EE362 Final project

Efficient Measurement and Correction techniques for Lens Distortion

Abhishek Das
Dinesh Patil

Introduction

Consumer digital cameras have been criticized, mostly because of their unacceptable geometric distortions. These effects show disturbingly on architectural and similar shots, as shown below in figure 1. Geometric distortion is an error on an image, between the actual image coordinates and the ideal image coordinates. Among various such nonlinear distortions, the radial distortion, which is along the radial direction from the center of distortion, is the most severe part and is also the most common. Straight lines in the undistorted subject (a window) bulge in the characteristic barrel fashion, in the image. Straight lines running through the image center remain straight and a circle concentric with the image center remains a circle, although its radius is affected.

Figure 1: Test image taken from canon PowerShot S30 showing barrel distortion

The most typical cases of distortion, barrel and pin-cushion distortion, are primarily radial in nature, with a relatively simple one or two parameter model accounting for most of the distortion. A cost effective alternative to an expensive lens is to correct for the distortion using the model. A microprocessor or DSP could be used as the implementation platform for such an algorithm if processing were done offline. However, most users will like to see the images already corrected, thus saving operational costs. A fixed hardware approach would be ideal, provided it is cheap and fast.

This paper reports on a non-linear model of radial distortion across 5 different consumer digital cameras using simple measurements. We also present an efficient technique of its removal, based on these coefficients, which can be incorporated directly into the firmware of the digital cameras with minimal use of extra memory. We finally present near perfect images, obtained by using this correction algorithm and the simple mathematical model.

Background

A standard lens gives a natural field of view that is not subject to perspective distortion because it's focal length matches the sensor or film format being used. The exit angle of the image matches the entry angle, as shown figure 1. When a lens with a field of view greater or smaller than this is used, the entry and exit angles are no longer the same, and this causes perspective proportions to become distorted.

With a wide field of view the image has to be squeezed into a smaller space, as shown in figure 2(b). The result is that scale and distance proportions between foreground and background are increased, objects appearing smaller and distance greater the further away they are. When using a lens with a narrower field of view, a telephoto lens, the image needs to be stretched to fit the space, as shown in figure 2(c). The perspective is again distorted, but in the opposite way. Scale and distance proportions between foreground and background diminish so that objects far away appear larger than they are and nearly as big as those close by, and it appears there is virtually no distance between them.

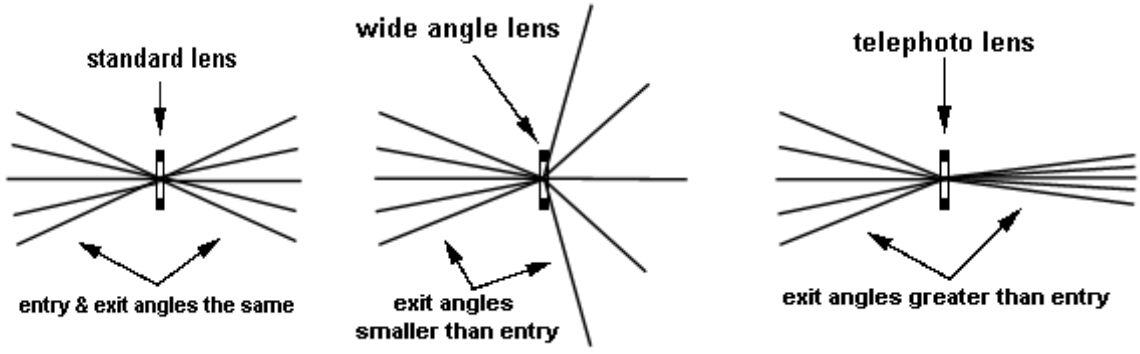

Figure 2: Entry and exit angles for a) standard, b) wide-angle and c) telephoto lens

Using lenses that provide wide or telephoto fields of view bring with them certain other problems that have to be taken into account, besides that of perspective, in order to get the best use out of them and produce worthwhile shots, and some of these relate to general lens design.

Barrel and Pin-cushion distortion

Two basic lens distortions that occur are called *pin-cushion* and *barrel* distortion because that is the effect they have upon an image. Symmetrical lenses, as exemplified by the standard lens, have complementary elements at the front and back, which cancel out distortion, but telephoto and wide-angle types, which are asymmetrical, do not. Barrel distortion is found in wide-angle views because it tries to squeeze the image in a smaller space, while pin-cushion distortion is found in telephoto since it tries to stretch the image to fit the space. The squeezing and stretching of images vary radially due to the design of the lenses, making these distortions visually most prominent at the image corners and sides. These distortions (exaggerated) are shown below in figure 3.

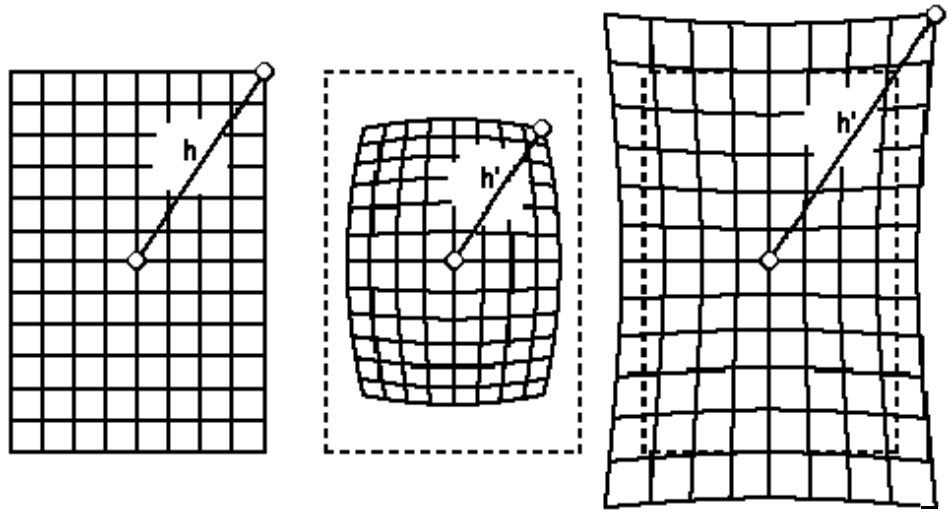

Figure 3: a) straight lines, b) barrel distortion and c) pin-cushion distortion

Corrective Lens elements are used to reduce these faults as much as possible and some lenses exhibit much less distortion than others. The quality of a lens, and it's type, single focal length [prime] or zoom, usually determine how much distortion occurs. Prime lenses tend to produce less distortions because there are fewer elements and less need for optical compromise and it can be optimized for it's particular focal length, while a zoom involves many elements and some compromise, and the wider the focal range, the more compromise is involved. All lenses produce aberrations, but they are hardly seen in the very best lenses, whilst in the cheaper types they may be quite prominent.

These distortions are commonly corrected by first applying a parametric radial distortion model, then estimating the distortion coefficients, and eventually correcting

the distortion. Most of the existing works on the radial distortion models can be traced back to an early study in photogrammetry, where the radial distortion is governed by the following polynomial equation (r_u: undistorted radius, r_d: distorted radius):

$$r_d = f(r_u) = r_u + k_1 r_u^3 + k_2 r_u^5 + \ldots.$$

Method

To correct for the distortion, one needs to model it. For this we need the fitting function that will analytically estimate the amount and kind of distortion and then we need known regular patterns for which we can accurately measure the deviation of the image obtained from the real one.

Modeling Function

Geometric distortion is a typically given as a function of the radial distance of the point from the center of the image (which is assumed in our study to coincide with the center of the lens). In this project we are assuming that there is negligible de-centering distortion. Theory predicts that the distorted radius of a point r_d is a polynomial in terms of the undistorted radius r_u as given above.

Note that $f(r_u)$ contains only odd order terms. In general real lenses will deviate from this ideal relationship and the distortions they produce might not be accurately represented by this formula. Hence some researchers have also tried to include some even order terms. Highly accurate data is needed for such second order effect modeling. This means using the lens to image on an extremely high resolution imager and exactly centering it and so on. The common cameras we had could not provide any of these conditions. So we modeled using the first and the third term only. This has the advantage of being the most robust modeling (as was confirmed by our experience described later) since it fits the general trend of the data points rather than their actual positions which is already imprecise due to discretization by the pixels. Also from inspection it is clear that a negative value of κ_1 means the lens is causing barrel distortion while positive value means its pincushion. Hence the problem of modeling the distortion is reduced to that of finding the right κ_1 that fits the distorted image.

Modeling Patterns

We used two regular patterns for our modeling as shown in figure 1(a,b) below. We actually printed out three, and had planned to use the radial one for the actual measurement since we can measure the relative difference between the radii of the different circles and make our model. However, rendering a circle on a square grid of pixels is already inaccurate and it was not required at the end, as we

developed a better method for accurately finding k_1 from the grid and the check board pattern itself.

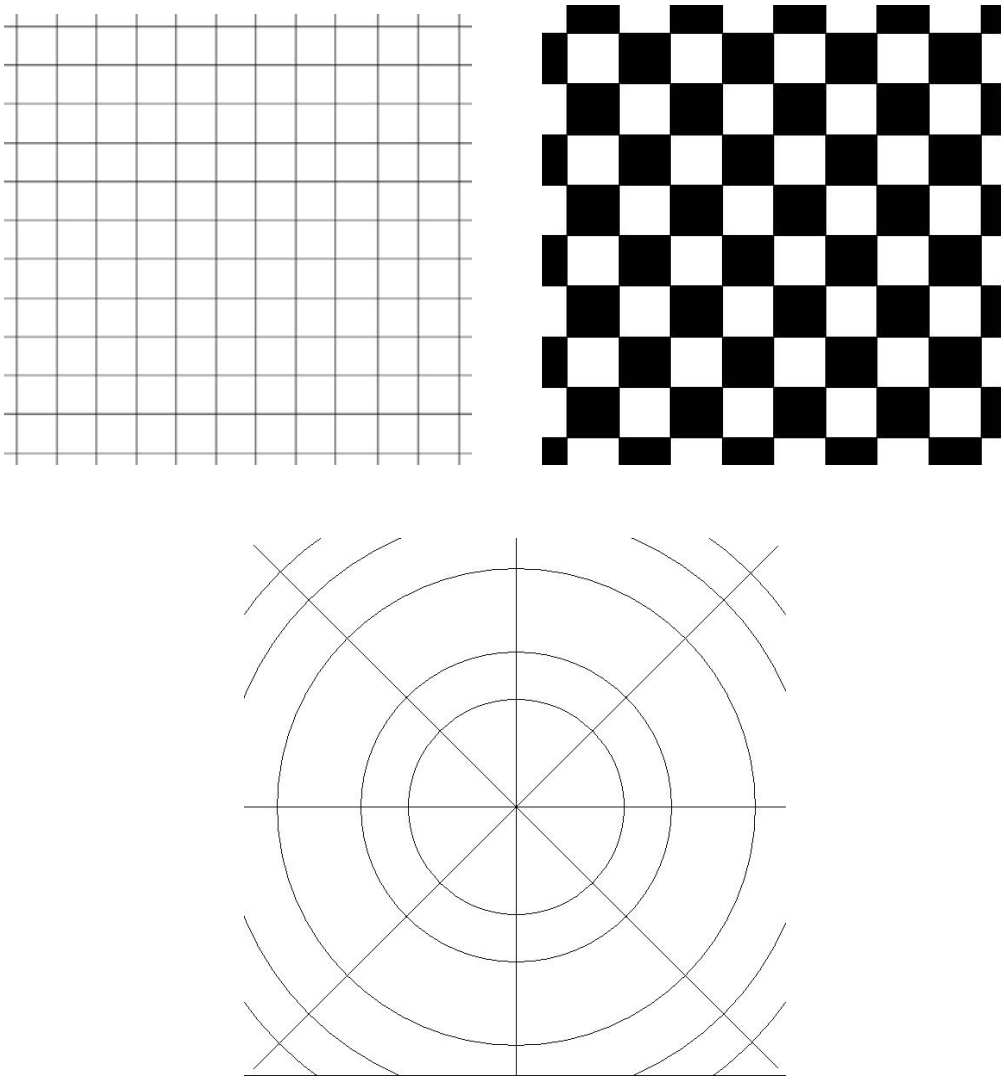

Figure 4: Testing Image patterns

Modeling Algorithm

We had to do a lot of experimentation and correction before we got to the final modeling algorithm that was robust and needed the least human intervention. Here we present the final algorithm here. We shall describe the details of how we arrived at it later in the discussion section. The algorithm flow chart is shown below.

1. Get the center and the corresponding X axis of the image (the horizontal line through the center)
2. Find the trend of pixel intensities to measure the vignetting. Accordingly we can make a cut off point to detect whether a pixel belongs to a grid line or white surface.

3. Get the center of the grid lines by finding the mean of the position of the pixels that make up each dip of the intensity.
4. Having obtained the centers pick a set of consecutive grid lines and find the distance of each from the center along the X axis (this is the r_d values
5. Fit the values with a polynomial in r_u by assuming that at low r, $r_u = r_d$.
6. Let the grid spacing be "a" (an unknown variable). Formulate the problem in terms of finding k_1 and a. Solve it using MATLAB's "fsolve" function or use least squares. "fsolve" will need a starting point so a couple iterations will be necessary.
7. Once you have k_1 check to see that your assumption in 5 was correct.

Description

The basic idea is to take a photograph P from the Camera Under Test (CUT) and do image processing on it to find out the data points necessary for distortion. Let P be the image in figure 5. The data points here mean a set of r_us and a corresponding set of r_ds. Of course, we already have a set of r_ds since we have the distorted image P.

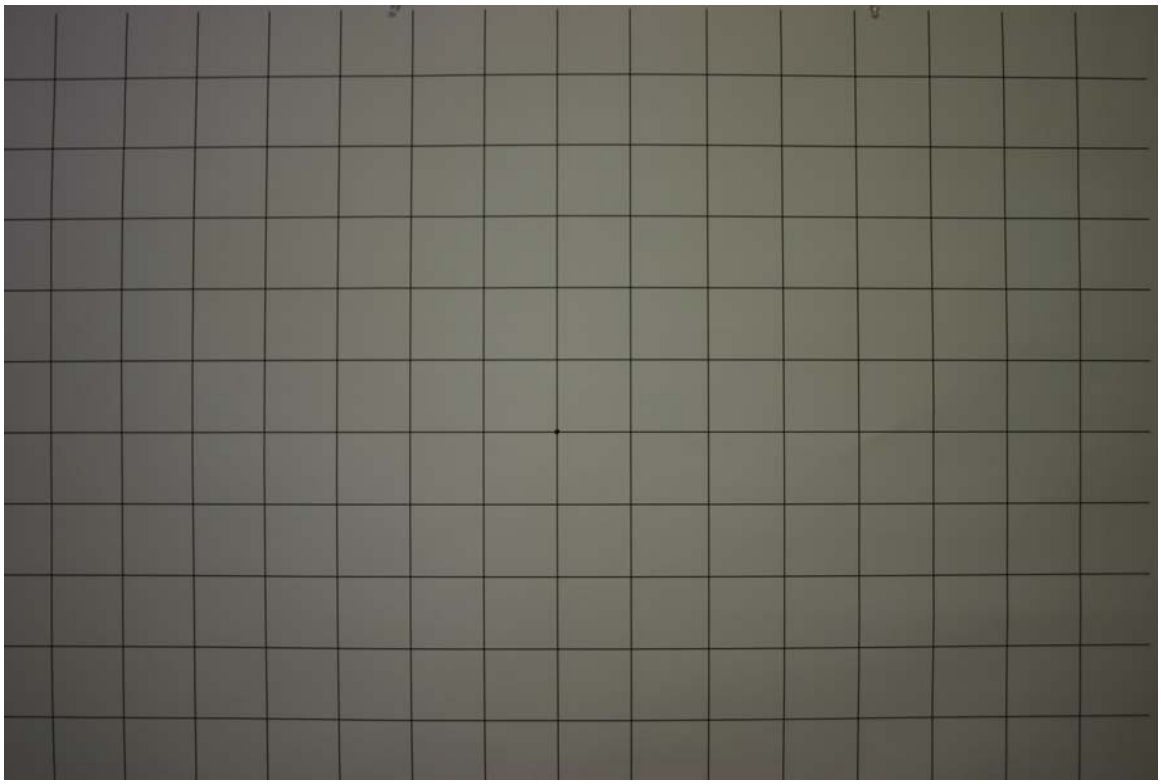

Figure 4: An image of a regular grid suffering from geometric distortion and vignetting. Taken by a canon EOS digital rebel.

To find r_u, one would have to overlap the actual imaged area on P (to remove the scaling factor) and then see the relative difference. This is not only complicated but also ambiguous, since the boundary of P is not well defined. It is already a scaled version of the real boundary and so on. Hence absolute measurements are not possible.

But differential measurements can be done. Note that due to distortion, equal radial distances on the actual image will result in unequal radial distances in P. The most easiest way to use the sizes of the grid squares on P. Now if we take the image such that the squares are aligned with the sides of the image and consider the center of the image as the origin, then they align with the X and Y axis, and if measure along the X or Y axis, we would be measuring along the radius. By measuring the distance between the grid lines we can find out how the squares shrink as we go radially outwards along X or Y. Since the X dimension is wider we measure along the X axis. The problem is reduced to finding the center of the grid lines.

Figure 6 shows the plot of pixel intensities along the X axis, the dips representing the lines. The circles represent the pixel locations. Clearly there are multiple pixels per dip and there is a problem with identifying the center. There is also peaking near the edges (which is mainly due to the inverse DCT operation of JPEG). In a few cameras, which provided raw data, this peaking was very minimal. The geometric distortion effects are minute and hence one needs to have very accurate difference measurements in order to get the right polynomial fit. To get the centers one can take the average depth by considering only the pixels below a certain intensity.

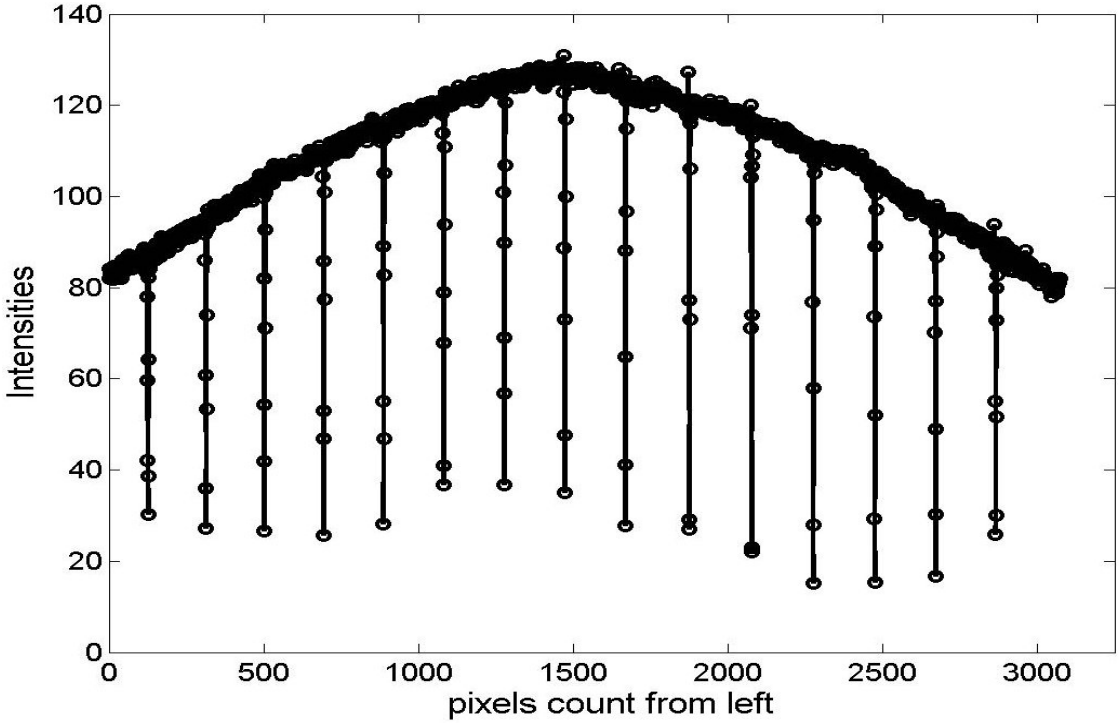

Figure 5: Pixel intensity along the X axis of the image

Due to significant vignetting, we have to change the threshold of choosing the pixels according to the region we are in. This can be done if we have the idea of the average change in the illumination across the X axis. This is achieved by low pass filtering the intensities using moving averages over a 100-200pixels. The averaging bin size is of the order of the resolution of the grid in pixels and so depends on the image resolutions and the number of gird squares on the axis. To get a smooth plot, we had to do this twice. Since the averaging for the first pixel is the average of the next 100-200 pixels, with vigettting it is intuitive that it is values will be beefed up. The result of this phenomenon is to make the average curve shifted with respect to the actual trend as shown in figure 7.

In order to get it right, we need to shift it to the right by the moving average bin size, but preserve its curvature. We do this by extrapolating it on the left using MATLAB functions. Having obtained that we can lower this curve by a certain level (in the order of 10-15 pixels – depending on the maximum noise level you would expect after considering artifacts of DCT used in JPEG compression). This is shown in figure 8. It is important that the moving average bin size be of the order of grid size distance, else it can lead to wrong extrapolation.

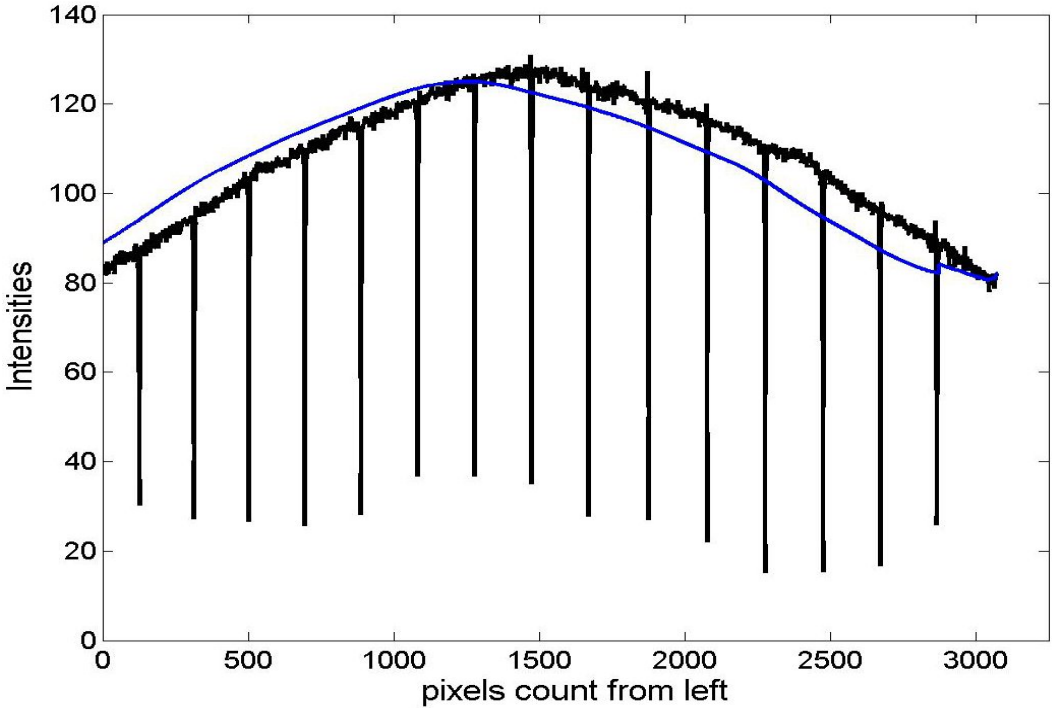

Figure 6: Effect of low pass filtering on the pixel intensities (blue line)

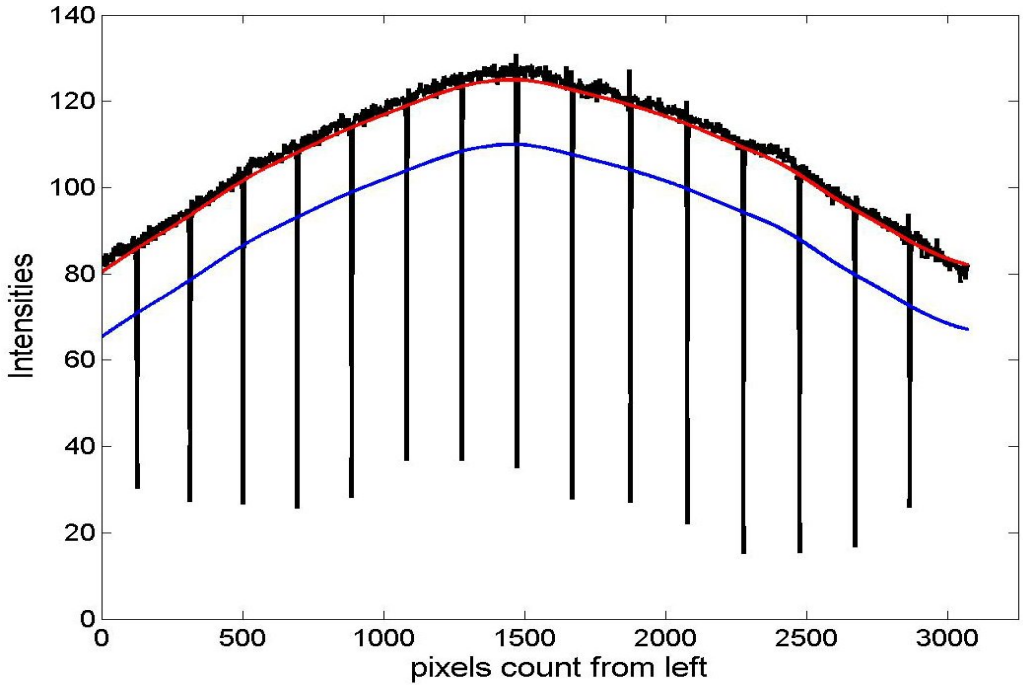

Figure 7: The average curve shifted from figure 6 (in red) and the cut off curve (in blue)

Having done that we can then take the distance along the X axis of various grid intersections, after finding the center of the lines by averaging the chosen pixels per dip. Thus we now have a set of r_ds. We do not have the corresponding r_us but we know that the different r_ds represent the distance from origin of equally spaced lines. The figure plots the distance between neighboring lines w.r.t their number from the origin (this plot is for one side of the image only). Assuming the grid spacing to be *x* pixels and also assuming that at distances closer to the center the geometric distortion is not big enough so that we can have $r_u = r_d = r_1$ for the first grid line we encounter we can formulate the problem as shown below.

Let R1, R2, R3, R4, be the measured increasing radial distances of the grid lines from the origin. The grid lines are spaced at resolution of "a" pixels (a is unknown). Also as per our assumption R1 is undistorted. So then we can find the distortion coefficient k_1 by solving

$R1 + a + k_1 * (R1 + a)^3 = R2$
$R1 + 2*a + k_1 * (R1 + 2*a)^3 = R3$
….. for R4, R5 etc.

This can easily be solved using the "fsolve" command in MATLAB. Although there are only two variables viz. k_1 and a, we can have an over-constrained problem so that the data fits the general trend of the curve. Instead of consecutive points, one can also take alternate points (if one has enough dips to play with).

From the result obtained, we can test our assumption about r_1. Indeed it turns out that the difference between r_u and r_d for the first grid line is less that 1% of a pixel. Since our resolution of measurement is already limited to half a pixel at best, our assumption is pretty reasonable. Another small point is that since most images have an even number of pixels in the X and Y dimensions, we have 4 pixels at the center rather than 1. This puts us off by half a pixel in the Y direction when we choose any of these 4 points. Hence we choose the center (while taking the picture) such that approximately, at the location of the first grid line, $x >> 0.5$. This basically implies that $x \approx r$. Hence for us origin is the center of some grid square in the middle of the image rather than at the crossing of grid lines.

Correcting Algorithm

The modeling gives us a formula for finding r_d as a function of r_u. For our case here, it would seem natural that we need to take an inverse to have r_u as a function of r_d, because what we have is the distorted image. But this is not necessary, though some references have done that [XXX references]. All we need to do is to allocate the pixels at radial distance r_u from the image center (which could now be a virtual point and not a pixel), the intensity of the pixel that is at a distance r_d along the same angle. This means finding x and y locations of the pixel whose values is supposed to be allocated to the current pixel under consideration. The relation

$$r_d = f(r_u) = r_u + k_1 r_u^3$$

is equivalent to

$$x_d = x_u(1 + k_1 r_u^2), \text{ and}$$
$$y_d = y_u(1 + k_1 r_u^2)$$
$$\text{where, } r_u^2 = x_u^2 + y_u^2$$

Thus all we need to do is to go around the image and substitute the pixel value in every (x_u, y_u) by the value in the pixel at the distorted location (x_d, y_d).

Image Rendering

Creating the corrected image from P can be done without spending extra space on creating the copy of P, by modifying P in place. In this case though, its necessary to ensure that pixels do not get overwritten before their value is needed in other pixels. This can happen if, say in case of a barrel distortion, pixel “a” is overwritten by another pixel that is radially closer to the origin and then we move to a radially outer pixel “b” that needs the value of “a”. It will get the new value of “a” which is incorrect.

Thus depending on the type of distortion the modification of the pixels starts form outer to inner (in case of barrel distortion) or the other way around for pincushion. A few image rendering patterns are shown in the figure 9 for barrel

distortion. For pincushion one would render in the opposite direction. Since typical commercial cameras read images by rows or columns the patterns in figure 9(b) and 9(c) are better suited for actual real time application. Also in figure 9(b) (c), its important to note that inside a row or column, the rendering is from outside to inside for barrel.

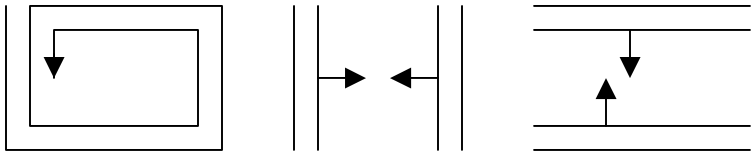

Figure 9: Few rendering patterns (a) (b) and (c)

Because of radial symmetry the actual calculation of radius and the distortion coefficient has to be done only on one quadrant of the image and corresponding symmetric points in the other three quadrants can be processed simultaneously.

Results

We used the above algorithms on 5 different cameras and found coefficients for telephoto and wide-angle lens position. The values of different distortion coefficients (k_1) are listed in table 1. Some manual intervention is needed to find the right set of points to consider for fitting to the polynomial, because of inherent measurement noise and also to give an initial point to the algorithm. Also we observed that consistent with theory, the distortion coefficients
k_1 start out negative (barrel distortion) for wide angle imaging and then progressively turn positive (pincushion) at telephoto setting. The cameras considered have on an average 3x optical zoom. Also the cameras that started with high barrel distortion give less or no distortion at telephoto while those that started with smaller barrel distortion show slight pincushion at telephoto. Thus there is a tradeoff between distortion and zooming capabilities of the camera. The Toshiba camera gave both kinds of distortion at telephoto. So we were not able to characterize it properly.

Table 1: Results for 5 popular consumer digital cameras

Name of the camera	Image resolution in Mpix	Distortion coefficient k_1 for tele-photo	Distortion coefficient k_1 for wide-angle
Canon PowerShot S30	3.2	-1.1e-8	-1.36e-8
Olympus Camedia C-2040	2.1	~0	-7.344e-9
Toshiba PDR-M25	2.2	+/-	-1.173e-8
Sony	5.1	7.741e-9	-2.014e-9
Canon EOS digital rebel	6.2	-2.69e-9	-7.303e-9

We used the k_1s obtained from the grid pattern to correct the distortion in the check board pattern image. The result was a reasonable straight image after correction. A corrected image of figure 5 is shown in figure 10.

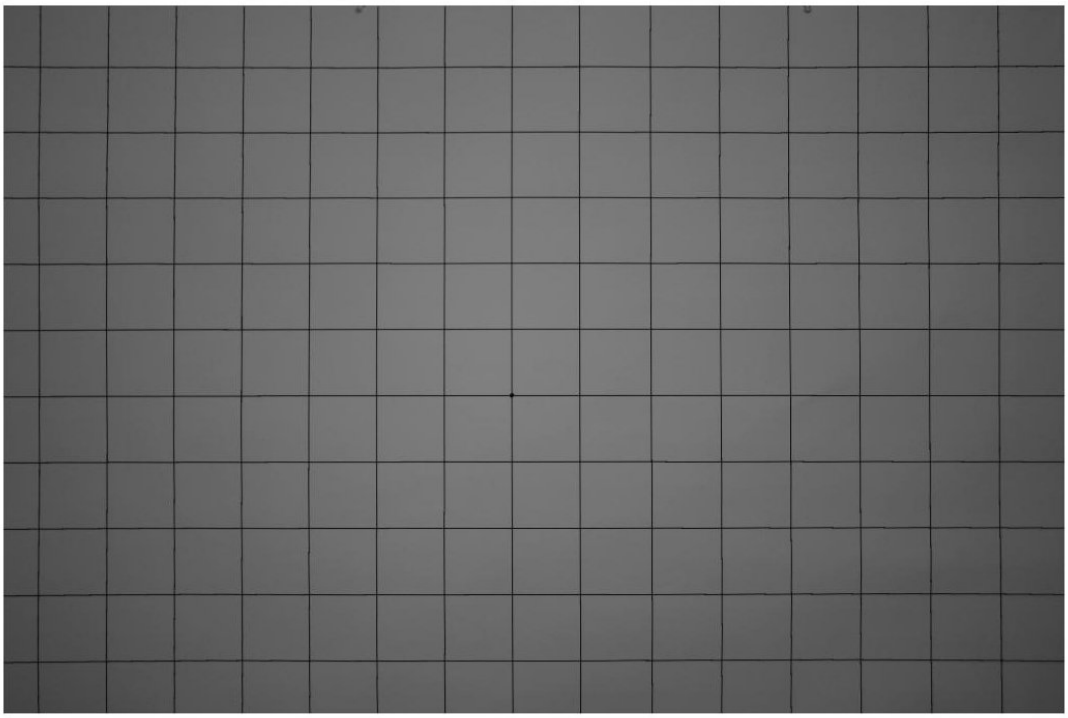

Figure 10: Correct image for test pattern in figure 5 (Canon EOS digital rebel)

We also show in figure 11, the grid pattern and corrected grid pattern for Canon PowerShot S30. Notice that the hue has changed between the original and the new image, since to obtain geometric distortion we converted the original image that was in color to Y image using the YCbCr formula for Y. Hence the corrected image is more B/W than the original one.

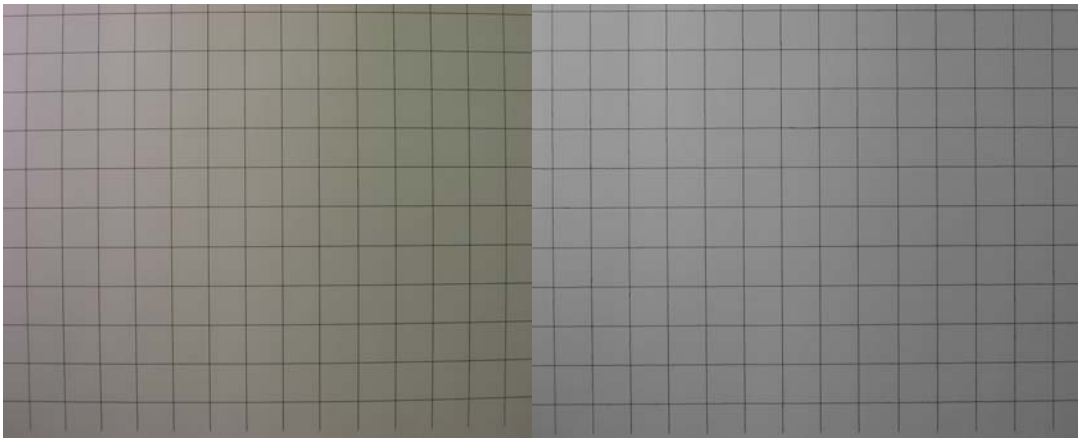

Figure 11: Original (left) and corrected (right) image from Canon PowerShotS30. Notice any difference?

The corrected image of the window taken from the PowerShot S30 camera is also shown in figure 12.

Figure 12: Corrected window image using the distortion coefficient for canon power shot S30.

Implications of image rendering using ASIC

The distortion parameter k_1 changes with the focal length. But its value does not vary by huge amount. In most case (unless the zoom is really large) it is okay to have a few selected values of k_1 for different focal length regions. Having selected k_1, it is clear that its not necessary to calculate the radial distance and the correction factor and so on every time. All this can be calculated during the calibration of the lens and the *x y* mapping between the pixels can be stored in a memory. Since only a quarter of the image is needed for calibration, and given a common camera with 3x zoom, say we have 4 values of k_1 for various focal length settings, we can save the *x y* mapping table in the almost the same size memory as would be needed to store a single image (since the quarter and the 4 cancel each other). This is not a huge amount by today's standards. Once the image is captured, the processing can take place in real time as it now only involves storing the pixel values at different location than the one they came from. No processing is involved. Thus without the need of a frame buffer, one can just relocate the pixels to correct for geometric distortion.

Discussion (What worked and what did not)

While rendering the image after having the distortion coefficient is relatively simple, we had a tough time calibrating the camera for obtaining its value. At first we started with an image on A4 sheet and found it to be inadequate for filling up the image of a common camera unless we get the camera too close, which caused blurring of the edges. Then we set up a projector and decided to take the image of that, but the projector seemed to have its own artifacts in terms of color aberration and limited resolution. Finally we got a huge poster printout with the three test patterns shown in figure 4. We have been using that for all our measurements.

We first started with the check board pattern, thinking that detecting edges would be easier than detecting centers. The pixel intensity plotted along the X axis is as shown in figure 13.

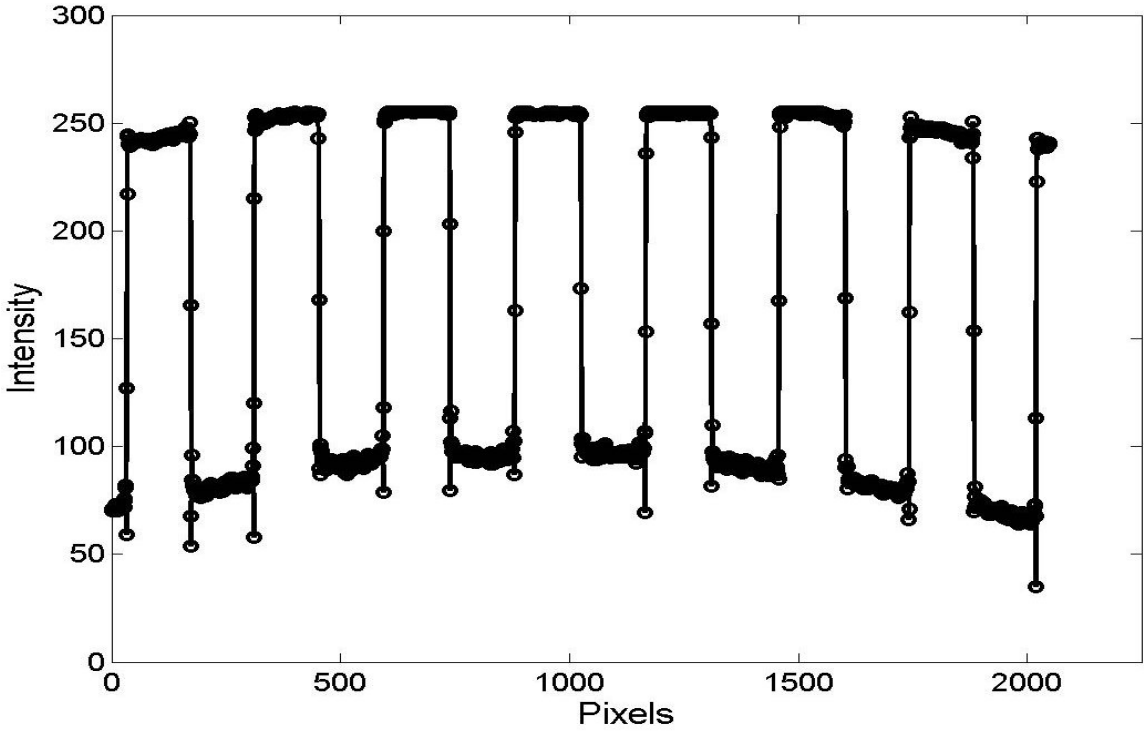

Figure 13: Pixel intensity pattern for check board image

As we can see, here it is important to find the exact crossover points. In presence of vignetting, it was then necessary to find the average cross over point. Considering a check board pixel intensity patter as a special case of grid pattern with thicker lines, we notice that it is much harder to average using simple moving averages because the dips are so huge. Also there are edge effects resulting from the JPEG-DCT transformation that affect the accuracy of the cross over point. In case of grid lines, the spike is symmetric around the center of the grid line and thus the error introduced by them minimal while measuring the center. This is not the

case with check board, where on either side of the transition the spike is in opposite direction. We used a crude eye estimation to get the change of the average intensity. That did not give us a good trend of block widths, so we got an accurate estimation with a MATLAB script. With this we could find the cross over point by interpolating between two pixels whose values lied on either side of the average. The resulting check board square sizes starting from the left to right are shown in figure 14.

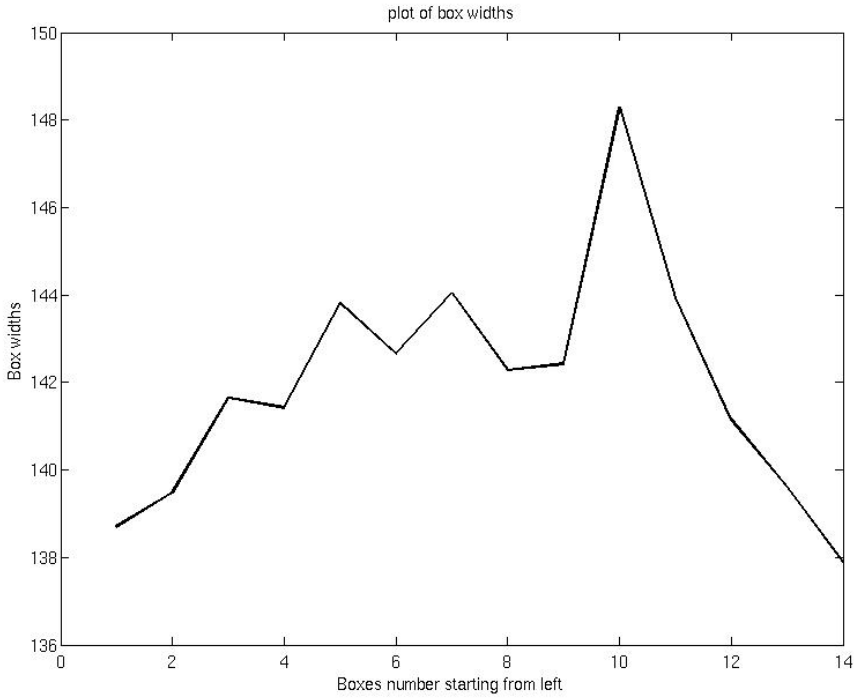

Figure 14: Check board square widths for squares counted from left to right. Note that the center of the image is around box number 7.

There are two observations to make. The jig saw pattern indicates that the widths measured for the black and white squares are different (though they respectively follow the same trend). Thus it is difficult to fit a polynomial here, unless we consider a black and white square together as one unit. This reduces the number of data points by half, which is bad. The reason is simply due to the JPEG-DCT algorithm that causes spiking differently for different transition direction. Secondly we could not understand why there is a sudden peaking of the block widths, whereas what we expected was to see the widths reducing on the other side of the center as well. After noticing the same effect with many cameras, we finally measured the square on the poster board to find to our big surprise that the poster printer had printed the squares unequal! Thus all the measurements were correct, they just had to be normalized to the actual size of the square. A direct fit on some such highly inaccurate data resulted in either weird patterns or over correction. We also tried using more terms in the model to accurately follow the trend of points, but that is not advisable given the accuracy of obtained data. Over modeling over a wrong data

can give weird images (figure 15). Other cases gave us slightly over corrected images (figure 16).

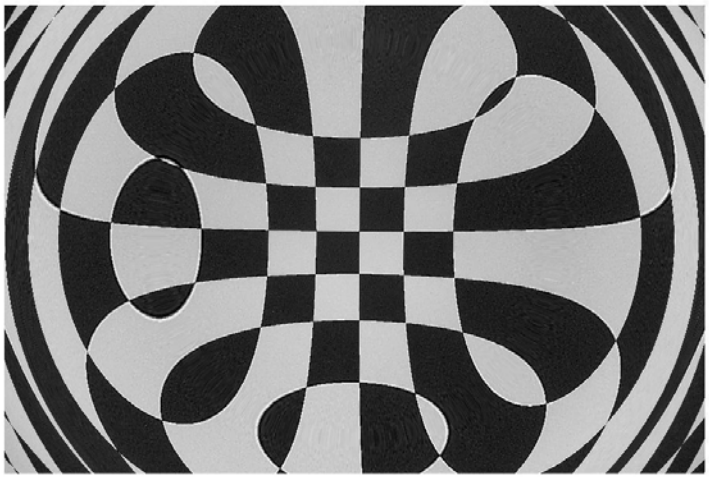

Figure 15: Wacko Image generated due to wrong polynomial fitting

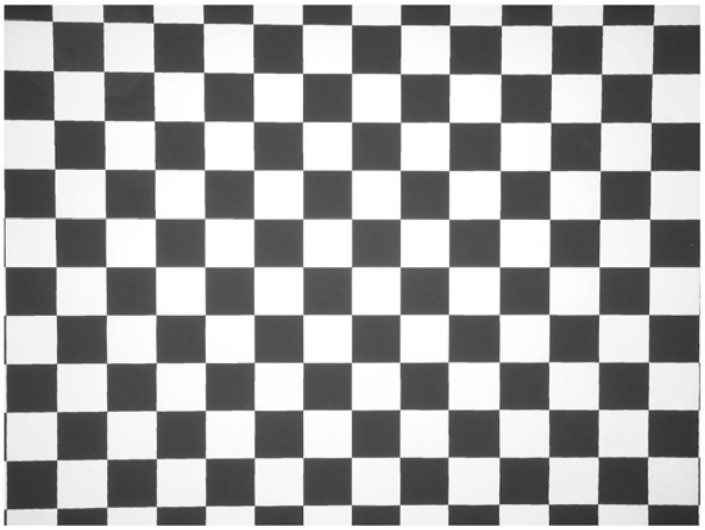

Figure 16: Overcorrected image due to over-modeling over inaccurate data points.

Conclusion

Lens design is inherently a process of compromise between not only the specifications of the lens but also different aberrations. In many cases it appears that high distortion is tolerated when low cost or long zoom ranges are desired. The methods described here shows, fortunately, that distortion is very easy to measure and correct digitally. Even with a rudimentary setup we have achieved perceivable correction. The correction process could be incorporated directly into the firmware of

a digital camera. This would allow more freedom in designing the lens but still an excellent image quality.

Suggestions for next year

Although our original goal was to measure geometric distortion for various lens considering chromatic aberration, time did not permit us to do so. Instead we focused on efficient and simple techniques to measure and correct for this distortion. It would be interesting to measure chromatic aberration, especially close to the edge of the lens where the spread is the largest. It would need more accurate camera positioning tripod.

Another idea would be to correct for the vignetting in general images.

References

[1] McLean, G., ***Image warping for calibration and removal of lens distortion,*** **IEEE Pacific Rim Conference onCommunications, Computers und Signal Processing, Vol.1, pp. 170 -173, 1993**

[2] D. Eadie, F. Shevlin & A.P. Nisbet, ***Geometric Correction of Image Distortion using FPGAs,*** **Proceedings of SPIE Conference on Optical Metrology, Imaging and Machine Vision, September 2002, Galway, Ireland, Volume 4877, pg 28-37,ISBN 0-8194-4658-0.**

[3] L. Lucchese and S.K. Mitra, ***Correction of geometric lens distortion through image warping,*** **Proc. 3rd International Symposium on Image and Signal Processing and Analysis (ISPA'03), Rome, Italy, September 2003**

[4] D. Zorin and A. Barr. ***Correction of Geometric Perceptual Distortion in Pictures.*** **Computer Graphics, 257--264, 1995**

[5] H.A.Beyer ***Accurate Calibration of CCD Cameras,*** **Proceedings of IEEE Computer Vision and Pattern Recognition Conference, CVPR-92, pp. 96-- 101, 1992**

[6] J. Weng, P. Cohen, and M. Herniou. ***Camera calibration with distortion models and accuracy evaluation*****. IEEE Transactions on Pattern Analysis and Machine Intelligence, 14(10):965--980, Oct. 1992. 233**

[7] G. E. Karras, G. Mountrakis, P. Patias, E. Petsa, (1998). ***Modeling Distortion of super-wide-angle lenses for Architectural and Archaeological applications.*** **International Archives of Photogrammetry & Remote Sensing, 32(5), pp. 570-573, Hakodate, Japan**

[8] ***http://www.imatest.com/docs/tour_distortion.html***

Appendix I - Matlab scripts

The test images and source codes are available at
www.stanford.edu/~ddpatil/ee362project/sourceCodeAndImages.tar.gz

Appendix II - Distribution of work

Though it is hard to actually segregate the efforts, since both of us were involved in it at mostly all aspect, the work was mainly divided into two parts. One was image processing and other was image gathering (or data collection). Abhishek did most of the work of collecting cameras, accurately setting them up and taking images. We both put us a small lab in our apartment where we stuck all the posters images on the walls. Dinesh did most of the work of image processing.