Practical Camera Calibration for Large Rooms

Jacky Baltes
CITR, University of Auckland
Tamaki Campus, Building 731, Auckland, New Zealand

Email: j.baltes@auckland.ac.nz

Abstract: This paper describes our practical experiences and methods for calibrating a large room. We show a semi-automatic system to assign real world coordinates to image features. Our system uses a two stage process in which easily recognizable objects (squares) are used to sort the individual data and to find missing objects. Fine object features (corners) are used in a second step to determine the image real world coordinates. An empirical evaluation of the system shows that the average and maximum errors are sufficiently small for our problem domain (autonomous mobile agents playing soccer).

Keywords: Camera calibration

1 Introduction

Our research work focuses on the design of intelligent agents in highly dynamic environments. As a testbed, we use the RoboCup domain, which is introduced in section 2. In this domain, small toy cars play a game of soccer. Position and orientation information is provided by a global vision system.

This paper describes an accurate, cheap, portable, and fast camera calibration system. After an initial preprocessing step (which is guided by the user), it automatically computes real world coordinates for features of the image. Tsai's camera calibration is then used to compute the parameters of the camera model.

Section 5 shows the accuracy that can be obtained by our method in a synthetic and a real world problem. Both the average and maximum error are sufficiently small for our application.

In section 6, we discuss ideas for further research. In particular methods for improving the accuracy and methods for using the color information in a calibration picture to determine color thresholds and regions with different color thresholds.

2 The RoboCup World

RoboCup [1] is a domain initially proposed by Alan Mackworth ([2]) to provide a challenge problem that requires the integration and coordination of a large number of techniques. The problem is to create autonomous softbots and robots that can play a game of soccer.

RoboCup is a difficult problem for a team of multiple fast-moving robots under a dynamic environment that requires the designer to incorporate: autonomous agents, multi-agent collaboration, strategy acquisition, real-time reasoning, robotics, and sensor-fusion. RoboCup also offers a software platform for research on the software aspects of RoboCup.

The RoboCup environment at the University of Auckland consists of a commercially available cheap video camera mounted on a tripod. The video camera is connected to a video server (a Pentium PC). The video server interprets the video data and sends it position and orientation information to other clients on the network (three PCs). The playing field is a rectangular area of roughly three by five meters with a grey carpet. Lighting is provided by fluorescent lamps on the ceiling. All the equipment is readily available and most of the room has been unchanged. Figure 1 shows our environment.

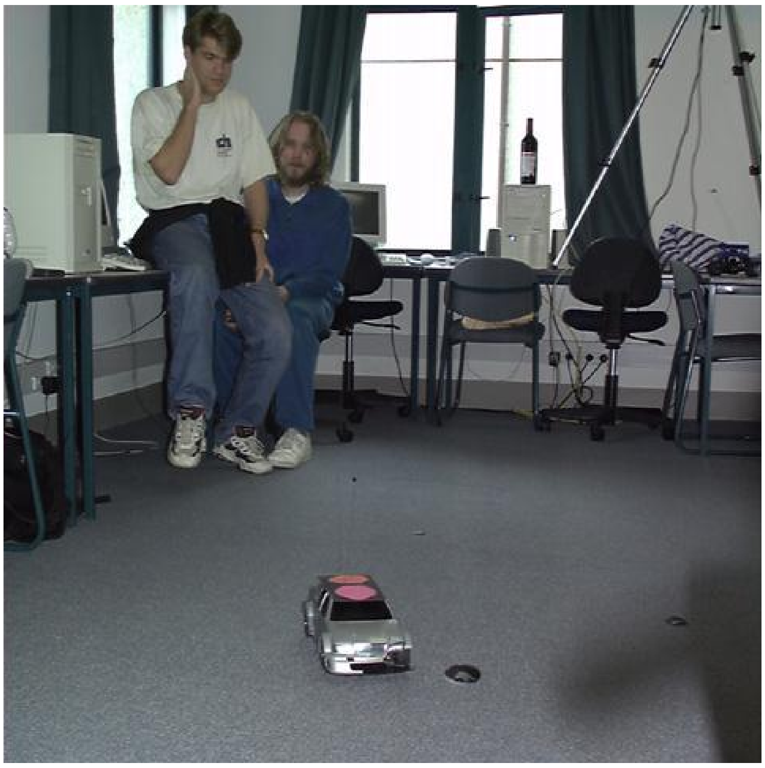

Figure 1: Aucklandianapolis at the University of Auckland. The tripod of the vision system can be seen on the top right corner of the image. The video camera is just out of the picture. The video server determines position and orientation of the cars by bright dots on the car. As can be seen, the speed trials took their toll on our cars.

Since we are often asked to give demos of our system for different occasions, we needed an accurate, cheap, portable, and fast method for camera calibration.

3 Camera Calibration

Traditional camera calibration relies on the availability of known image coordinates for some known world points, that is the real world coordinates for at least 12 image points must be known. Once a sufficient number of matching points have been found, well known camera calibration techniques can be used. For example, the Tsai calibration method uses an eleven parameter model with six external and five internal parameters [3]. In our work, we are using a public domain implementation of the Tsai calibration method, which is available from the WWW [5].

This paper focuses on the problem of finding a suitable set of matching points for camera calibration. The need for portability and speed of the calibration method ruled out traditional methods of using feature points inherent in the scene (since these feature points will not be available when moving to different rooms) or of painting feature points into the scene (a labour intensive and error prone task for a large set of points). The creation of a special calibration pattern of sufficient size and with a sufficient number of points was also too expensive. For example, a large wooden board with calibration points (a) would be difficult to move, (b) may not fit into rooms that do not have similar geometry (e.g., a part of the rectangle is cut out by a wall), and (c) expensive and labour intensive to manufacture.

However, we clearly needed a portable calibration pattern, so we decided to use readily available material. We looked at a number of possibilities including carpets (have a dense texture and are expensive) and linoleum carpets (accurate pattern, but expensive and has an undesirable warping property).

In the end, we decided to use a duvet cover (250x200cm) with a square pattern on it. The back half of the duvet cover was removed to reduce artifacts due to the transparency of the cloth material. The duvet cover is well suited for our environment, since it is easily portable and can be adapted to room outlays[1]. Drawbacks are that the cloth material stretches and warps. Both drawbacks can be minimized through the handy use of an iron. However, they can not be eliminated and thus introduce errors, which limit the accuracy of the camera calibration that can be obtained.

Figure 2 shows a picture of the calibration duvet cover as seen by the video camera.

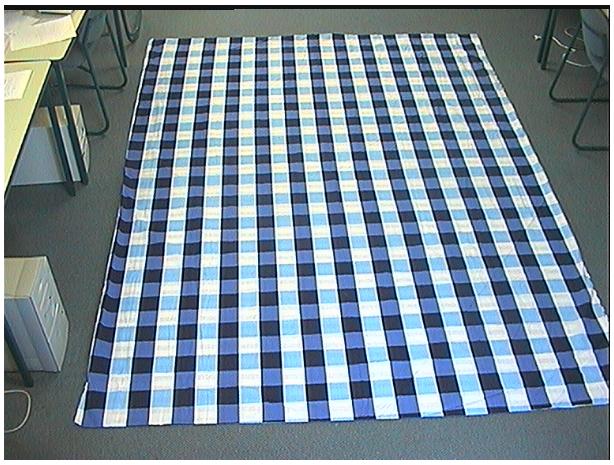

Figure 2: Calibration Pattern as seen by the Camera

4 Find Matching Points Algorithm

Given the picture shown in fig. 2, our system uses a semi-automatic method for calculating the matching points. In the preprocessing step, the user removes unwanted parts of the picture, such as the table top on the left side of the calibration picture. Secondly, the color image is converted into a gray scale image and thresholded, so that only the white squares are left in the image. Currently, we are only a global threshold value on the red channel, which was sufficient for our environment.

After this initial preprocessing step, the system automatically computes the matching points. The idea is to find features in the image that can be assigned world coordinates by the known geometry of the calibration pattern (i.e., by knowing that the dimensions of the squares is $8 \times 8.1cm$). A false color image of the result of the preprocessing step can be seen in fig. 3. The figure shows some of the problems in assigning real world cooridinates to image features: (a) some of the squares are missing from the right side of the image, and (b) some parts of the squares are missing (e.g., in the bottom right corner).

First, the system uses a simple pattern to find the white squares in the picture. This step ignores small artifacts and handles missing squares. The centre point of each square is computed by calculating the moments along the x and y direction. Then, the squares are sorted. This sorting step is of critical importance, since if it is done in the wrong order, the assigned real world coordinates will be wrong, which will result in inaccurate calibration.

The following algorithm `find_real_world` is used to sort the squares and to assign real world coordinates to the centre of the squares. The algorithm takes a unsorted sequence of squares as input and assigns a real world coordinate to the centre of each square. First,the squares are sorted in increasing order of their y coordinate (line 3). This is used to repeatedly extract the next row from the sequence. A row is defined by an initial sequence of squares from `y_sort_squares`, whose y coordinates are within the tolerance limit `eps`. The system also initializes the variable `guess_y`, which is used as a guess of the

[1]It is also a handy blanket for my graduate students when they get caught up in their work and end up sleeping in the lab

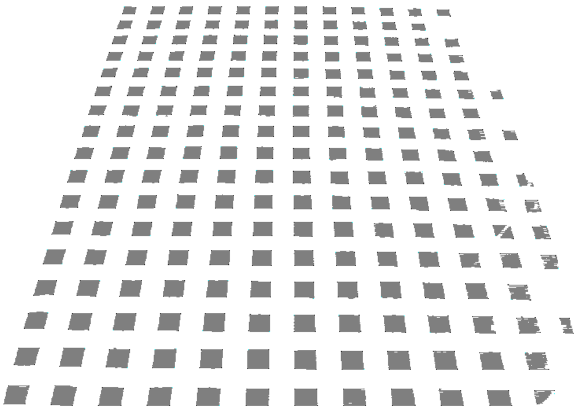
Figure 3: Calibration Pattern after Preprocessing

distance in pixels between the previous and the current row. Lines 8-12 calculate the ratio between the actual average distance in pixels between the previous row to the current row to this estimate. This ratio is used to overcome the problems of missing rows in the input image. The current y coordinate `Wy`, and `guess_y` are updated in lines 13-14. Similarly to the rows, the squares within a row are then sorted based on their x coordinate (line 16) and an x coordinate is assigned (line 24) based on a guess of the distance in pixels to the next square `guess_x` (lines 20-23 and line 27).

```
1  Procedure find_real_world_coors(unsorted_squares) {
2
3    y_sort_squares=sort(unsorted_squares,y-direction);
4    guess_y=0; prev_avg_y=0;
5    Wy=0;
6
7    while (row=extract_row(y_sort_square,eps)!=empty) {
8       avg_y = average_y_coor(row);
9       if (guess_y != 0)
10         factor = round((avg_y-prev_avg_y)/guess_y);
11      else
12         factor = 0;
13      Wy=Wy+factor*SQUARE_Y_DIMENSION;
14      guess_y=avg_y-prev_avg_y;
15
16      x_sort_squares=sort(row,x-direction);
17      guess_x=0; prev_square=null;
18      Wx = 0;
19      foreach square in x_sort_square {
20         if (guess_x != 0)
21            factor=round((square.x-prev_square.x)/guess_x);
22         else
23            factor=0;
24         Wx=Wx+factor*SQUARE_X_DIMENSION;
25         square.realworld_x = Wx;
26         square.realworld_y = Wy;
27         guess_x = square.x - prev_square.x;
28         prev_square = square;
29      }
30      prev_avg_y = avg_y;
31 }
```

Table 1: Algorithm for finding real world coordinates

n	Synthetic Picture			Real picture		
	avg. err	stddev	Max. err	avg err	stddev	Max. err
50	0.9936	0.0653	0.7291	15.2802	7.6748	85.0945
100	0.0964	0.0553	0.3307	17.2455	7.9873	50.0908
150	0.0931	0.0511	0.3068	13.0654	3.8769	37.0576
200	0.0939	0.0557	0.5121	13.8500	5.0923	55.2477
300	0.0904	0.0498	0.3186	13.6753	4.3130	43.3685
400	0.0901	0.0504	0.3207	13.6320	4.2632	56.5799
500	0.0899	0.0497	0.3152	13.5105	3.6942	34.5634

Table 2: Results of the Evaluation

Note that the estimates to the next row and column are adaptive, so this method will work in pictures with obvious perspective distortion (as can be seen in fig. 2) as long as the change from one row to the next is not more than 50%.

After approximate real world coordinates have been assigned to the centres of all squares, the system uses four edge detection steps to find the coordinates of all four corners. If a corner has been identified, it is assigned a real world coordinate by the geometry of the calibration pattern (for example in the first column, the first top left corner has coordinates $0.0, 8.1$, the bottom left corner of the next square is $0.0, 16.2$ and the top left corner of the second square is $0.0, 24.3$.

This means that the assignment of the real world coordinates to the corners is independent of the assigned real world coordinates of the centres of the squares themselves. This is an important feature in our algorithm, since the centres of objects are distorted by the perspective projection and are moved to the lower end of the picture, which means that they are unsuitable for applications that require a high accuracy. Of course, given an accurate camera model, this perspective distortion can be compensated for, but this leads to a chicken and egg problem, since we are using this information to calibrate the camera.

The real world coordinates of the centres are only used for sorting the squares, which means that only their relative values are important to determine, which square is the next square in a row or column or whether a square is missing.

Also we found in our tests that this two-stage approach (sort centre of squares, find corners for each square) works better than assigning world coordinates to all corner points. Missing squares or missing data points makes this one step assignment very difficult and error prone.

After the computation of the matching points, we use a PD implementation of Tsai's camera calibration to compute the parameters of the camera model.

5 Evaluation

We evaluated the system in practice (by calibrating three different rooms on a number of occasions) and quantitatively through the use of a synthetically generated and a real camera picture.

The synthetic picture was generated by computing a perfect image of all feature points (corners of squares) given our current camera setup (camera mounted on a tripod, 1.78m above ground). Since in this case the matching points are 100% accurate, it gives an indication of the maximum accuracy that can be obtained of an eleven parameter camera model.

Given the input image shown in fig. 2, the corner detection finds 815 corner points. The following table summarizes the average error and the standard deviation of the error with increasing number of calibration points n. The data in the table was generated by averaging the results of three cross validation runs for each program. In each test, n points were selected at random. The camera was calibrated with the data from the calibration points and then the average error, standard deviation, and the maximum error (all in millimeters) were computed.

As expected, increasing the number of calibration points improves the calibration of the camera in the synthetic picture. A similar trend can be observed in the real picture.

The differences in errors between the synthetic and the real world image are due to warping of the material in inaccuracies in determining the feature coordinates.

Also, even when using only 150 points, the predictive power of the algorithm is sufficient for our purposes. The error of the calibration is less than $1.3cm$ on average and the maximum error is $3.4cm$. This data is confirmed by testing the accuracy of the coordinates in uncovered areas of the picture (on the very top and bottom of the image). Although, there were no calibration points that covered these ares, the measured error for this region is around $1.5cm$.

6 Conclusion

This paper describes a practical implementation of camera calibration in large rooms. It combines the use of a well known calibration algorithm with a semi-automatic method for computing the matching points.

The method uses a two stage approach. An initial approximation of the centre of an object (in our example squares) are used to sort the objects, but specific features of the object are used to assign real world coordinates. We intend to use feature detection mechanisms with subpixel accuracy, such as the ones described in [4] in the future to improve the accuracy of the calibration.

Currently, only information in the calibration image is used to calibrate the geometry of the camera only. Another problem is the color calibration of the image. The individual cars are detected by our vision server through bright red and green dots on top of the cars.

We found that under different lighting conditions the parameters for the detection routines of the colors had to be adjusted. Another problem is that the lighting over such a large area is not uniform. Areas directly under the ceiling lights are brighter than other areas.

Our next goal in this research is to use the information in the picture to calibrate the geometry as well as the color information in the system. The system would compute the correct thresholds for detecting red and greed dots based on the color of the blue squares in the image. Furthermore, if there are large differences in brightness due to non-uniform lighting, the system will have to break up the room into different parts and compute color thresholds for each region separately.

References

[1] Hiroaki Kitano, editor. *RoboCup-97: Robot Soccer World Cup I.* Springer Verlag, 1998.

[2] Alan Mackworth. *Computer Vision: System, Theory, and Applications*, chapter 1, pages 1–13. World Scientific Press, Singapore, 1993.

[3] Roger Y. Tsai. A versatile camera calibration technique for high-accuracy 3d machine vision metrology using off-the-shelf tv cameras and lenses. *IEEE Journal of Robotics and Automation*, RA-3(4):323–344, August 1987.

[4] Robert J. Valkenburg, Alan M. McIvor, and P. Wayne Power. An evaluation of subpixel feature localisation methods for precision measurement. In *Videometrics III*, volume SPIE 2350, pages 229–238, 1994.

[5] Reg Willson. Tsai camera calibration software. WWW, 1995.