Gortler, Radek Grzeszczuk, Richard Szeliski, and Michael F. Cohen. The lumigraph. In *Proceedings of SIGGRAPH 96*, Computer Graphics Proceedings, Annual Conference Series, pages 43–54, New Orleans, Louisiana, August 1996. ACM SIGGRAPH / Addison Wesley. ISBN 0-201-94800-1.

[3] Q. Ji, M. Costa, R. Haralick, and L. Shapiro. An integrated linear technique for pose estimation from different features, 1999.

[4] Jeremy Yermiyahou Kaminski and Amnon Shashua. On calibration and reconstruction from planar curves. In *ECCV (1)*, pages 678–694, 2000.

[5] Long Quan. Conic reconstruction and correspondence from two views. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 18(2):151–160, 1996.

[6] C A Rothwell, A Zisserman, C I Marinos, D Forsyth, and J L Mundy. Relative motion and pose from arbitrary plan curves. *Image and Vision Computing*, 10(4):251–262, 1992.

[7] D M Song. Conics-based stereo, motion estimation, and pose determination. *International Journal of Computer Vision*, 10(1):7–25, 1993.

[8] Andrei State, Gentaro Hirota, David T. Chen, William F. Garrett, and Mark A. Livingston. Superior augmented reality registration by integrating landmark tracking and magnetic tracking. *Computer Graphics*, 30(Annual Conference Series):429–438, 1996.

[9] Zhengyou Zhang. Flexible camera calibration by viewing a plane from unknown orientations. In *ICCV*, pages 666–673, 1999.