Enhanced Camera Calibration for Machine Vision using OpenCV

Article *in* International Journal of Artificial Intelligence · September 2014

CITATIONS
0

READS
301

1 author:

Shubham Rohan Asthana
Birla Institute of Technology and Science Pilani
5 PUBLICATIONS **21** CITATIONS

SEE PROFILE

Enhanced Camera Calibration for Machine Vision using OpenCV

Shubham Rohan Asthana[1]

Abstract-In several machine vision applications, a fundamental step is to precisely determine the relation between the image of the object and its physical dimension by performing a calibration process. The aim is to devise an enhanced mechanism for camera calibration in order to improve the already existing methods in OpenCV. A good calibration is important when we need to reconstruct a world model or interact with the world as in case of robot, hand-eye coordination. In order to meet the rising demands for higher accuracy various calibration techniques have been developed but they are unable in obtaining precise results. In this paper we propose an enhanced camera calibration procedure using a special grid pattern of concentric circles with special markers. The overall objective is to minimize the re-projection for good camera calibration.

Keywords: Camera Calibration, Concentric Ring Pattern, Machine Vision, Image Processing.

1. INTRODUCTION

Camera calibration is a necessary step in 3D computer vision in order to extract metric information from 2D images. It has been studied extensively in computer vision and photogrammetric. The main aim of camera calibration is to determine the parameters of the transformation between an object in 3D space and the 2D image observed by the camera from visual information such as images. The transformation parameters include:

- Extrinsic parameters: The camera's location and orientation in the world which is denoted by rotation and translation of the camera.

- Intrinsic parameters: The relationship between pixel coordinates and camera coordinates.

One of the main uses of camera calibration is to figure out where a camera was in relation to a scene in a photograph. One can also use camera calibration to take an image sent to a computer, and figure out where various coordinates are in the real world. This type of deduction is crucial to the functioning of robots that are meant to interact visually with the physical world. These robots can then use a photographic or video input, device and calibrate in order to figure out where objects it sees might actually be in the real world, in actual terms of distance and vector.

Camera calibration can also be used to figure out other things about the camera in relation to the scene [6]. For example, using formulas we can figure out the focal length that the scene was shot at. We can also figure out the skew factor of the image, and any lens distortion that may have been introduced, creating a pincushion effect. We can also figure out whether the actual camera pixels were square or not, and what the horizontal and vertical scaling factors for the pixels might have been.

1. *Shubham Rohan Asthana is student, of Computer Science and Information Systems Department, BITS Pilani, K.K Birla Goa Campus India Zuarinagar, Goa-403726, India*
Email: asthana.st.francis@gmail.com

The calibration procedure typically consists of either localizing the calibration pattern control points such as square corners [1] or circle centres [2] and then solving for the camera parameters, or using some geometric property of the pattern itself to solve for the camera parameters directly.

However, a major source of error that affects this camera calibration approach of either localizing the control point or using geometric properties of the pattern directly, is that the input camera calibration images are non-fronto parallel images (different orientations with respect to the camera) that suffer from nonlinear distortion due to camera optics and the perspective the camera lens receives. Therefore, precise localization of control points or accurate determination of geometric properties under such conditions is a very difficult task, where even small errors may lead to imprecise camera calibration.

In this paper, the intension is to reduce this error by a considerable amount by using a grid of concentric circles with special markers. It helps in the accurate positioning of control points. This uniqueness in determining orientation reduces the error that arises due to imbalance in construction of the calibration object. Edge detection, ellipse fitting and contour detection is used to locate these concentric circles, the weighted mean of their radii gives the circle centres as the control points. Camera parameters are obtained from these accurate control points. The structure of this article is as follows. In section 2 we summarize the related work in area of camera calibration. Next, in Section 3 we describe the proposed approach for calibrating the camera. Section 4 covers the experimental results which are followed by conclusion and future work covered in Section 5.

2. RELATED WORK

The initial work in the field of camera calibration started with the use of squares as control points by Tsai [3] in which used 60 control points to determine the camera position and orientation parameters along with focal length and radial lens distortion. Tsai's calibration approach presumes that some camera parameters are provided by manufacturer to minimize the initial guess of the estimation. In order to solve the calibration problem it requires n feature points per image to solve a set of n linear equations which are based on radial alignment restraint.

Zhang [1] first time proposed to use a planar checkerboard pattern which required only a few set of images to calculate the calibration parameters. The proposed technique is such that either the camera or calibration pattern can be moved freely irrespective of direction of motion which is insignificant to the algorithm. The approach only requires the camera to observe a planar pattern shown at a minimum of two different orientations. The approach consisted of an initial closed form solution of the camera parameters, followed by a nonlinear refinement using Levenberg-Marquardt.

Chen et al. [4] describes a calibration procedure that uses only a single image of two coplanar circles with arbitrary radius. Their proposed approach aims to calculate both the focal length and extrinsic camera parameters simultaneously. One drawback of this approach is that the features it relies on cannot handle the non-linear distortion by itself because it is not possible to distinguish if the difference in the feature parameters comes from the distortion or from projective effects.

Heikkila in [5] uses solid circles as control points and uses a non recursive technique for reversing the distortion model. The algorithm performed a minimization over the weighted sum of squared differences between the observation and the camera model using Levenberg-Marquardt.

The techniques employed by Tsai, Heikkila and Zhang's all use the pinhole projective model to map three dimensional scenes to the two dimensional camera image plane.

3. PROPOSED APPROACH

There are two major sources of error that have adverse effects on calibration results. The first one is the imperfection of the calibration object. Since the assumptions made for the conventional camera calibration are based on a perfect planar target with ideal patterns, the imprecision of the calibration target may lead to inaccurate results. The second problem is the uncertainty in locating the control points directly from the geometries of the calibration patterns in the captured raw images which suffers from lens distortion as well as perspective distortion [2].

In order to deal with the above errors we propose to use a lens distortion model which takes into account tangential and radial distortion .and using a frontal image concept achieves precise localization of control points.

We take an input is a set of 24 images, each having different orientations of the pattern with respect to the camera. The pattern is of special type wherein there is a grid of concentric rings. Each such unit is made up of four circles with the exception of three units that have five concentric circles. These three patterns also called the marker points of the image and are positioned at the corners of the image. Each marker point can be uniquely identified based on their mutual distance and hence acts like the coordinate axis which helps to define an orientation of the image. (Refer Figure 1) .Such a pattern helps to distinguish between two images more accurately as in this ring pattern vertical and horizontal rotations meant different images, which was not so in chessboard .

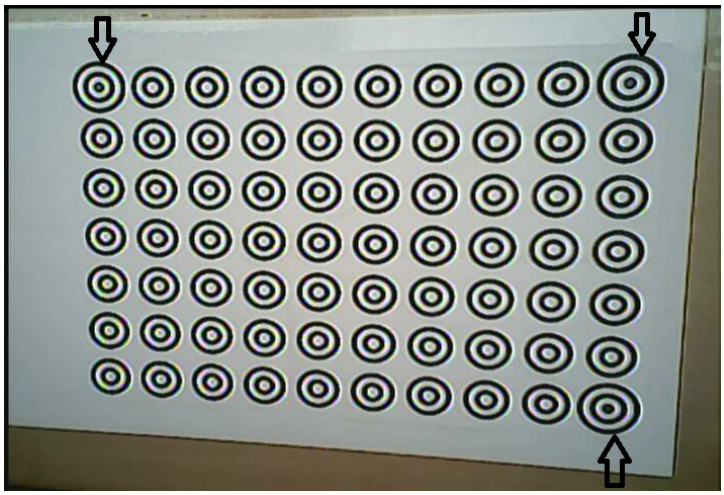

Figure 1.Grid of concentric circles (One of the input images. Arrows denote the three marker circles)

The uses of concentric circles help improve the calibration procedure. This is due to the fact a single,

large concentric circle image may, however, suffer from nonlinear distortion leading to inaccurate determination of geometric properties of the concentric circle, and therefore, inaccurate camera calibration. A grid of concentric circles, on the other hand, is less susceptible to distortion effects.

This approach comprise of these steps:

1. Detection of circles in each image
2. Using the detected circles, finding the center of each of the concentric ring pattern
3. Segregating these centers as marker or non-marker points
4. Arrangement of all the center points in a systematic order that is universally followed for all images
5. By the use of these ordered center points, calibrating the camera
6. Creating the frontal image for each image by the use of the parameters derived from camera calibration

3.1. Detection of Circles in Each Image

The first task involves devising an algorithm to detect these circles with minimum possible error margin. For this purpose, there are two general approaches:

a. Hough Transform

The Hough transform [7] is a feature extraction technique used in image analysis, computer vision, and digital image processing. The purpose of the technique is to find imperfect instances of objects within a certain class of shapes by a voting procedure. This voting procedure is carried out in a parameter space, from which object candidates are obtained as local maxima in a so-called accumulator space that is explicitly constructed by the algorithm for computing the Hough transform. We used a function called cvHoughCircles which available in OpenCV. In order to use this function, pre-processing is required on the images. This pre-processing includes:

-**Thresholding**, which is the simplest method of image segmentation. Image segmentation [8] is the process of partitioning a digital image into multiple segments (sets of pixels). The goal of segmentation is to simplify and/or change the representation of an image into something that is more meaningful and easier to analyze. Here the image is divided into foreground and background super pixels. From a greyscale image, thresholding can be used to create binary images. In OpenCV, this is implemented using *cvThreshold function.*

-Edge detection [9] is the name for a set of mathematical methods which aim at identifying points in a digital image at which the image brightness changes sharply or, more formally, has discontinuities. The points at which image brightness changes sharply are typically organized into a set of curved line segments termed edges. In OpenCV, this is implemented using *cvCanny* function.

After this pre-processing, when Hough transform was implemented faulty results were achieved like detection of no circles , drawing inappropriate circles with greatly deviated centers or drawing multiples circles on one circle.(Refer to figure 2) Such errors were due to the fact:

-The function *cvHoughCircles* asks for a parameter which is the minimum distance between centers of two circles to be detected and should be a non-zero positive number. Hence, such procedure cannot be used to detect concentric circles as these circles have zero distance between their centers.

- Since the images were inclined, the rings present in the image were no more circular, but instead elliptical (An inclined circle when seen from front view appears like ellipse).

Thus, an algorithm was required that detects ellipses as that procedure would be generalized since circles are also special cases of ellipse wherein eccentricity=1 (i.e. major and minor axis are equal to radius).

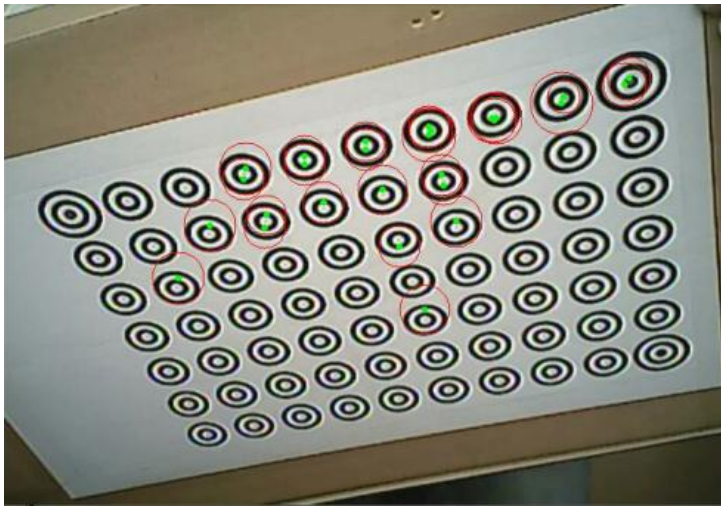

Figure 2. Faulty and no circles detected in the pattern using cvHoughCircles

b. Ellipse Detection

Due to the drawbacks of Hough Transform, second approach was carried out for the purpose. It involved:

- First the raw image needs to be loaded, followed by pre-processing. The pre-processing involves converting the raw image to greyscale (using *cvCvtColor* function) and Gaussian blur it (using *cvBlur*).
- Thresholding was applied on the pre-processed image using the *cvThreshold* function with a threshold value of 100. This value was used due to the fact that in the images which were inclined by a very large angle with respect to the plane of projection, the furthermost points after thresholding with other values were becoming almost blur or was getting diffused with other concentric rings.
- Using *cvFindContour* function on the thresholded image, the contours present in the image were found. Contour tracing is one of many pre-processing techniques performed on digital images in order to extract information about their general shape. These contours are saved separately in contour vector.
- Found the minimum bounding rectangle using *cvMinAreaRect* and minimum bounding ellipse using *cvFitEllipse* function. (Refer to figure 3)
- Centers of these fitted ellipses are used as centers for further process.

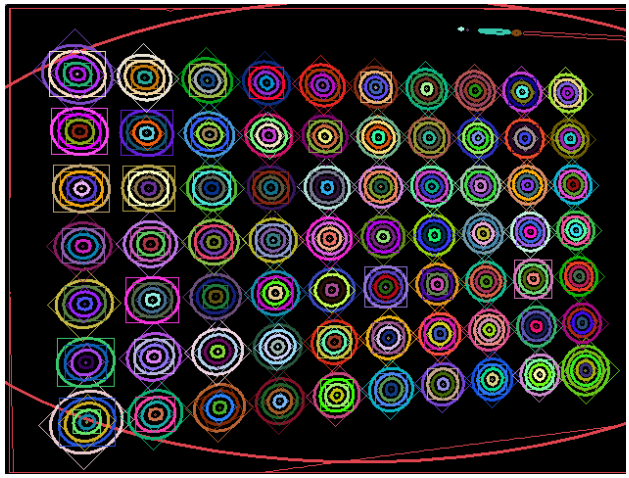

Figure 3: Detecting circles by fitting ellipses and minimum bounding boxes over the detected contours in the image.

3.2. Using the Detected Circles, Finding the Center of Each of the Concentric Ring Pattern

First a linked list of centers of all the binding ellipses was created having a user-defined data type with attributes as X and Y coordinates of the center, count of the number of circles whose weighted mean would be taken to find final center values and a value that is proportional to cube of area of binding ellipse. This value would be used as weight for finding the weighted mean. The push and pop functions were especially designed for the purpose.

The push function was implemented in such a way that if the linked list was empty then just adds the points as nodes else traverse along the list and check whether any already existing point was at an epsilon distance from the point. This epsilon value depends on the image, that this distance is less than the difference between centers of adjacent circles. If true, then took the weighted mean of the already existing point and the passed point based on cubic power of a value. This value is proportional to area of the binding ellipse of each contour. Such a weight was chosen in order to give more weight to outer circles than inner ones in each pattern. This is due to the fact that since the inner circles were too small and if the plane inclined, their size shrunk further. Thus detection of outer circles would be more appropriate than inner ones. The weighted mean is calculated by maintain the summation of all the cubic weights and summation of coordinate value and its weight divided by summation of weights. Whenever a new point is pushed and a mean need to be calculated, the weighted mean value is multiplied by the summation of weights, then the new point is added to this value after multiplying with this value. The summation is updated by adding the weight of the newly added point and the new mean (calculated in the last step) is divided by this summation. The pop function was written in such a way that nodes having count value that is the number of circles whose weighted mean were taken to find the final center values, was not either marker_points or non_ marker_points. This was due to the fact that the marker points in the image were composed of marker_points circles and other ring patterns had non_marker_points circles. Thus by first pushing all detected center points in the linked list and then popping out the nodes having count other than non_marker_points or marker_points, center of the circles of the pattern were detected with significant level of accuracy. (Refer to figure 4)

. The first approach is to find the distance between each marker, these three distances are expected to be different, and hence equality can be used to distinguish between markers. But because of the different projections and perspective errors, the inequality is violated for images with extreme orientations. The second approach is to find number of points between each marker.

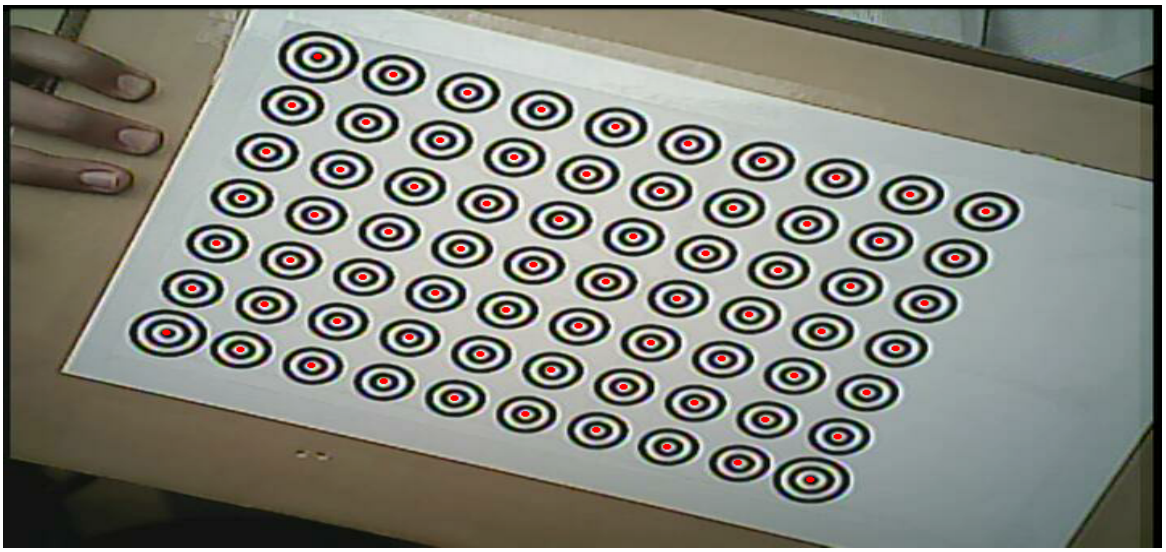

Figure 4: Detected centers of the circles in the pattern

Three equations are possible for three markers. There are different numbers of points lying on each line for every image, though it is difficult to specify a concrete number for every image due to difference in orientation, a common inequality exists. If the three markers were to define a coordinate system: Y-O-X , then the number of points lying on OX is more than the number of points lying on OY which is in turn more than the points lying on YX. It is important to take into account the possibility of an epsilon deviation while testing whether a point lies on a line or not. After distinguishing the different points from each other, they are stored in a two dimensional array. These are further used for calculating slope for arranging the centers.

3.4. Arrangement of All the Center Points in a Systematic Order that is universally followed For All Images

After uniquely identifying each marker point, each image has a fixed orientation which aides in determining the direction in which the points need to be traversed and stored. The algorithm requires the first step to be a **Delaunay Triangulation** over all centers detected. In geometry, triangulation is the subdivision of a 2D geometric object into triangles. A Delaunay triangulation [10] for a set P of points in a plane is a triangulation DT(P) such that no point in P is inside the circum-circle of any triangle in DT(P) Delaunay triangulations maximize the minimum angle of all the angles of the triangles in the triangulation; they tend to avoid skinny triangles.

The output of Delaunay triangulation using predefined procedures of OpenCV is a set of edges. Followed by Delaunay triangulation, the points on the OY axis are ordered based on their distance from the origin marker and are stored in an array. For each element in this array in order, the whole row of centers along the OX axis is traversed. For this, initially all centers are included in a linked list. According to Y-O-X orientation, the ordering needs to start from the origin marker. After the origin marker, its adjacent markers need to be added; to accomplish this, the application traverses through all the edges of Delaunay triangulation to find edges with source or destination as the origin marker.

All these edges are selected and their other endpoints (which can be a source or destination for the edge) are put in another linked list. From this linked list, only those points are selected whose respective edges have slope similar to OX line (with a required epsilon deviation). A more efficient method is to match the x and y components of the OX line, as this takes into consideration the error induced by differently oriented images .It also helps to deal with vertical lines as tangent of 90 degrees tends to infinity, hence dividing into sine and cosine components helps to get rid of this problem as the values lie between -1 and 1. Another round of filtering is made of the selected points. The selected points in the linked list are made

to traverse through the big linked list. If the point is present, then that is the next point. The original starting point is popped out from the big linked list and starting point is shifted to the detected point. This procedure is iteratively applied on all center points. In this way, one row is completed.

After completing one row, the next point on the array of OY points is pushed into the final arranged list and the similar procedure is iteratively applied. This is to be done on all points on OY line. The algorithm aims to arrange all the points from origin in the rightward direction (Taken the fact that the origin lies in the bottom left corner). Thereafter these arranged center points are stored in the array called corners having type CvPoint2D32f. This is the required format for calibration. Once the center points are ordered in a universally followed manner for each image, which is the format required by *cvCalibrateCamera2* procedure, the OpenCV procedure *cvDrawChessboardCorners* is used to join the detected centers in orderly format. (Refer to figure 5) These arranged points act as image points needed for the calibration procedure. (The array of corner points is saved as the image points). The program uses the dimensions of the calibration pattern to obtain the object points. (Refer to figure 5)

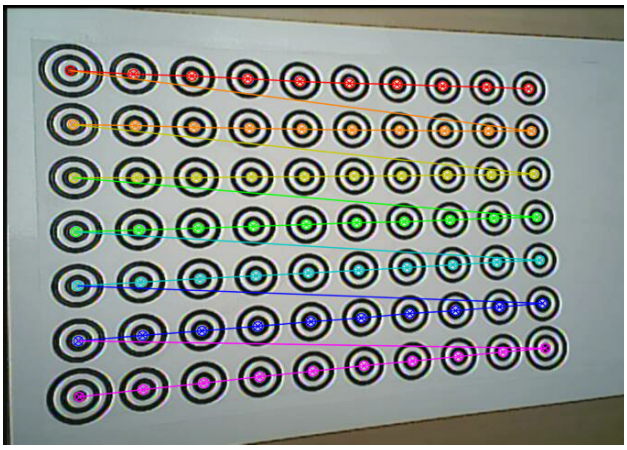

Figure 5: Drawing the arranged center points starting from the origin marker points moving leftwards with respect to image frame.

3.5. Using Ordered Centers, To Calibrate the Camera

After determining the image points and object points the camera is calibrated using *cvCalibrateCamera2* procedure. This function calculates re-projection error (equal to approximately 0.297 in the first case). It also calculates the intrinsic matrix and distortion coefficients. Using the distortion coefficients, each image is undistorted. This undistorted image is used for finding the frontal image.

3.6. Creating the Frontal Image for Each Image by the Use of the Parameters Derived From Camera Calibration

The orientation of the object relative to the camera coordinate system is described in terms of a rotation and a translation. Using *cvFindExtrinsicCameraParams2* function, the application finds the *rotationVector* & *translationVector* for each image by passing the intrinsic matrix and distortion coefficients. Once the camera is calibrated, the camera intrinsic parameters are used to obtain the extrinsic parameters for each image .These extrinsic parameters i.e. Rotation vector and Translation Vector are required to obtain the transformation matrix. Rotation matrix is obtained from rotation vector using Rodrigues Transform which is applied via cvRodrigues2 function. Then the transformation matrix is computed manually by augmenting the rotation matrix and translation vector .The undistorted image which was obtained using distortion coefficients, is passed to *cvWarpPerspective* function, which generates the frontal image by applying the transformation matrix on each image. (Refer to figure 6).

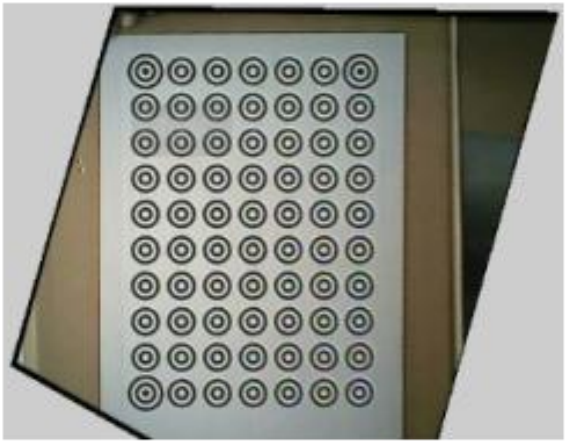

Figure 6. Frontal Image of Ring Pattern

The quality of the frontal image obtained by application of cvWarpPerspective procedure depends a lot on the interpolation method adopted.

Interpolation [11] is way through which images are enlarged. It is the process of determining the values of a function at positions lying between its samples. It achieves this process by fitting a continuous function through the discrete input samples. This permits input values to be evaluated at arbitrary positions in the input, not just those defined at the sample points. Thus it is best if the quality, or visible distinction for each pixel, is retained throughout the enlargement process.

The process of interpolation is one of the fundamental operations in image processing. The image quality highly depends on the used interpolation technique. Several interpolation techniques have been developed and the most commonly used methods are the nearest neighbour, linear, area and cubic techniques.

1. Nearest Neighbour
The simplest interpolation from a computational standpoint is the nearest neighbour, where each interpolated output pixel is assigned the value of the nearest sample point in the input image. This technique is also known as point shift algorithm and pixel replication. This technique achieves magnification by pixel replication and minimisation by sparse point sampling. For large-scale changes, nearest neighbour interpolation produces images with blocky effects.

2. Linear Interpolation
A better algorithm than "nearest neighbour"; that takes into account the gradual transition of pixel colour values. By finding the means between two pixel values, the filler pixel is better suited for overall image enhancement. In other words, it just looks plain better. Linear interpolation is a first degree method that passes a straight line through every two consecutive points of the input image. After linear interpolations, edges are blurred. To remedy this, spline interpolation is used. After interpolation, the edges are more visible so edge enhancement is much more successful and visible. Edge detection works by taking a weighted sum of pixels around a single pixel to determine its new value.

3. Area Interpolation
It is re-sampling the image using pixel area relation. It is preferred method for image decimation that gives moiré-free results. In case of zooming it is similar to nearest neighbour interpolation method.

4. Cubic Interpolation
Cubic interpolation is the simplest method that offers true continuity between the segments. As such it requires more than just the two endpoints of the segment but also the two points on either side of them.

In image processing, cubic interpolation is often chosen over linear interpolation or nearest neighbour in image re-sampling, when speed is not an issue. In contrast to linear interpolation, which only takes 4 pixels (2x2) into account, cubic interpolation considers 16 pixels (4x4). Images re-sampled with bi cubic interpolation are smoother and have fewer interpolation artefacts. The interpolated surface is smoother than corresponding surfaces obtained by linear interpolation or nearest neighbour interpolation. Cubic interpolation can be accomplished using either Lagrange's polynomial or cubic convolution algorithm.

4. EXPERIMENTAL RESULTS

When the traditional calibrating approach was applied on the Ring Grid pattern a re-projection error of 0.2976 was found along with the intrinsic matrix and distortion coefficients. It is worth noting that the re-projection error of the traditional calibration over Ring grid pattern was significantly more than that of chessboard which we were able to obtain around 0.218. This is due to deviation of control points i.e. the centers of the concentric circles from the centers of the detected ellipses because of non-frontality of images. Another reason for this error could be the lack of sub-pixel accuracy. This is because in the raw image which has various orientations and the perspective view that the camera receives, the centers of the concentric circles do not coincide. The centers of the concentric circles of a single set are perceived as two different points because of non-linear distortions. The different centers do not even coincide with the actual control point. To prevent this, frontal images are obtained which are free of distortions, thus giving us the accurate control point.

Given below are the xml files of the intrinsic parameters and distortion parameters achieved by calibrating ring grid pattern. (Figure 7 and Figure 8)

```
<?xml version="1.0"?>
<opencv_storage>
<Intrinsics type_id="opencv-matrix">
  <rows>3</rows>
  <cols>3</cols>
  <dt>f</dt>
  <data>
    7.33987732e+002 0.              3.14758331e+002
    0.              6.57016235e+002 2.37581253e+002
    0.              0.              1.</data></Intrinsics>
</opencv_storage>
```

$$\begin{bmatrix} x \\ y \\ w \end{bmatrix} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix}$$

Figure 7: Snapshot of the xml file of intrinsic parameters achieved by calibrating ring grid pattern

```
<?xml version="1.0"?>
<opencv_storage>
<Distortion type_id="opencv-matrix">
  <rows>5</rows>
  <cols>1</cols>
  <dt>f</dt>
  <data>
    1.24356449e-001 -2.14284241e-001 8.12645070e-003 -9.64543596e-003 2.06597671e-001
    </data></Distortion>
</opencv_storage>
```

$$\text{Distortion}_{\text{coefficients}} = (k_1 \quad k_2 \quad p_1 \quad p_2 \quad k_3)$$

Figure 8: Snapshot of the xml file of distortion parameters achieved by calibrating ring grid pattern

5. CONCLUSION AND FUTURE WORK

The calibration procedure suggested in this paper utilizes concentric circular control points and performs mapping from world coordinates into image coordinates and backward from image coordinates to the three dimensional plane. With this proposed approach we have been able to achieve a re-projection error of 0.297. In order to improve our results and as a future work to this we propose to use iterative calibration that utilizes the parameters obtained from indigenous calibration algorithms as initialization to perform undistortion and unprojection of calibration images to obtain a frontal image. This frontal is then used to localize the calibration pattern control points, reproject them back on raw image and recomputed the camera parameters in an iterative refinement until convergence. Undistorting and unprojecting the calibration pattern to the frontal plane increases the accuracy of control point localization and consequently of camera calibration. After obtaining the frontal images, the calculated positions of the control points in the frontal images are used as initial guess to refine the positions using the iterative calibration method. This is because they are fronto-parallel and do not suffer from distortion effects. Then reversely project the detected control points in the frontal images back to the raw images. Re-optimize the camera parameters together with the world coordinates of the control points. This iterative refinement approach is bootstrapped using standard calibration routine of OpenCV, which provide initial estimates for radial distortion and camera parameters.

Our work has helped to derive the inefficiency of Hough Transform algorithm for circle detection. The algorithm cannot be used to determine concentric circles because it requires a minimum positive distance between the centers of the set of circles to be detected. Moreover, most of the images are actually perceived as ellipses due to their orientation. Hence, though the calibrating object is a ring grid pattern, Hough Circle Transform does not assist in the calibration procedure. Hence the algorithm designed by our project team would be beneficial in finding the circles with almost negligible error margin.

6. REFERENCES

[1] Z. Zhang. A flexible new technique for camera calibration PAMI 22(11), 2000.

[2] Ankur Datta, Jun-Sik Kim, Takeo Kanade,Accurate Camera Calibration using Iterative Refinement of Control Points

[3] R. Tsai. A versatile camera calibration technique for high-accuracy 3d machine vision metrology using off-the-shelf TV cameras and lenses. IEEE JRA, 3(4), 1987.

[4] Q. Chen, H.Wu, and T.Wada. Camera calibration with two arbitrary coplanar circles. In ECCV, 2004.

[5] J. Heikkila., Geometric Camera Calibration using circular control points. PAMI, 22(10), 2000.

[6] Johan C. van den Heuvel, Jan C.M. Kleijweg, Wannes van der Mark, Christiaan M.Lievers,Leon J.H.M. Kester, Obstacle Detection For People Movers Using Vision And Radar .

[7] http://en.wikipedia.org/wiki/Hough_transform

[8]http://en.wikipedia.org/wiki/Segmentation_%28image_processing%29

[9] http://en.wikipedia.org/wiki/Edge_detection

[10]http://en.wikipedia.org/wiki/Delaunay_triangulation

[11] http://en.wikipedia.org/wiki/Interpolation