Vision, Modeling, and Visualization (2010)

CALTag: High Precision Fiducial Markers for Camera Calibration

B. Atcheson[1] and F. Heide[2] and W. Heidrich[1]

[1]University of British Columbia, Canada
[2]University of Siegen, Germany

Abstract

We present a self-identifying marker pattern for camera calibration, together with the associated detection algorithm. The pattern is designed to support high-precision, fully-automatic localization of calibration points, as well as identification of individual markers in the presence of significant occlusions, uneven illumination, and observations under extremely acute angles. The detection algorithm is efficient and free of parameters. After calibration we obtain reprojection errors significantly lower than with state-of-the art self-identifying reference patterns.

1. Introduction

The typical process for calibrating cameras involves photographing a calibration target from multiple viewpoints, and then identifying calibration points in the image that correspond to known points on the target. One of the most frequently-used targets is a black and white planar checkerboard, where the calibration points are the corner points between squares. This pattern is simple to produce and allows for high accuracy because the corner points can be detected to subpixel precision [Bou08].

The problem in using checkerboards for camera calibration applications lies in how each corner point is detected and identified. The left and center of Figure 1 show common failure cases for automatic checker detection: partial visibility due to clipping against the image boundary, and due to occlusion. It would be useful if we could just place a scan target directly on top of a calibration pattern for stereo acquisition with a handheld camera. This is not possible with checkers due to occlusion and shadows. Instead, the checkers would have to be geometrically well separated from the scan object, thus reducing both the calibration accuracy and the useful image resolution for the actual target object. Manual intervention and labeling can overcome this limitation to some degree, but is cumbersome for multi-camera arrays, videos or large image sequences.

An alternative to the common checker board are individually identifiable (fiducial) markers that allow for detection and thus calibration, even if only a small percentage of tags are visible. Unfortunately for our purpose, most fiducial markers are designed with AR-style applications in mind, where it is important to create isolated markers at a low spatial density. As we will see later, this design compromises the precision of the marker localization. In our work, we focus on the development of a fiducial marker system, which we dub CALTag ("CALibration Tags") that provides

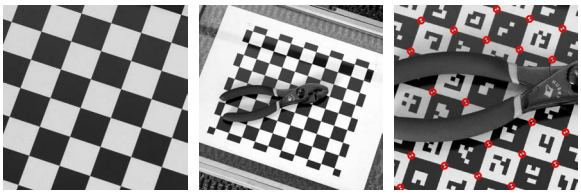

Figure 1: *Partial visibility due to clipping (left image) or occlusion (center) are common failure points of calibration methods involving a checker pattern. By comparison, a calibration system using fiducial markers such as ours (right) can easily deal with partial visibility.*

- accurate localization of calibration points using subpixel saddle point finders,
- high area density of both calibration points and markers,
- robustness under occlusion, uneven illumination, radial distortion and observation under acute angles,
- minimization of false positives and false negatives through use of checksums, and
- automatic processing without parameter tweaking for convenient handling of videos and large image sequences.

DOI: 10.2312/PE/VMV/VMV10/041-048

As a result, our method also supports fully automatic calibration of complex multi-camera configurations where it is difficult or impossible to obtain "nice" views in which each camera sees the entire calibration pattern.

We are presenting two slightly different pattern layouts, using the same fiducial markers as building blocks, which can be detected with the same algorithm. The first layout exhibits a somewhat higher marker density, while the second can be detected more robustly at steep viewing angles. Although our discussion in this paper focuses on dense, planar calibration grids, our method extends naturally to non-planar configurations. The use of individual markers in AR-style settings is possible through a separation of the marker identification and the point localization method (Section 3).

2. Related work

Checker boards. As mentioned above, checker boards are among the most commonly used calibration patterns. For example, the popular OpenCV library [Ope10] contains functionality to automatically locate plain checkerboards. Since the corners of the squares in a checker board are touching, a saddle point finder can be used to find the sub-pixel location of the calibration points with high accuracy and robustness. As mentioned above, the downside of checkers is that it is next to impossible to automatically identify which calibration point is which, unless the full pattern is visible.

The basic checker pattern can be augmented with additional markers to identify additional information. For example, Yu and Peng [YP06] add five double-triangles to the corners and center of a checkerboard and locate those markers using correlation. This works only when the entire board is visible in the field of view, and the orientation cannot be uniquely determined. One of our two proposed calibration patterns is also an extension of a checker board, albeit with fiducial tags inside each field.

Fiducial markers. Fiducial (i.e. individually identifiable) markers have become increasingly popular in recent years. Such markers can be used in a variety of settings. Individual large-area markers are used as 2D barcodes to encode data beyond a simple identifier (e.g. [ISO06a,ISO06b]). More interesting for camera calibration are smaller fiducial markers that only encode a unique code for identification purposes. Even in this category, there are a large number of markers documented in the literature.

Some of the most common fiducial marker designs include concentric rings, where the center is the calibration point, and the ring pattern identifies the marker (e.g. [GGSC96, CN01, SBGD07]), central dots demarking the calibration point, combined with radially arranged code patterns (e.g. [LMH02, NF02]), and finally rectangular patterns with identification codes in the interior (e.g. [ZFN02, OXM02, Fia05, FS07]). An interesting property of the rectangular design is that every marker encodes four calibration points, i.e. corners, rather just than one. These points have been localized by fitting lines to the edges of the rectangle and computing the intersection points. While this approach provides better accuracy than the center-of-mass-style calculations used in many circular designs, we show that it falls short of the precision provided by saddle point finders employed in checker patterns and in our design.

Another shortcoming of many existing fiducial markers is that they require a lot of empty (white) space between them, and can thus not be packed tightly on a calibration pattern. This is particularly true for the circular designs. However, a high density of calibration points is very desirable for camera calibration: first, a large number of point correspondences improves the fitting results for homographies and other camera models, and second, many small markers make detection more robust under occlusion and high frequency illumination than few large markers.

Our CALTag design is based on rectangular encodings, but they can be packed tightly so as to allow for both a high marker density and the use of high precision saddle point finders. Like some other recent designs (e.g. [FS07]), our marker IDs allow for error *detection*. They do not, however, provide error *correction*, since we anticipate CALTags will be used in larger groups, so that not identifying a subset of the markers is no problem as long as the corresponding calibration points can still be localized.

3. CALTag design

The CALTag design involves two major components, the marker design (Section 3.1), and the detection algorithm (Section 3.2). For use in a calibration grid, we propose two possible geometric layouts for the same basic markers. As we will see in Section 3.3, the first, straightforward layout provides a slightly higher density of markers and calibration points, while the second layout facilitates both more robust and more efficient detection.

3.1. Marker design and layout

Marker design. For robustness under different lighting conditions and easy printing, we choose a binary marker design. Each CALTag marker consists of an $M \times N$ matrix of black and white squares (“pixels”), surrounded by a K pixel boundary that is either solid white or solid black. While we have conducted experiments with other configurations, we restrict ourselves to configurations with $M = N = 4$ and $K = 2$ for this paper (see Figure 2). The choice of code resolution is a tradeoff between the size of the codebook and the physical size of the pattern. As described shortly, not every possible code can be used, so a small pattern limits the number of available markers and hence the number of corner points in a calibration grid. On the other hand, for the same physical marker area, smaller code patterns afford a larger printed pixel size δ.

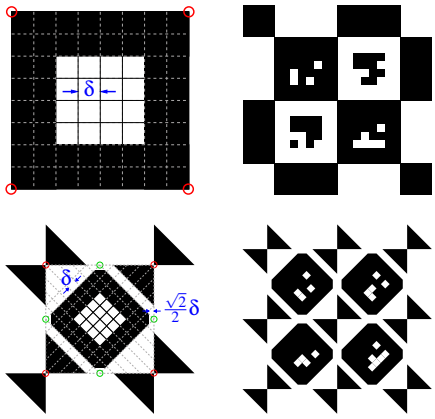

Figure 2: *Top left: basic CALTag marker. Top right: checker-style layout in which touching squares provide the calibration points. Bottom left: rotated CALTag markers with additional bowtie symbols providing calibration points. Bottom right: grid layout using the rotated markers.*

Of the total 16 bits, we use the first $p = 10$ bits to represent the identifier, and the remaining $MN - p = 6$ bits for a checksum (CRC-6-ITU). The binary string is then rearranged into a 2D matrix for form the code. This allows for 2^p potential codes with a minimum Hamming distance of 3, meaning that all possible one- or two-bit flips can be detected *in a 1D code vector*. However, not all of these codes can be used, for two reasons. The first is that, in order to avoid inter-marker confusion under bit flips *in our 2D grid arrangement*, we must ensure that all rotated versions of marker codes have a minimum Hamming distance of 2 from all other used marker codes. The second reason is that patterns that are mostly white or mostly black are more likely to occur as textures or random patterns in normal images. For this reason, we choose only those codes with between 25% and 75% of their total pixels "on". This second criterion eliminates a relatively small percentage of codes in which both the data portion and the CRC portion has a very one-sided intensity distribution.

We used a greedy search algorithm to find a set of valid codes. The net effect of the two constraints was that, out of 1024 codes for our 4×4 grid layout, 30 codes were rejected due to the bit count constraint, and 302 codes due to the symmetry constraint. In total, 692 codes remain to be used as valid calibration patterns. Enforcing a minimum Hamming distance of 3 under all rotations would reduce the number of codes to 280. When assembling a calibration pattern, we use all valid codes in numerical order, without further attempts to maximize Hamming distance. Due to the minimum Hamming distance of 2, these codes allow for the detection of *any single* bit flip under *any rotation*. However, the CRC codes are more powerful than that. In 1D, they can also detect any "burst" errors (flips of subsequent bits) with burst lengths of up to 6 bits. Although our 2D layout reduces the usefulness of this property, there are situations where this feature of CRC codes is helpful. For example, if a whole row of code pixels is occluded, the resulting pattern change can be detected. In all our experiments with CALTag patterns, we have never observed a false positive marker identification of random scene structure.

Pattern layouts. Once the markers have been defined, they need to be arranged into a calibration pattern. As outlined in Section 2, we desire a dense packing of the markers to maximize the number of markers and calibration points per unit area. Also, we would like to derive a layout in which the calibration points are given by local "bowtie" image topologies, in which black and white image portions touch like the corners in a checker board. With this kind of layout, calibration points can be localized with very high accuracy using a saddle point finder (also see Section 4).

A straightforward layout that achieves these goals is to pack markers with a black border and markers with a white border like the squares in a checker board (Figure 2, top right). This first layout optimizes marker density, but its detection may suffer from merging of different marker regions under difficult photometric conditions. These issues are discussed in detail in Section 3.3.

Our second layout overcomes these merging problems by spatially separating the markers from each other. The calibration points in this layout are provided by additional bowtie shapes, as shown in Figure 2, bottom right. Note that, in this second layout, the corners of the markers are boundary are clipped slightly (Figure 2, right). This does not affect the detection algorithm, which we discuss next.

3.2. Detection algorithm

The stages of the detection algorithm are depicted in Figure 3. Beginning with the recorded image, we first find the potential markers using simple image processing techniques and some carefully chosen filtering criteria. The true markers are then confirmed by reading their binary codes. Finally, any missed calibration points are located using prior knowledge of the checkerboard layout. The output is a set of ordered 2D image coordinates corresponding to the calibration points.

3.2.1. Connected components

This first stage of the algorithm is the only one that differs slightly between the two pattern layouts. For the first layout, the input image is converted to grayscale, and its edges are detected using a Sobel filter. After thresholding, but before thinning the filter response, we clean the data by inverting any zero pixels that have two nonzero unconnected neighbors and then apply a 3×3 median filter. Next, a binary thinning operation is applied, after which we remove any isolated pixels that remain. Finally, we invert the edge

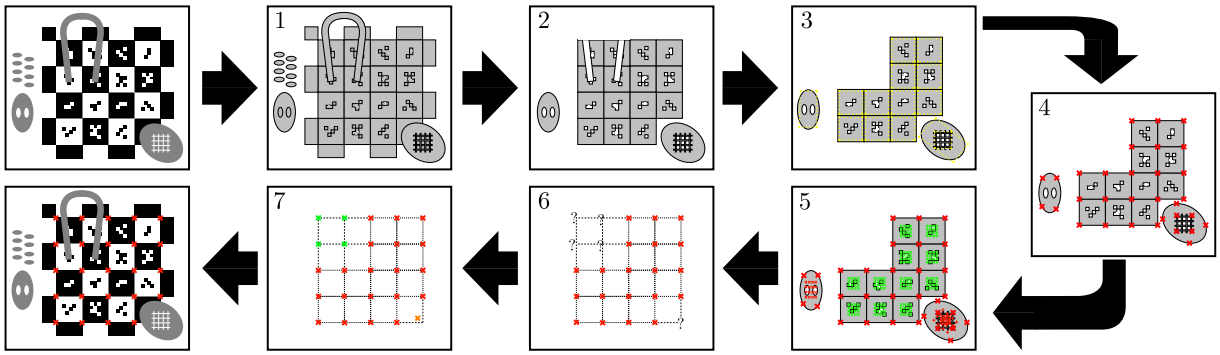

Figure 3: *Flowchart of the detection process. Numbered blocks correspond to the subsections 3.2.1 through 3.2.7 below.*

image and extract the connected components. We experimented with many different variations of this image processing pipeline in search of one that would work well across a variety of resolutions and image quality settings. Gaps in marker edges should be closed to prevent merging of markers (see Section 3.3) but at the same time it is important to not connect the marker edges to the edge pixels of the code dots. Since the marker shapes are not touching in the second case, we can skip the edge detection and morphological filtering steps and directly compute the connected components after adaptive thresholding of the grayscale image.

3.2.2. Identification of potential markers

The previous stage outputs more connected components than there are markers in the image; random background objects, as well as small segments of highly textured regions all result in components. The following two criteria are used to reject components that cannot possibly be markers:

(a) Area. We assume that each code pixel must cover an area of at least 2×2 image pixels in order to be reliably resolvable. By design our markers are 8×8 units, so each one must cover at least 16^2 pixels. This lower bound often helps to remove thousands of tiny regions that can occur in highly textured regions, such as grass or carpet. For an upper bound we use $1/8^{\text{th}}$ of the input image size, since having fewer than 8 points would typically be insufficient for calibration.

(b) Euler number. The Euler number of an image is defined as the total number of objects in the image, minus the number of holes in those objects. Computing the Euler number for an individual connected component gives us a measure of how many interior holes there are. This calculation can be performed very efficiently [Gra71]. The maximum possible number of holes would arise in the case of a marker with alternating black and white code dots, so we use a threshold of $-(MN/2)$, although in practice most markers have between 1 and 3 holes. Nested holes do not pose a problem – the entire internal code region would be considered as a separate marker, fully enclosed by the surrounding checkerboard square, and then rejected due to it having either too small an area, or an invalid binary code. The advantage of filtering based on Euler numbers is that they are resolution independent and require no parameter tweaking.

Approximate convexity was also investigated as a filtering criterion (markers are often not truly convex, due to image noise, edge detection errors and aliasing), but we found it to be expensive to compute and unnecessary given the success of the above two criteria.

3.2.3. Quadrilateral fitting

We next attempt to fit quadrilaterals to the remaining components. While the checkerboard as a whole may be distorted, the individual squares should be small enough that their boundaries can be well approximated by four linear segments. As Figure 8 shows, patterns in images with high radial distortion can still be detected.

The first step is to trace the outline of the region, in any direction, to obtain image coordinates for the region's edge pixels. For each sample point on this boundary we compute the approximate gradient using central differences and then smooth these gradients. The smoothing kernel size is set based on the size of the component so as to remove spurs and holes in the boundary. These gradients are fed into Lloyd's K-Means clustering algorithm [Llo82], with $K = 4$, to obtain the four dominant edge orientations. A least-squares line fit through each of these clusters is then used as the initial guess in finding the four boundary lines, again via Lloyd's algorithm. At this point we have the four best fitting boundary lines (regardless of what shape the region is and how many edges it actually has) without any ordering. To extract a quad we therefore find the two most parallel lines, taking these to be opposite edges. This is sufficient to obtain a cyclic ordering of the corner points, which are themselves obtained via intersections with the other pair of lines.

Note that the quadrilateral fitting is not affected by the fact that the markers are technically octagons in the second pattern layout, since the four additional edges are less than one quarter of the length of the long edges, and are therefore dominated by those in the clustering steps.

3.2.4. Saddle points

We now find subpixel-accurate saddle points in the greyscale image I using the same algorithm as that used by OpenCV [Ope10]. It considers all points p within a small window around an approximate saddle point x. Nonzero image gradients only occur along edges, where they are orthogonal to the edge itself. Hence, if x is a saddle point, $\nabla I(p) \cdot (p - x) = 0$ for all p near x. This leads to a system of linear equations that can be iteratively solved for successively more accurate saddle point positions. The initial guess is given by the quadrilateral fit, as well as the known location of calibration points with respect to the markers in the two layouts (Figure 2). For the checker-style layout, initial guesses are provided by the intersections of the four fitted corner lines. For the rotated layout, the initial guesses are given as the columns of $H \cdot \begin{pmatrix} -0.5 & 0.5 & 1.5 & 0.5 \\ 0.5 & 1.5 & 0.5 & -0.5 \end{pmatrix}$, where H is the homography between the marker square and the detected quadrilateral (also see the next step in the pipeline).

There are two difficulties with applying the saddle finder to every corner point: first, it can have an impact on performance if there are many points, and second, the guesses arising from line intersections can be so poor that the corner cannot be found. But due to the layout of the markers, we know that each corner point should have up to four guesses corresponding to it, from each of the detected adjacent markers. We therefore cluster together nearby guessed corner points and consider only their average. Doing so provides us with an improved initial guess, and eliminates the redundancy of searching for saddle points multiple times in the same image region. We use half of the average side length of the associated marker as a Euclidean distance threshold for grouping nearby points.

3.2.5. Marker validation

At this point we have a collection of regions, most likely (although not guaranteed to be) quadrilaterals, along with four corner points for each region. Our task is to read the binary code depicted in the middle of the marker. Given a uniform square, the positions c_i of the code dots inside this square are known by construction of the markers. We must therefore map a unit square to the region's corners and then sample the image at the points dictated by applying the same mapping to the c_i.

The corner points are ordered cyclically, clockwise around their centroid, but we do not yet know which point corresponds (arbitrarily) to the top left corner of the marker. All four possible orientations must therefore be considered in searching for a valid code (in this work we ignore mirror reflections, but they could easily be accommodated by testing the other four permutations too). A 2D homography H from the unit square to corner points is generated, giving us the sampling points for the code pixels. Rather than sampling the grayscale image directly, we first apply adaptive thresholding to it. In this case, the radius of the Gaussian smoothing kernel is chosen to be three times the width of the marker. The filtering neighborhood therefore should contain enough black and white parts of the pattern that a local average can be reasonably estimated. The thresholded image is now sampled at the supposed code dot points, and converted in columnwise order to a string of binary characters.

The binary code is validated by computing the checksum of the first p bits and comparing it to the sampled checksum under all four possible rotations. Had an error-correcting coding scheme been employed, we could also correct for small errors in sampling the pattern, or for partially occluded patterns, but we found the 16 bit combination to work well enough in practice that most of the markers are detected correctly. False negatives do not pose a problem, since the checkerboard can be detected even when only a few (or potentially just one) of the markers are correctly detected.

The markers with valid checksums are now filtered to remove any that have markedly different orientations to the others, where the orientation is taken to be the angle that the vector from the top left to the top right corner makes with the horizontal axis.

3.2.6. Locating missed points

In this stage we attempt to find any calibration points that are visible in the input image, but that were missed during detection, for example because the surrounding markers could not be identified. If at least one marker is correctly identified, then because we know where it lies in the checkerboard pattern from its ID, we can guess where the remaining saddle points should lie in the image. As before, we fit a homography to the detected points using RANSAC, and from that obtain the approximate image coordinates of the missing points. At these points we run the saddle finder, and if it converges we add that point to the collection of calibration points.

Due to lens distortion, a homography may not adequately describe the positions of the image points. We must therefore estimate the amount of distortion, undistort the points before fitting the homography, find the missing points, and then redistort them before looking for saddles. We chose to model only the leading term of radial distortion since it dominates, and because higher order terms are unlikely to be reliably estimated with only the data available in a single image. The distortion coefficient is estimated via nonlinear optimization, where as an error metric we measure the collinearity i.e. the squared sum of orthogonal distances from each image-space point in each row and column of the checker grid to a straight line fit through those points [WM94].

3.2.7. Saddle validation

Convergence of the saddle finder is no guarantee that a saddle is actually present, since occlusions or specular highlights can result in false positives. We must therefore validate the potential points identified in the previous stage.

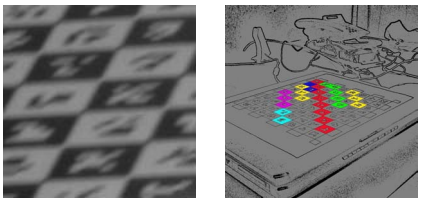

Figure 4: *Left: motion blur causes diagonally adjacent markers to merge together. Right: connected components consisting of more than one marker.*

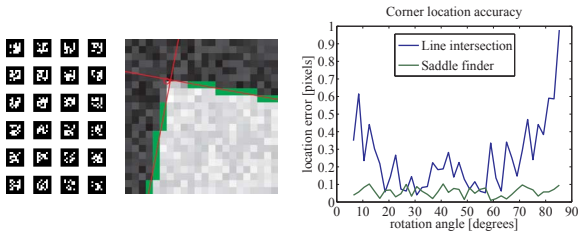

Figure 5: *Left: ARTag calibration grid. Center: corner localization test with synthetic data, containing a white square on black background with added noise. The true corner point is indicated by the red circle. Green pixels indicate the output of the edge detector and are used to fit straight lines. For nearly vertical or horizontal lines, aliasing causes very poor line fits, resulting in a large error. Right: precision experiments comparing a saddle finder [Bou08] using similar image data with a bowtie shape to ARTag's line intersection method for various rotation angles. The results demonstrate that the saddle finder gives superior results while also being insensitive to rotation angle.*

We perform two tests, both of which must be passed for a point to be considered a saddle. First, the local neighborhood of pixel intensities must conform to the distribution of intensities around the other, known, saddles. A beta distribution describes these distributions well, with parameters $0 < \alpha \approx \beta < 1$. The skewness is dependent upon the relative orientation of the grid to the camera. Image blur produces flatter distributions. We fit such a distribution to the known saddles and then reject any potential points where the parameters differ by more than some threshold from the median.

The second test counts edges. We sample a few evenly spaced points, ordered on a small circle around the point, in the adaptively thresholded image. The radius of the circle can be chosen based on the homography so as not to intersect any code dots. After smoothing the sampled signal we check to see if it alternates between black and white exactly four times.

3.3. Merged marker problem

The main cause of failures in detecting markers in the first layout is when they appear to merge together, either at steep angles, or when motion blur is present. Figure 4 illustrates the problem, also noted by Fiala and Shu [FS07]. Instead of having one connected component per grid square, we see groups of diagonally connected squares which prevent marker detection.

To solve this problem reliably, we designed the second, rotated layout. However, for completeness sake we would like to describe some of the algorithmic solutions that seemed plausible and were tried, but ultimately proved too unwieldy or not robust enough. The key difficulty of course is to find a sequence of operations and parameters that works across a wide range of image types and illumination conditions. Changes designed to fix one particular problem can easily create more problems elsewhere.

Morphological openings can be applied to break the connection between diagonally adjacent squares. The size of the structuring element is resolution-dependent and must be set very carefully – too small and it will not break the bridges, too large and it will break open the markers where the code dots are too close to the edge. To complicate matters further, the appropriate size can also be spatially varying due to perspective distortion. Linear structuring elements, orthogonal to the bridge direction could be employed to break the bridges, but this requires higher level knowledge of the checkerboard orientation, which is not available in the early stages. We also attempted to locate the bridge points using the method described by Sun et. al. [SYXH08] where interior corners are found by sampling the image on a circle around a particular point. These ordered samples are converted into a 1D code that should alternate four times between white and black. Again, this approach works in our final saddle validation step because we know the size of the markers, but without the homography we cannot choose an appropriate radius for all input images. Linking edges along the region contour could also potentially help to locate the bridges, but we found this to be too sensitive to the thresholds used to break up line segments.

4. Analysis and results

As a primary point of reference for our approach we use the ARTag markers [Fia05, FS07], since they represent a state-of-the-art fiducial marker system and can (and have) been applied for camera calibration (e.g. [BBH08, BPS*08]). Like our approach, ARTags consist of a binary dot-matrix, laid out in a 2D rectangular grid (Figure 5). Various image processing techniques are used to locate potential markers, and then sample the interior code points to obtain a binary sequence. The sequence comprises 36 bits, 10 of which encode the marker ID while the remainder are dedicated to a CRC and a Reed-Solomon error correction code.

Although ARTags can be detected and identified reliably,

they are not ideal for camera calibration, primarily because the corner localization is comparatively poor (Figure 5). Each ARTag marker is reported along with the positions of the quadrilateral corners. These are found by detecting edges in the image, linking them to make up quadrilaterals, fitting lines through adjacent edges and computing their intersections. Localization of the corner is thus dependent on a line fit through pixels far away from the actual point. Since this takes place before calibration, the image edge may not be straight due to lens distortion. In addition, edges cannot be perfectly detected and localized, and so the choice of filter kernel used in edge detection could compromise the accuracy of the point localization. Our method detects saddle points instead.

Figure 6 (left) shows that CALTag detection performance is roughly linear in the input image size. Our implementation could certainly be optimized further (MATLAB source code in included with our submission. We plan to post it online). In comparing the density of markers between ARTag and CALTag, we normalize the pattern scales so as to use the same pixel size δ for the codes, since it is that pixel size which determines if code bits can be read. ARTag uses 36 bit error-corrected codes, while CALTag uses 16 bit error detection codes. While the larger code size and error correction ability are useful in AR applications, they do not provide additional advantages for camera calibration and instead consume space that could be used for more markers. The requirement to separate the ARTags markers by whitespace further reduces the density of ARTag markers. However, each ARTag marker provides four calibration points, whereas the calibration points are shared between adjacent markers in the CALTag system. Figure 6 (top right) shows that for our first, checker-like grid layout, the net effect is a point density that is similar to that of ARTag. The horizontal axis represents the side length of the entire square pattern, with a code pixel being one unit long. The vertical axis shows the total number of corner points, taking into account all the necessary padding of each pattern. The point density of the second layout is about half that of the first.

We performed calibrations using both ARTag and CALTag patterns, scaled and cropped to have the same printed code pixel size and, as closely as possible, the same printed physical area. For ARTag this meant we had 5×6 markers (120 calibration points) whereas the first CALTag layout had 8×9 (90 calibration points). Using a 10 megapixel SLR camera with a 20mm lens, we captured sixteen images, two of which are shown in Figure 7. CALTag was able to automatically detect 1438 of the 1440 points across all images. ARTag detected 1912 out of 1920 points. While ARTag's execution time for single image is much faster than our CALTag implementation, the difference was negated by the need to manually mask out the background in the images (almost 10 minutes). Without masking, ARTag ran out of memory, most likely due to the large number of connected components in the highly textured carpet region. Using our own implementation of Zhang's calibration algorithm [Zha00], we obtained a mean reprojection error of 1.073 pixels for the ARTag points, versus 0.316 pixels for the CALTag points.

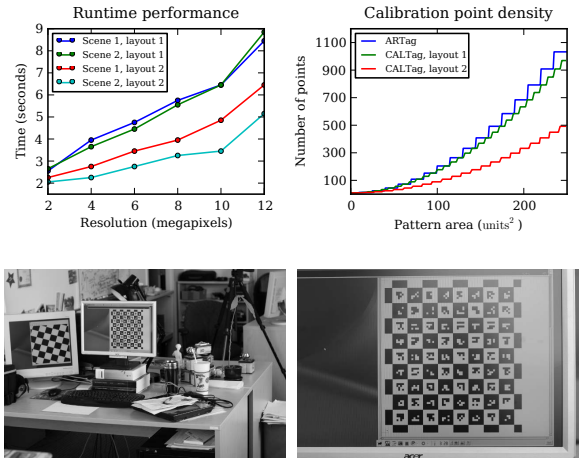

Figure 6: *Top left: performance is roughly linear in the image size. Timings are for our Matlab(R) implementation on an Intel(R) Core(TM)2 6400 CPU. Scene 1 is shown below left, scene 2 is below right. Top right: calibration point density for the various patterns.*

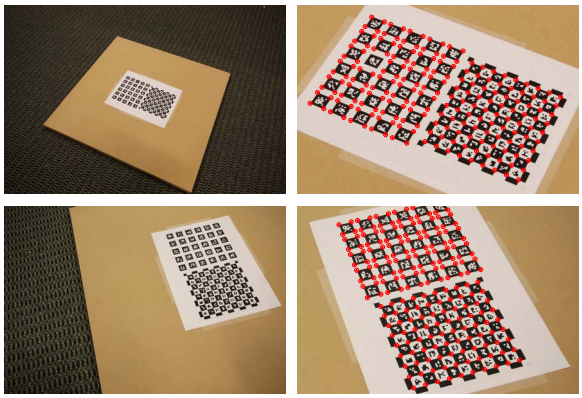

Figure 7: *Two of the sixteen images used for the calibration test. The left column shows the captured images, while the right column shows the detected points overlaid on zoomed regions. Zoom in on the PDF to examine the points more closely.*

Figure 8 shows several more results for just the CALTag detection, including calibration grids that occupy a small percentage of the image area, radial lens distortion, and extremely acute observation angles. The pattern detection is successful and robust even under extremely difficult conditions.

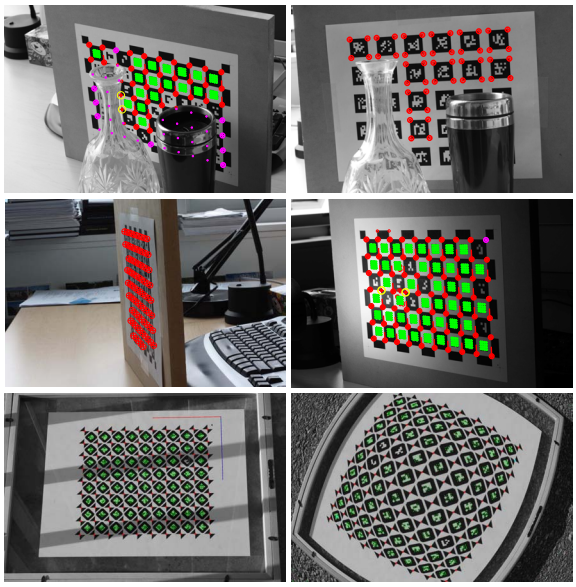

Figure 8: *More results, under difficult conditions. Zoom into the images for more detail. Red circles show the calibration points. Green crosses are the sampled code locations. Magenta crosses are guessed saddle locations. Magenta circles are guessed locations deemed to be valid points. Yellow circles shows points excluded due to failing the validation tests (dirt and scratches on the pattern cause some true saddles to be excluded). Top row shows occlusion, where CALTag (left) is able to detect points where the marker is partially occluded. ARTag (right) misses those corners. Middle row shows steep angles and strong lighting variation. Bottom row shows harsh shadows and radial distortion on the second layout.*

5. Conclusion

We have presented a new fiducial marker pattern targeted at camera calibration along with an efficient and robust method for detecting it. CALTag affords two main benefits. First, the accuracy of calibration point localization via saddle points is demonstrably superior to the line fits through quad edges used in previous work. This results in much lower reprojection errors for camera calibration. The second, and perhaps more important benefit, is its ease of use, particularly in multi-camera configurations. Calibration images can easily be captured without having to carefully position the grid in the field of view of each camera, and without having to manually identify the points in all the images. In addition, no paramaters need be set by the user to handle different scene types or resolutions.

References

[BBH08] BRADLEY D., BOUBEKEUR T., HEIDRICH W.: Accurate multi-view reconstruction using robust binocular stereo and surface meshing. In *Proc. CVPR* (2008).

[Bou08] BOUGUET J.-Y.: Camera calibration toolbox for Matlab, 2008. http://www.vision.caltech.edu/bouguetj/calib_doc/.

[BPS*08] BRADLEY D., POPA T., SHEFFER S., HEIDRICH W., BOUBEKEUR T.: Markerless garment capture. *ACM Trans. Graph.* (2008).

[CN01] CHO Y., NEUMANN U.: Multi-ring color fiducial systems for scalable fiducial tracking augmented reality. *Presence: Teleoperators and Virtual Environments 10*, 6 (2001), 599–612.

[Fia05] FIALA M.: ARTag, a fiducial marker system using digital techniques. In *CVPR* (2005), vol. 2, IEEE, pp. 590–596.

[FS07] FIALA M., SHU C.: Self-identifying patterns for plane-based camera calibration. *Machine Vision and Applications 19*, 4 (July 2007), 209–216.

[GGSC96] GORTLER S. J., GRZESZCZUK R., SZELISKI R., COHEN M. F.: The lumigraph. In *Proc. Siggraph '96* (1996), pp. 43–54.

[Gra71] GRAY S. B.: Local properties of binary images in two dimensions. *IEEE Trans. Computers 20*, 5 (1971), 551–561.

[ISO06a] ISO/IEC 16022:2006: Information technology – automatic identification and data capture techniques âĂŤ data matrix bar code symbology specification, 2006.

[ISO06b] ISO/IEC 18004:2006: Information technology – automatic identification and data capture techniques – QR code 2005 bar code symbology specification, 2006.

[Llo82] LLOYD S. P.: Least squares quantization in PCM. *IEEE Trans. Inf. Theory 28*, 2 (1982), 129–137.

[LMH02] LÓPEZ DE IPIÑA D., MENDONÇA P., HOPPER A.: Trip: A low-cost vision-based location system for ubiquitous computing. *Personal Ubiquitous Computing 6*, 3 (2002), 206–219.

[NF02] NAIMARK L., FOXLIN E.: Circular data matrix fiducial system and robust image processing for a wearable vision-inertial self-tracker. In *Proc. ISMAR* (2002).

[Ope10] OPENCV: *OpenCV 2.0 C++ Reference*. http://opencv.willowgarage.com/documentation, 2010.

[OXM02] OWEN C., XIAO F., MIDDLIN P.: What is the best fiducial? In *Proc. IEEE Workshop on Augmeted Reality Toolkit* (2002), pp. 98–105.

[SBGD07] SATTAR J., BOURQUE E., GIGUÈRE P., DUDEK G.: Fourier tags: Smoothly degradable fiducial markers for use in human-robot interaction. In *Proc. Computer and Robot Vision* (2007).

[SYXH08] SUN W., YANG X., XIAO S., HU W.: Robust checkerboard recognition for efficient nonplanar geometry registration in projector-camera systems. In *Proc. PROCAMS* (2008).

[WM94] WEI G.-Q., MA S. D.: Implicit and explicit camera calibration: Theory and experiments. In *IEEE Trans. Pattern Analysis and Machine Intelligence* (May 1994), vol. 16, pp. 469–480.

[YP06] YU C., PENG Q.: Robust recognition of checkerboard pattern for camera calibration. *Optical Engineering 45*, 9 (September 2006), 093201–9.

[ZFN02] ZHANG X., FONZ S., NAVAB N.: Visual marker detection and decoding in AR systems: A comparative study. In *Proc. ISMAR* (2002).

[Zha00] ZHANG Z.: A flexible new technique for camera calibration. *IEEE Transactions on Pattern Analysis and Machine Intelligence 22*, 11 (2000), 1330âĂŞ–1334.