AprilTag 2: Efficient and robust fiducial detection

John Wang and Edwin Olson

***Abstract*— AprilTags and other passive fiducial markers require specialized algorithms to detect markers among other features in a natural scene. The vision processing steps generally dominate the computation time of a tag detection pipeline, so even small improvements in marker detection can translate to a faster tag detection system. We incorporated lessons learned from implementing and supporting the AprilTag system into this improved system.**

This work describes AprilTag 2, a completely redesigned tag detector that improves robustness and efficiency compared to the original AprilTag system. The tag coding scheme is unchanged, retaining the same robustness to false positives inherent to the coding system. The new detector improves performance with higher detection rates, fewer false positives, and lower computational time. Improved performance on small images allows the use of decimated input images, resulting in dramatic gains in detection speed.

I. Introduction

Fiducials are artificial visual features designed for automatic detection, and often carry a unique payload to make them distinguishable from each other. Although these types of fiducials were first developed and popularized by augmented reality applications [1], [2], they have since been widely adopted by the robotics community. Their uses range from ground truthing to object detection and tracking, where they can be used as a simplifying assumption in lieu of more sophisticated perception.

A few key properties of fiducials make them useful for pose estimation or object tracking in robotics applications (Figure 1). Their uniqueness and high detection rate are ideal for testing SLAM systems. Fixed fiducial markers can be used for visual localization or as a ground truth estimate of robot motion. Fiducials mounted on objects can be used to identify and localize objects of interest.

This work is based on the earlier AprilTag system [3]. The design of AprilTags as a black-and-white square tag with an encoded binary payload is based on the earlier ARTag [2] and ARToolkit [1]. AprilTag introduced an improved method of generating binary payloads, guaranteeing a minimum Hamming distance between tags under all possible rotations, making them more robust than earlier designs. The tag generation process, a lexicode-based process with minimum complexity heuristics, was empirically shown to reduce the false positive rate compared to ARTag designs of similar bit length.

Based on feedback from AprilTag users in the robotics community, we determined that most users do not accept tags with decode errors. In these cases, features such as support for recovering partially-occluded tag borders are seldom useful. This functionality must be weighed against the costs of additional computation time and an increased false positive rate.

The authors are with the Computer Science and Engineering Department at the University of Michigan in Ann Arbor, MI, USA. {jnwang,ebolson}@umich.edu; http://april.eecs.umich.edu

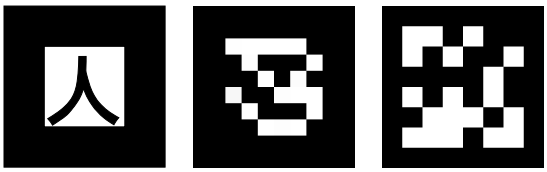
Fig. 1: Applications and users of AprilTags. From top left, clockwise: robot-to-robot localization and identification on MAGIC robots, object localization for Boston Dynamics' Atlas robot, testing of virtual reality headsets at Valve, and tracking individual ants to study their social organization [4].

This work describes a method for improving AprilTag detection speed and sensitivity while trading off the ability to detect partially-occluded tags. We show that this method is faster than the previous detection method, reducing the rate of false positives without sacrificing localization accuracy. The contributions of this paper are:

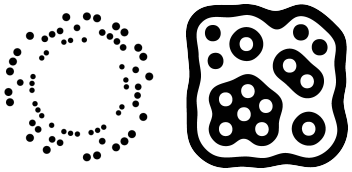
- an AprilTag detection algorithm that improves detection rate for small tags, exhibits fewer false positives, and reduces computation time compared to the previous algorithm
- a new tag boundary segmentation method that is responsible for many of the performance improvements, and could be applied to other fiducial detectors
- an evaluation of the effect of fewer tag candidates on false positive rates
- an experimental characterization of the localization performance of our detector on real and synthetic images

II. Prior Work

One of the earliest visual fiducial systems was introduced by ARToolkit [1], a library for augmented reality applications. ARToolkit introduced the black square tag as a tracking



(a) ARToolkit (b) ARTag (c) AprilTag



(d) RUNE-Tag (e) reacTIVision

Fig. 2: A comparison of visual fiducial tags. ARToolkit tags allow arbitrary pixel patterns inside the black border, while ARTags and AprilTags use 2D binary codes. RUNE-tags and reacTIVision markers are two different existing approaches to fiducial markers.

marker, which has the advantage of providing a full 6-DOF pose estimate from a single marker of known scale. ARToolkit distinguished tags by embedding arbitrary image patterns inside the square, which were matched against a database of known patterns for identification. As the database of recognized patterns grew, so did the computational cost of matching and the likelihood of confusing distinct patterns. ARTag [2] attempted to rectify the problem of inter-tag confusion by introducing a 2D binary barcode pattern. The binary barcode allowed bit errors in detection to be corrected. An improved detector algorithm used image gradients to detect tag edges, an improvement over the primitive thresholding method of ARToolkit. Surviving forks of the project include ARToolkitPlus [5] and Studierstube Tracker [6].

AprilTag [3] built upon the advances of ARTag, introducing a lexicode-based system for generating tags. AprilTags guarantee a minimum Hamming distance between tags under all possible rotations, while enforcing a minimum complexity constraint to reduce the rate of false positives from arising in natural images. Localization accuracy was improved over ARTag's previous state of the art. Moreover, AprilTag provided a popular open source detector implementation which encouraged its adoption by the academic community.

The original AprilTag detector used image gradients to detect high-contrast edges. This has the advantage of being robust to shadows and variations in lighting over previous methods which used naive thresholding. Detection of partially occluded tags was made possible by first fitting segments to the gradients, then searching over combinations of segments that formed four-sided shapes, or quads. A disadvantage of the segment-first approach is the large number of candidate quads that are generated. Much processing time is spent attempting to decode invalid candidate quads. Empirically, the AprilTag detector spends most of its time fitting lines to gradient edges, many of which will not be part of valid tag detections.

Besides square-shaped binary tags, other tag encoding schemes have been proposed. In particular, reacTIVision [7] uses a unique topological tag recognition system introduced by d-touch [8]. FourierTags [9] are radially symmetric tags designed to increase detection range by degrading smoothly. RUNE-Tags [10] are named after the circular dot patterns (rings of unconnected ellipses) which make up the fiducial marker. The dots are chosen to provide localization accuracy at the expense of computation time, while being robust to blurring, noise, and partial occlusion. Pi-Tag [11] uses cross-ratio to recognize markers, noting that the cross-ratio of four points in a line is invariant under camera projective geometry. ChromaTags [12] are an extension of AprilTags, where two bicolor tags are blended in order to maximize the gradient magnitude in the CIELAB color space. The colorspace conversion reduces the number of edges compared to the grayscale image, thus speeding detection.

III. Tag Detector

Our system features an improved quad detector which finds candidate tags in a grayscale image. Each candidate is then decoded to determine if they are valid AprilTag detections. The method leads to fewer false positives than the previous state of the art detector while reliably detecting valid unoccluded quads, contributing to an overall lower false positive rate.

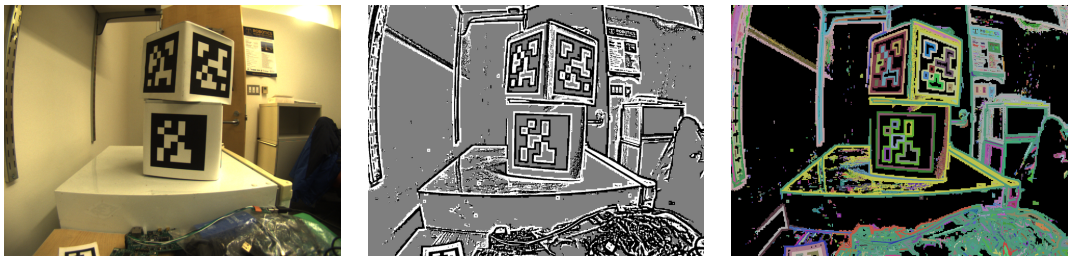
A. Lessons learned

The improvements to the tag detector were inspired by user feedback about common use cases. We learned that in most deployments, detection of partially occluded tags is of limited utility. Occluded tags often have one or more bit errors, and most users disable decoding of tags with bit errors due to the impact on false positive rates. No known users accept tags with more than two bit errors, which enables a faster decode algorithm. In our experience, the increased detection speed is a favorable tradeoff against the ability to recover partially occluded tag borders.

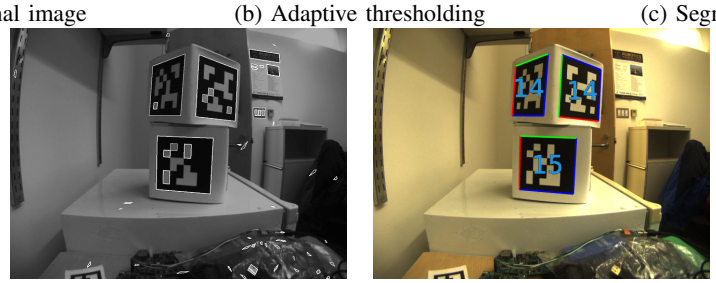
B. Adaptive thresholding

The first step is to threshold the grayscale input image into a black-and-white image. Some thresholding methods attempt to find a global threshold value for the entire image [13], while others find local or adaptive thresholds [14]. We adopt an adaptive thresholding approach, where the idea is to find the minimum and maximum values in a region around each pixel.

Instead of computing the exact extrema (max and min values) around every pixel, we divide the image into tiles of 4x4 pixels and compute the extrema within each tile. To prevent artifacts from arising between tile boundaries with large differences in extreme values, we find the extrema in a neighborhood of 3x3 surrounding tiles, ensuring a minimum of one tile overlap when computing extrema for adjacent pixels. Each pixel is then assigned a value of white or black, using the mean value $(max+min)/2$ as the threshold (Figure 3b). For our application, we only need to consistently



(a) Original image (b) Adaptive thresholding (c) Segmentation



(d) Detected quads (e) Tag detections

Fig. 3: Intermediate steps of the AprilTag detector. The input image (a) is binarized using adaptive thresholding (b). The connected black and white regions are segmented into connected components (c). Component boundaries are segmented using a novel algorithm, which efficiently clusters pixels which border the *same* black and white region. Finally, quads are fit to each cluster of border pixels (d), poor quad fits and undecodable tags are discarded, and valid tag detections are output (e).

differentiate the light and dark pixels which form the tag. Regions of the image with insufficient contrast, colored in gray in Figure 3b, are excluded from future processing to save computation time.

C. Continuous boundary segmentation

Given the binarized image, the next step is to find edges which might form the boundary of a tag. A straightforward approach is to identify edge pixels which have an opposite-colored neighbor, then form connected groups of edge pixels. However, this approach breaks down when the white space between tag boundaries approaches only a single pixel wide, which may happen for physically small or faraway tags. If two tag boundaries are incorrectly merged, the tags will not be detected. Our proposed solution is to segment the edges based on the identities of the black and white components from which they arise.

Connected components of light and dark pixels are segmented using the union-find algorithm [15] (Figure 3c), which gives each component a unique ID. For every pair of adjacent black and white components, we identify the pixels on the boundaries of those two regions as a distinct cluster. This clustering can be done efficiently by using a hash table, indexing each cluster by the black and white components' IDs, as described in Figure 4. In the aforementioned case of a single pixel-wide white component separating two distinct black components, we have solved the problem by allowing the same white pixels to appear in *both* resulting clusters.

D. Fitting quads

The next step is to fit a quad to each cluster of unordered boundary points, partitioning the points into four groups

function FINDBOUNDARIES(im, w, h)
  ▷ Find connected components using union-find
  $uf \leftarrow$ UnionFind($w \cdot h$)
  for each pixel (x, y) **do**
    for each neighbor (x', y') **do**
      if $im[x, y] = im[x', y']$ **then**
        uf.union($y \cdot w + x, y' \cdot w + x$)
      end if
    end for
  end for
  ▷ Group pixels which form a continuous boundary
  $h \leftarrow$ HashTable()
  for each pixel (x, y) **do**
    for each neighbor (x', y') **do**
      if $im[x, y] \neq im[x', y']$ **then**
        $r_0 \leftarrow uf$.find($y \cdot w + x$)
        $r_1 \leftarrow uf$.find($y' \cdot w + x'$)
        $id \leftarrow$ ConcatenateBits(Sort(r_0, r_1))
        if $id \notin h$ **then**
          $h[id] \leftarrow$ List()
        end if
        $p \leftarrow (\frac{x+x'}{2}, \frac{y+y'}{2})$
        $h[id]$.append(p)
      end if
    end for
  end for
end function

Fig. 4: Algorithm for continuous boundary segmentation. The neighbors of (x, y) using 8-connectivity are $\{(x + 1, y), (x - 1, y + 1), (x, y + 1), (x + 1, y + 1)\}$.

corresponding to line segments. However, computing the optimal partition which minimizes total line fit error is computationally expensive. Even for an ordered list of n points, there are $O(n^4)$ possible ways to partition the points. Our method computes an approximate partition by finding a small number of corner points, then iterating through all possible combinations of corner points.

First the points are sorted by angle in a consistent winding order around their centroid. This ordering allows us to define "neighboring points" as ranges of sorted points. Cumulative first and second moment statistics are computed in a single pass through these points, enabling the first and second moments to be computed for any range of points in constant time.

Corner points are identified by attempting to fit a line to windows of neighboring points, and finding the peaks in the mean squared error function as the window is swept across the points. Line fits are computed using principal component analysis (PCA) [16], in which an ellipse is fit to the sample mean and covariance. The best fit line is the eigenvector corresponding to the first principal component. Using the precomputed statistics, all the candidate line fits may be computed in $O(n)$ time, where n is the number of points. The strongest peaks in the mean squared error are identified as candidate corners.

Finally, we iterate through all permutations of four candidate corners, fitting lines to each side of the candidate quad. At this step we select the four corners which result in the smallest mean squared line fit errors. Prefiltering is performed to reject poor quad fits, such as those without at least four corners, whose mean squared errors are too large, or whose corner angles deviate too far from 90°.

The quad fitting step outputs a set of candidate quads for decoding (Figure 3d). Note that the quad detector correctly finds many quad-shaped structures in the environment, including specularities, switches, and individual tag pixels. The decoding step, which compares the contents of the quad to known codewords, filters out the false quad candidates.

E. Quick decoding

A straightforward approach to decoding tags is to XOR the detected code (in each of its four possible rotations) with each the codes in a tag family. A tag is identified as the code with the smallest Hamming distance from the detected code. However, if we limit the number of bit errors corrected to two bits or fewer, it is possible to enumerate all $O(n^2)$ possible codes within two bit errors of valid codes in a tag family. These codes can be precomputed and stored in a hash table, speeding up decoding from $O(n)$ comparisons to $O(1)$, where n is the size of the tag family.

F. Edge refinement

The threshold image, while useful for segmentation and quad border detection, may introduce noise into the threshold image. For example, shadows and glare can impinge upon an edge after thresholding, leading to poor localization accuracy with the resulting tag. We provide an optional, computationally inexpensive method to refine the edge using the original image.

The idea is to use the image gradient along the edges of the candidate quads to fit new edges, approximating the behavior of the original AprilTag detector. Along each edge, at evenly-spaced sample points, we sample the image gradient along the normal to the edge to find the location with the largest gradient. Knowing tags are dark on the inside and the winding order of the points in the quad, we reject points whose gradient is not the expected sign (i.e. from noisy individual pixels). We compute a weighted average of the points along the normal, weighted by the gradient magnitude. The line fit along these weighted average points are then used as the edges of the quad. The quad corners are computed as the intersections of these lines.

Edge refinement is not crucial if one is only interested in detecting tags, although it can help with the decoding of very small tags. However, the edge refinement step improves localization accuracy when tags are used for pose estimation.

	Candidate quads	False detections	False positive rate
Old	51,075,971	145	0.000284%
New	13,623,725	6	0.000044%

TABLE I: False positive rate on the LabelMe dataset (421,049 images). Both detectors used AprilTag-36h11 with up to 2 bit errors corrected, which has a theoretical false positive rate of 0.000570%. Noisy image regions were most likely to be decoded as false positives. The new detector's continuous boundary segmentation is less likely to fit a quad to noise, further reducing an already low false positive rate.

IV. Experimental Results

A. False positive rate

A key advantage of AprilTags is its resiliency against false positive detections in natural scenes. The previous detector was shown to have a lower rate of false positives than theoretically expected, largely due to the complexity heuristic in the tag generation process. We note that the number of false positives is not only a feature of the tag codewords themselves, but also a function of the number of candidate quads generated by the detector. A detector which generates fewer candidate quads should be expected to generate fewer false positives.

We ran an experiment to compare the performance of the new detection algorithm against the previous one, using the same LabelMe [17] dataset as the previous paper. This dataset consists of images of natural scenes, none of which contain AprilTags. Note that the likelihood of false positives is intentionally increased by allowing up to 2 bit errors to be corrected. The number of false positives is reduced by more than we would expect from the lower quad detection rate alone (Table I). The detector is also more selective, resulting in a lower false positive rate. An analysis of the images which generated false positives shows that noisy image regions were more likely to accidentally decode to a valid codeword. The continuous boundary segmentation in the new detector

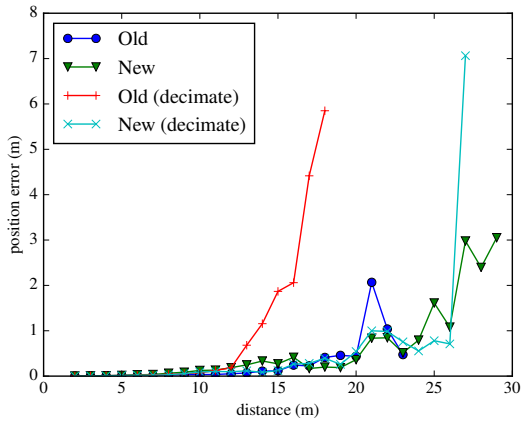


Fig. 5: Position error vs. distance. When decimated images are used, localization performance is similar to the old detector's performance using full-size images.

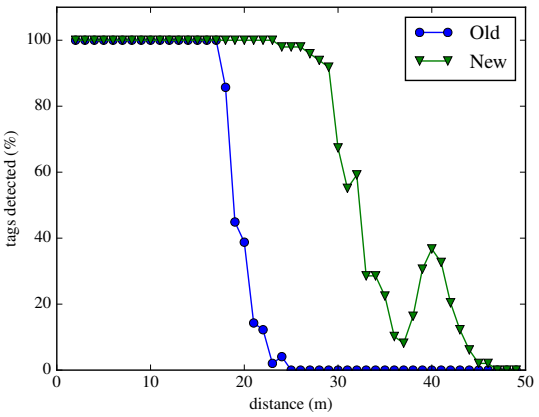


Fig. 7: Percentage of tags detected vs. distance. The new detector is better able to detect tags which are small and/or far away.

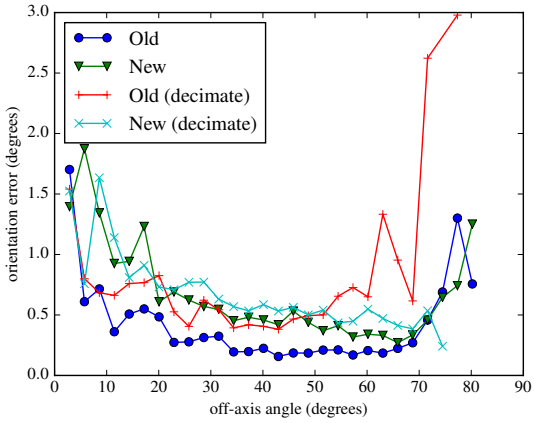
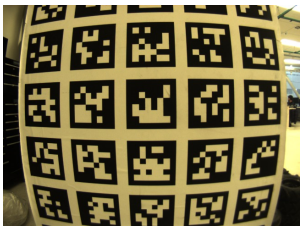


Fig. 6: Orientation error vs. off-axis angle. Both variants of the tag detector perform similarly at small angles, but the new detector performance does not degrade as quickly as the old one when using decimated images.



(a) 0.6m (closest)

(b) 7.0m (farthest)

Fig. 8: AprilTag mosaic for distance test on real images. Images were taken with a Point Grey Chameleon at 1296 x 964 pixels, and each tag is 0.167m wide. This experiment shows that the tag detection and localization performance observed in simulated images translates to real imagery. (Rectification was performed before tag detection and localization.)

algorithm is likely responsible for this increased robustness, as it is less likely to fit a candidate quad to noise.

B. Localization accuracy

To characterize the localization accuracy of our detector, we generated raytraced images with an ideal pinhole camera model, where the tags' true position and orientation were known. A single tag with known side length was placed in the scene while varying the distance and orientation.

In the first experiment, tag positions were generated randomly, constrained to be a fixed distance from the camera center, while the orientation of the tag was fixed parallel to the image plane. The error in estimated distance is plotted with respect to the distance of the tag from the camera (Figure 5).

In the second experiment, the tag position was fixed so that the camera's optical axis passed through its center. The tag orientation was generated randomly, constrained so that its normal vector makes the same angle with the camera axis. The error in estimated orientation is plotted with respect to the off-axis angle (Figure 6).

Both localization error experiments were repeated with the same images decimated to half their original size. The new detector vastly outperforms the old detector when decimated images are used, without noticeably affecting localization accuracy vs. non-decimated images. This observation is borne out by the detection rate as the tags are moved farther away in the simulated images; the new detector is far more capable at detecting small tags (Figure 7). The ability to decimate input images is one of the keys to the computational efficiency of the new detector.

Another question we seek to address is whether the simulated results will translate to real world performance. To answer this question, we collected real images of a large AprilTag mosaic at increasing distances. The camera was aligned with the center tag of the mosaic, and moved perpendicularly away from the mosaic plane. The ground truth was measured using a laser tape measure. The estimated distance to the center tag is plotted in Figure 9. In addition to improved localization accuracy, the new detector also detects tags at the full range of distances, while the old detector

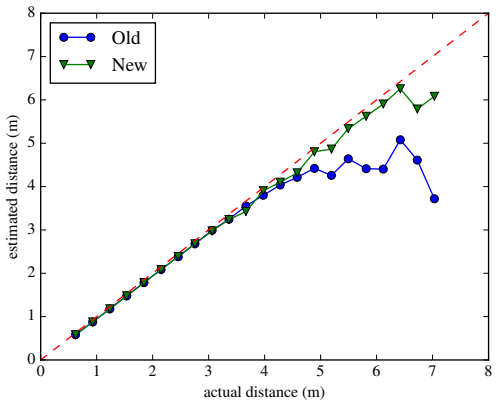


Fig. 9: Estimated distance to the tag mosaic center, showing the extended range and accuracy on real data.

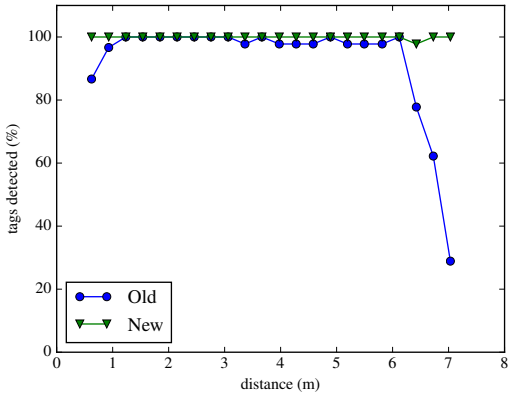


Fig. 10: Percentage of tags detected using real data, showing the improvement in detection range.

experiences a rapid fall-off in detection rate (Figure 10).

C. Computation time

In the LabelMe experiment, we logged the dimensions and wall time needed to process each image. Both tag detectors were run in single-threaded mode on an Intel Xeon E5-2640 2.5GHz core. The time per pixel, averaged across all images in the dataset, was about 0.254 microseconds per pixel for the new detector, compared to 0.374 microseconds per pixel for the old. This translates into about 78 ms and 115 ms, respectively, for a 640 x 480 image. (The absolute times are not meant to be representative, and are meaningful only in relation to each other. Computation time varies by processing speed and the number of quads in an input image.)

As we showed above, using decimated images with the new detector does not significantly affect localization error. With decimation by a factor of 2, the new detector only takes 0.072 microseconds per pixel, or about 22 ms for a 640 x 480 image. The detector performance is good enough to run on the relatively lower-powered iPhone and similar smartphone processors, opening up new possibilities in embedding AprilTags into small-scale applications.

V. Conclusion

This paper describes a new AprilTag detection algorithm which improves upon the previous detector, reducing the rate of false positives, increasing the detection rate, and reducing the amount of computing time needed for detection. These improvements make robust tag detection viable on computation-limited systems such as smartphones, and extends the usefulness of tag tracking in real-time applications. A free AprilTag detector app is available in the iPhone App Store[1]. The detector implementation, which was released last year, is open-source and freely available on our website[2].

References

[1] H. Kato and M. Billinghurst, "Marker tracking and hmd calibration for a video-based augmented reality conferencing system," in *Augmented Reality, 1999.(IWAR'99) Proceedings. 2nd IEEE and ACM International Workshop on*. IEEE, 1999, pp. 85–94.

[2] M. Fiala, "ARTag, a fiducial marker system using digital techniques," in *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, vol. 2. IEEE, 2005, pp. 590–596.

[3] E. Olson, "AprilTag: A robust and flexible visual fiducial system," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, May 2011, pp. 3400–3407.

[4] D. P. Mersch, A. Crespi, and L. Keller, "Tracking individuals shows spatial fidelity is a key regulator of ant social organization," *Science*, vol. 340, no. 6136, pp. 1090–1093, 2013.

[5] D. Wagner and D. Schmalstieg, "ARToolKitPlus for pose tracking on mobile devices," in *Proceedings of 12th Computer Vision Winter Workshop*, 2007.

[6] ——, "Making augmented reality practical on mobile phones, part 1," *Computer Graphics and Applications, IEEE*, vol. 29, no. 3, pp. 12–15, 2009.

[7] R. Bencina, M. Kaltenbrunner, and S. Jorda, "Improved topological fiducial tracking in the reactivision system," in *Computer Vision and Pattern Recognition-Workshops, 2005. CVPR Workshops. IEEE Computer Society Conference on*. IEEE, 2005, pp. 99–99.

[8] E. Costanza and J. Robinson, "A region adjacency tree approach to the detection and design of fiducials." *Vision, Video and Graphics (VVG)*, pp. 63–70, 2003.

[9] A. Xu and G. Dudek, "Fourier tag: a smoothly degradable fiducial marker system with configurable payload capacity," in *Computer and Robot Vision (CRV), 2011 Canadian Conference on*. IEEE, 2011, pp. 40–47.

[10] F. Bergamasco, A. Albarelli, L. Cosmo, E. Rodola, and A. Torsello, "An accurate and robust artificial marker based on cyclic codes," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. PP, no. 99, pp. 1–1, 2016.

[11] F. Bergamasco, A. Albarelli, and A. Torsello, "Pi-Tag: a fast image-space marker design based on projective invariants," *Machine vision and applications*, vol. 24, no. 6, pp. 1295–1310, 2013.

[12] A. Walters. (2015) ChromaTags: An accurate, robust, and fast visual fiducial system. Accessed on 2016-02-29. [Online]. Available: http://austingwalters.com/chromatags/

[13] N. Otsu, "A threshold selection method from gray-level histograms," *Automatica*, vol. 11, no. 285-296, pp. 23–27, 1975.

[14] C. Chow and T. Kaneko, "Automatic boundary detection of the left ventricle from cineangiograms," *Computers and biomedical research*, vol. 5, no. 4, pp. 388–410, 1972.

[15] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms*, 3rd ed. The MIT Press, 2009.

[16] K. Pearson, "On lines and planes of closest fit to systems of points in space," *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, vol. 2, no. 11, pp. 559–572, 1901.

[17] B. C. Russell, A. Torralba, K. P. Murphy, and W. T. Freeman, "Labelme: a database and web-based tool for image annotation," *International journal of computer vision*, vol. 77, no. 1-3, pp. 157–173, 2008.

[1] `https://itunes.apple.com/us/app/id736108128`
[2] `http://april.eecs.umich.edu/apriltag/`