# Flexible Layouts for Fiducial Tags

Maximilian Krogius Acshi Haggenmiller Edwin Olson

***Abstract*— Fiducials are artificial features with a variety of uses in computer vision such as object tracking and localization. We propose the idea of *flexible tag layouts* for visual fiducial systems. In contrast to traditional square tags, flexible tag layouts allow circular, annular, or other shapes as desired. One use of layout flexibility is to increase the data density of standard square shaped tags. In addition, we describe a detector that is faster and has higher recall than both the AprilTag 2 and ArUco detectors while maintaining precision.**

## I. Introduction

Fiducials are artificial visual features that are designed to be easy to detect. They have applications in computer vision, augmented reality, and robotics since they greatly simplify the perception problem. Popular fiducial systems have converged onto a standard layout with an easy-to-detect square border surrounding a unique pattern of data bits as in Figure 2a. However, this shape is not a perfect solution for all applications.

We have identified three problems with the traditional square layout. First, a large portion of the area of a standard layout fiducial tag consists of the detection border of the tag. This leaves less area for the data bits of the tag. Second, a square tag does not make efficient use of space on a circular object. For instance, the AprilTag fiducials have have been used to track everything from bees [1] to tiny circular robots [2], where these square tags fit like a square peg in a round hole. Third, applications such as marking UAV landing sites would benefit from tags which are detectable over a wider range of distances than is possible with traditional layouts.

Our proposed system addresses these issues by allowing the tag's layout to be customized for each application. We can generate layouts with a higher data density and smaller border (Figure 2b, c), circular tags (Figure 2e, f), or a custom tag layout (Figure 2f) with empty space in the middle in which a smaller tag could be placed, allowing this sort of recursive tag to be detected over a large range of distances. This sort of recursive tag can be used for marking quadrotor landing sites as in Figure 1.

One challenge in making flexible tag layouts is maintaining the low false positive rate that gives fiducials their utility. Previous fiducial systems achieve these low rates through the use of a complexity metric[3] which predicts the likelihood of the tag being similar to a naturally-occurring pattern. Finding a complexity metric which is effective across many tag layouts is necessary for maintaining robustness in new tag families.

The authors are with the Computer Science and Engineering Department, University of Michigan, Ann Arbor, MI 48104, USA. {mkrogius,acshikh,ebolson}@umich.edu

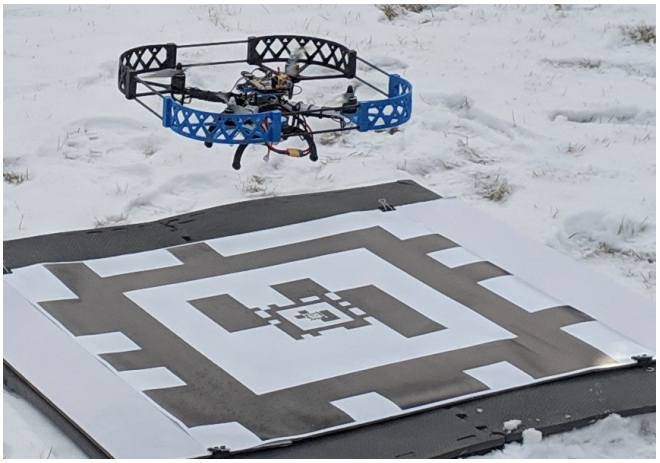

Fig. 1. A quadrotor using a doubly nested AprilTag for localization above its landing pad. This tag consists of a large recursive tag with a smaller recursive tag pasted inside of it and an even smaller recursive tag pasted inside of that one.

Our proposed tag designs could be adapted to many different visual fiducial systems. To enable direct comparisons to traditional tags we have incorporated our work into the popular AprilTag [4] fiducial system. Some examples of AprilTag use in recent years include providing ground truth for robotic arms [5], calibrating camera-lidar systems [6], allowing UAVs to track cars [7], and tracking construction materials [8].

We propose an evaluation scheme for fiducial systems which allows fair comparisons between detectors which are tuned for different use cases. Two key metrics for a fiducial system are the speed of the detector and the distance at which tags can be detected. There is a natural tradeoff between these two properties since spending more time in the detector's algorithm to increase the detection distance will lead to a reduction in detection speed. We show that our detector is faster and has a longer detection distance than previous fiducial systems.

The contributions of this work include:

- We introduce a flexible layout system whereby users can generate sets of tags with the data bits arranged in a specified shape (with a few restrictions).
- We introduce a complexity metric applicable to diverse tag layouts which we use to generate tags with low false positive rates.
- We introduce an evaluation scheme to fairly compare the speed and recall of fiducial detectors. We compare our detector against the AprilTag 2 and ArUco detectors.

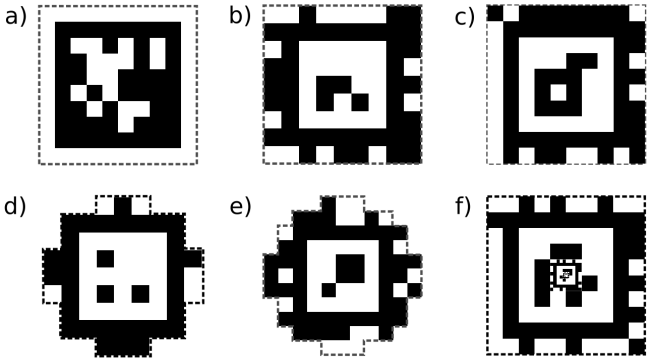


Fig. 2. Examples of tag families possible with AprilTag 3, with dotted lines showing the tag borders. a) The old-style tag family from AprilTag 2. b) and c) Tags with one layer of bits outside of the border with 41 and 52 data bits respectively. d) and e) Circular tags with 21 and 49 bits respectively. f) Tag with empty space in the middle for recursive tag applications.

## II. Related Work

We can categorize fiducial systems into two categories based on the shape of the tags: square tags with a black and white border on the outside; and tags with other shapes.

For square tags, one of the earliest examples was ARToolkit [9] which used a black border and an image on the inside of the tag. ARTag [10] introduced the 2D barcode to make tag decoding easier. AprilTag [3], [4] introduced a lexicode-based tag generation method to reduce false positive detections and a more-efficient detection algorithm. ArUco [11], [12] used mixed integer linear programming to generate tags and a detector also compatible with ARTag and AprilTag. ChromaTag used red-green color gradients to speed up the detection process[13]. While we have implemented our layouts with monochrome tags, the same idea could be applied to color tags like ChromaTag.

There were also many systems that used non-square tags. CALTag [14] used a grid of square tags to increase localization precision and occlusion robustness. RUNE-tag [15] used a tag composed of circular dots arranged in circular patterns. Fourier tags [16] proposed circular tags with bits encoded in the frequency of the radial intensity function of the tag. ReacTIVision [17] used tags of varying shape identified by their topology.

In contrast to the wide variety of non-square tag layouts, square fiducial tag systems appear to have converged onto a relatively standard 10x10 layout. Both AprilTag 2 and ArUco support this layout. In particular, no current fiducial system uses a non-square overall layout while maintaining the proven square detection border from these tag systems.

## III. Methods

We propose a method for generating and detecting tags with a flexible (i.e. user-specified) layout. This includes a complexity metric for reducing the false positive rate as well as a fast detector with high recall.

### A. Flexible Layout

Previous square fiducial systems have allowed only one layout parameter to vary: the size of the tag. The shape of the tag and location of the data bits within the tag were always the same. Our system allows the locations of the data bits to be arbitrary. Our tags are no longer necessarily square, except in that each individual bit is still represented by a square black or white pixel.

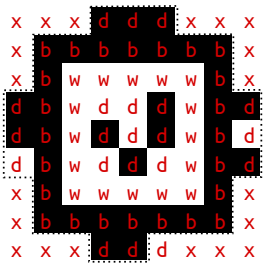


Fig. 3. The layout string for the 21h7 circular tag family overlaid onto an example tag from this family.

The layout of a tag is now specified as a string where each character corresponds to one pixel of the resulting tag family. There are four options: white ('w'), black ('b'), data ('d'), and ignore ('x'). Given this string, our system generates a family of tags with that layout. For example, our circular layout uses the ignore ('x') character in the layout string to create a raster approximation of a circle (see Figure 3).

The layout must have fourfold symmetry and contain a black and white border for detection of the tag. However, the data bits do not need to remain inside of the border, and the border may have either black or white on the outside.

We present a new standard layout which moves the border inwards and has one layer of data bits around the outside of the border (see Figure 2b), giving each tag an additional 16 data bits, regardless of overall tag size. While the smaller border potentially reduces pose accuracy and detection distance, the increase in data density and the corresponding decrease in false positive rate is desirable in many applications. We propose the name *uramaki* for these new style tags and the name *maki* for the old style tags since the placement of the data bits and border is analogous to the placement of the rice and seaweed in a sushi roll.

We can also specify a custom layout string in order to customize the tag shape, false positive rate, and number of tags. As an example we generate a recursive tag for use in marking quadrotor landing sites (see Figure 3f). The recursive tag uses a set of "ignore" bits in the center of the tag. Within that space we insert a smaller tag and within that smaller tag we insert an even smaller tag. The larger/middle tag will be detected from longer ranges but will exceed the field of view of the camera at shorter ranges, which is when the middle/smaller tag will be detected. (Of course, at some distances more than one tag may be detected, in which case the larger tag can provide additional localization precision).

### B. Complexity Metric

Some data bit patterns are more likely to appear in natural images by chance. For instance, a tag consisting of all zeros (which appears solid black) could be incorrectly detected on any dark rectangular object. Previous work showed that the

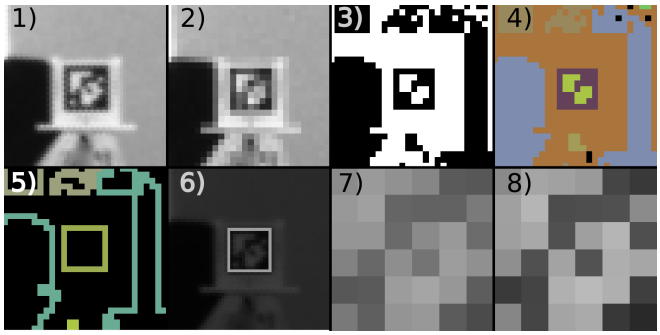


Fig. 4. The steps of the detection algorithm, illustrated by running the algorithm on a real image of a 36h11 tag. 1) The input. 2) Decimated, by a factor of two in this example. 3) Thresholded. 4) Connected components from a union-find algorithm. 5) Contours between white and black components. 6) Quadrilaterals fit to the contours. 7) Perspective correction and data bits are read out. 8) The data bits are sharpened and finally decoded.

false positive rate can be lowered if we only generate particularly unlikely tag patterns using a complexity metric [3]. For our flexible layouts it was not obvious that this particular choice of complexity metric would be the best. We tested the following three metrics and chose the Ising model based on our experimental evaluation.

Our first complexity metric is an extension of the rectangle complexity metric [3]. Our method renders the whole tag, including the border, and computes the number of (potentially overlapping) rectangles required to render that image. For tag layouts with the 'x' pixel we render the image twice, once each with all the 'x' bits set to either white or black, taking the minimum complexity of the two.

Our second metric is the Ising model energy of the rendered image, ignoring the 'x' bits.

$$E = \Sigma_{i,j}(\mathbb{1}[\text{image}(i,j) \neq \text{image}(i,j+1)] + \mathbb{1}[\text{image}(i,j) \neq \text{image}(i+1,j)]) \quad (1)$$

This is equivalent to the sum total length of edges between white and black pixels in the tag. This metric comes from the intuition that objects in the real world tend to have correlation in intensity between neighboring regions.

Our third metric is the number of connected components in the tag. A connected component here is defined to be a white or black 4-connected set of pixels.

### C. Detector Speed-up

While not essentially linked to the flexible tag layouts, we made several improvements to the AprilTag detector that improve its performance. We use the same high-level detection algorithm as AprilTag 2 [4], but have changed key parts of the algorithm to increase the overall speed. For reference, the main steps of the detection process can be seen in Figure 4.

*1) Decimation:* The decimation step of the algorithm reduces the size of the image, increasing the speed of the later steps. Choosing the decimation factor allows a trade off between recall and speed.

AprilTag 2 used a box filter for decimation. The value of each output pixel was computed by averaging the values of multiple input pixels in a "box" around the output pixel. We instead use point sampling. The value of the output pixel is set to the value of a single pixel from the input image.

```
1:  function UNIONFIND_IMAGE_LINE(image, y)
2:      for x = 1 to width(image) do
3:          if image(x, y) = image(x − 1, y) then
4:              union((x, y), (x − 1, y))
5:          end if
6:          if x = 1 or image(x−1, y) ≠ image(x−1, y−1)
            or image(x − 1, y − 1) ≠ image(x, y − 1) then
7:              if image(x, y) = image(x, y − 1) then
8:                  union((x, y), (x, y - 1))
9:              end if
10:         end if
11:     end for
12: end function
```

Fig. 5. The union-find algorithm which is run for each line of the image to find connected components. This is a standard union-find algorithm with the addition of the *if* statement on line 6. This additional *if* statement serves to reduce the number of calls to "union".

A box filter more closely approximates the low-pass filter typically used before decimation in other image processing applications while point sampling is better at preserving edges but is prone to creating aliasing artifacts. We experimentally determined that point sampling performs better for tag detection. This makes sense since the first stage of the tag detection pipeline relies on detecting edges in the input image.

*2) Union-Find:* The speed of finding connected components has been improved by reducing the number of calls to the union-find data structure. By tracking the values of the input image we can avoid redundant calls between inputs that have already been unioned, see Figure 5.

In addition we do early rejection of connected components that are so small that we know they could not produce a decodable tag.

*3) Fitting Quadrilaterals:* The algorithm fits quadrilaterals to the detected contours in the image. First points in a contour are sorted by their angle relative to the center of the contour's bounding box. Then four lines are fit to the set of points in the contour. A few changes have been made to speed up this process.

Computing the angle of all the points in each contour with respect to the bounding box center was time-consuming because of an expensive call to `atan2`. Instead of computing the sort key by computing the angle, the new detector computes the sort key as a combination of the quadrant the point falls into and the slope of the point within that quadrant.

$$\text{key} = \begin{cases} 2^{17} - x/y & \text{for } x < 0 \ \& \ y > 0 \\ 2^{16} + y/x & \text{for } x > 0 \ \& \ y > 0 \\ -x/y & \text{for } x > 0 \ \& \ y < 0 \\ -2^{16} + y/x & \text{for } x < 0 \ \& \ y < 0 \end{cases} \quad (2)$$

The value of $2^{16}$ is used since this is larger than any possible value of $y/x$ given the size of typical images. This gives a sort key that results in the same sorting order as the actual angle.

The next part of the algorithm fits lines to the contour. This requires computing the error of many different candidate least-squares line fits. The previous detector computed the normal to the line in order to compute the standard deviation of the fit error. We instead calculate the line fit error as the least eigenvalue of the covariance matrix of the points we are fitting a line to. This removes a call to `sin`/`cos` and adds only a call to `sqrt` while computing the same quantity.

*4) Tag Decoding:* The first step is to do perspective correction. This requires finding a transformation between the desired pose and the actual pose of the tag. In other words we want to solve for the 3x3 matrix $H$ such that:

$$\begin{bmatrix} w_i x_i^1 \\ w_i y_i^1 \\ w_i \end{bmatrix} = H \begin{bmatrix} x_i^0 \\ y_i^0 \\ 1 \end{bmatrix} \tag{3}$$

where the subscript $i$ runs from 1 to 4 for the points at each corner of the tag. Previously the algorithm used SVD to solve this 9x9 $Ax = 0$ equation. If we fix the scale of the matrix $H$ by adding the restriction that $H_{33} = 1$ then we can write this problem in the form of an 8x8 $Ax = b$ equation and solve using Gaussian elimination.

The restriction $H_{33} = 1$ fixes the scale of the matrix, but it also restricts us to fitting homographies where $H_{33} \neq 0$. We can justify this restriction by considering how the homography transforms the center of the tag in the case where $H_{33} = 0$. In this case we have:

$$\begin{bmatrix} w_i x \\ w_i y \\ w_i \end{bmatrix} = \begin{bmatrix} H_{13} \\ H_{23} \\ 0 \end{bmatrix} = H \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \tag{4}$$

In other words, a homography with $H_{33} = 0$ will map the center of the tag to a point at infinity on the image plane, which is a case that does not concern us since such a tag would not be detected in the first place.

### D. Detecting Small Tags

Instead of sampling the pixel value at the center of each tag using the nearest neighbor pixel, AprilTag 3 uses bilinear interpolation to extract the pixel value at the center of each tag cell.

The the pixel values are read out into a 2D array and sharpened. This has the effect of making it more likely for a lighter pixel with darker neighbors to be read out as a white pixel. The sharpening should help to counteract the blurring that occurs in very small images of tags. For example, if the tag is from the 36h11 family, the value at the center of each cell is read out into a 6x6 array. This 2D array is then sharpened with a 3x3 Laplacian kernel, and then decoding is attempted on these modified values. In other words, the original array is convolved with the following kernel before decoding:

$$K = I + 0.25 \begin{bmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{bmatrix} \tag{5}$$

### E. Experiment Setup

We evaluate the choice of complexity metric by measuring the false positive rate using the LabelMe [18] dataset. This dataset consists of natural images containing no Apriltags so any detection on this dataset is a false positive. LabelMe has 207920 images from which 6090028 candidate quads were extracted by the detector for the 41h12 family and 6128551 for the 21h7 family.

The different complexity metrics result in slightly different numbers of tags so we used the first 1500 tags for each of the 41h12 families and the first 35 tags for each of the 21h7 families to make these experimental results directly comparable within each group of tag families.

We evaluate the speed of the detectors on a dataset we have produced, consisting of 160 images of tags from the *maki* 36h11 family and 160 images of tags from the *uramaki* family. The images have a resolution of 1296x964 and were taken with a Point Grey Chameleon camera. Ten different tags from each family were photographed at distances from 20cm to 160cm from the camera in increments of 20cm. At each distance there is one image of the tag face-on to the camera and another image with the tag rotated 45 degrees away from the camera. Each tag was printed at a size of 4 cm measured across the outer limits of the tag.

We compare the performance of the ArUco detector using the DM_FAST mode, the old AprilTag 2 detector, and the new AprilTag 3 detector on the *maki* 36h11 images and the new AprilTag 3 detector on the *uramaki* 41h12 images. We vary the decimation parameter of the AprilTag 2 detector between 1 and 17, the decimation parameter of the AprilTag 3 detector between 1 and 24, and minMarkerSize parameter of the ArUco detector between 0 and 0.11 in steps of 0.01. These parameter choices were chosen to go from the minimum value up to the value where each detector fails to detect any tags at all. The speed of each detector was measured on an Intel® Core™ i7-7600U CPU running at 2.80GHz.

## IV. Results and Discussion

### A. Flexible Layout

We used the new flexible layout system (along with the new complexity metric, see subsection IV-B) to generate several new tag families. The first is the new *uramaki* AprilTag (see Figure 2b) with a hamming distance of 12. This tag family is 9x9 cells in size and has 2115 tags in total which compares favorably to the old *maki* 36h11 family (see Figure 2a) which is 10x10 cells in size and has only 587 tags. Putting data bits outside of the border of the tag allowed us to increase the proportion of the tag that consists of data instead of border. To emphasize, the new 9x9 *uramaki* tag layout has a greater hamming distance and encodes a larger number of tags than the conventional 10x10 *maki* tag, even though it has fewer overall cells. If printed at the same physical size, the 9x9 tag would have a larger bit pitch, which improves its detection range. In short, this improvement in tag performance is the main contribution of this paper.

| | Bit Errors Corrected | | | |
|---|---|---|---|---|
| **Complexity Metric** | **0** | **1** | **2** | **3** |
| Rectangle Complexity | 0 | 1 | 8 | 99 |
| Connected Components | 0 | 0 | 11 | 105 |
| Ising Energy | 0 | 0 | 4 | 44 |
| Expected (Random bits) | 0.0 | 0.2 | 3.4 | 44.3 |

TABLE I

False positives as a function of bit errors corrected on the LabelMe [18] dataset for the 9x9 *uramaki* 41h12 tag layout using different complexity metrics to generate tag families. The last row of the table shows the number of false positives that would be expected if bits were randomly distributed.

| | Bit Errors Corrected | | | |
|---|---|---|---|---|
| **Complexity Metric** | **0** | **1** | **2** | **3** |
| Rectangle Complexity | 152 | 3273 | 40462 | 303028 |
| Connected Components | 131 | 3600 | 45052 | 320174 |
| Ising Energy | 85 | 1867 | 22266 | 165862 |
| Expected (Random bits) | 102.3 | 2147.9 | 21479.1 | 136034.0 |

TABLE II

False positives as a function of bit errors corrected on the LabelMe [18] dataset for the circular 21h7 tag layout using different complexity metrics to generate tag families. The last row of the table shows the number of false positives that would be expected if bits were randomly distributed.

We generated a 10x10 tag family with the *uramaki* layout which has 52 data bits, a hamming distance of 13, and 48714 unique tags. These new tag families with many more possible tags could enable new applications such as tracking deformable objects or larger-scale deployments of robots.

We also made two circular tag families (see Figure 2d and Figure 2e). The smaller circular tag family has 38 unique tags and the larger one has 65698 unique tags. For applications in which tags were mounted on circle-shaped objects [1], [2], being able to make better use of the available space can result in a significant improvement in minimum hamming distance or number of distinct tags.

Finally, we generated a *recursive* tag family which has one layer of data bits on the outside, and has a 2x2 hole in the middle where there are no data bits. We printed out three tags from this family at different sizes such that each tag could be placed in the middle of another tag. We used this $90.2x90.2$ cm doubly nested tag to localize a quadrotor flying above its landing area as seen in Figure 1. This tag allows detection over a long range of distances, from $0.08 - 16.15 \pm 0.03$m, even using a low resolution 640x480 image from the onboard Raspberry Pi Camera Module V2.

### B. Complexity Metric

The false positive rates of the tag families generated using the three different complexity metrics were compared on the LabelMe [18] dataset (see subsection III-E). We can see from Table I and Table II that the best performing complexity metric for both the 41h12 and 21h7 families is the Ising energy. For the 21h7 family the Ising energy metric reduces the false positive rate even below the rate expected for a uniform random distribution of bits. Thus in AprilTag3 we use the Ising model exclusively.

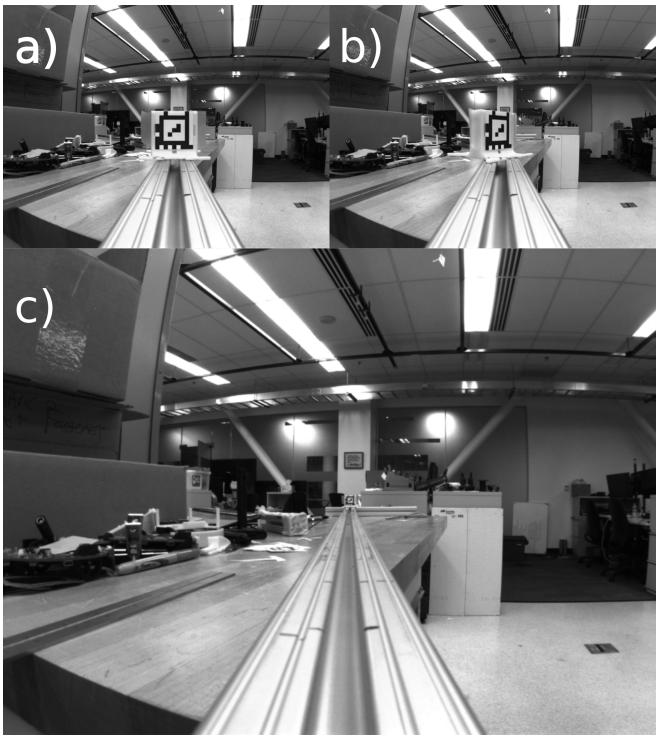


Fig. 6. Example images from the dataset. a) Tag at 20cm, face-on. b) Tag at 20cm, rotated 45 degrees. c) Tag at 160cm, face-on.

### C. Detector Speed

Both the AprilTag and ArUco detectors have parameters which allow a trade-off between speed and recall. For the AprilTag detector this is the decimation factor and for the ArUco detector it is minMarkerSize. Because of these parameters, it is easy to draw incorrect conclusions about the relative performance of two fiducial systems since they may be tuned differently. We do a parameter sweep of the relevant parameter for both detectors on a dataset containing a variety of different size tags. The results of this comparison are shown in Figure 7. We can see that the AprilTag 3 detector is faster and has higher recall than both the AprilTag 2 and ArUco detectors.

We believe that it is important to compare the speed and recall of different detectors by showing a parameter sweep for each detector, on a dataset that includes tags of all different sizes. The main tradeoff in a fiducial detector is between recall, particularly of small tags, and speed. Comparing the detectors in this manner allows a fair comparison between systems which may have different default parameter settings.

We can also compare the performance of the new detector when using the *uramaki* 41h12 tag family against the new detector using the *maki* 36h11 tag family. When tuned for maximum recall, the new detector has roughly equivalent performance using either family. However, when tuned for greater speed (high decimation factor), the recall of the 41h12 tag family is less than that of the 36h11 tag family. This may be an artifact of the fact that all the tags in our dataset are photographed on a white background, which gives the 36h11 tag an effectively larger outer border.

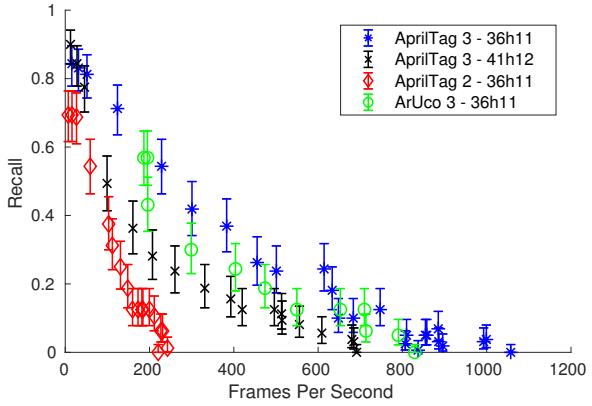


Fig. 7. Speed vs recall for the three different detectors on the 36h11 tag family as well as the combination of the new detector with the new 41h12 tag family. We do a parameter sweep for each detector to allow a sensible comparison between detectors which have different default parameter settings.

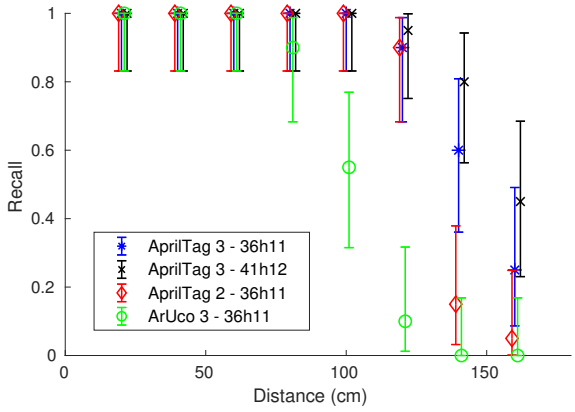


Fig. 8. Recall vs tag distance for the same cases as in Figure 7. Here we have chosen the parameter value for each detector that allows maximum recall in order to see each detector's maximum detection rate as a function of tag distance. We can see that the AprilTag 3 detector is capable of detecting tags at a greater distance than the other detectors.

### D. Detecting small tags

Using the same dataset as above we can look more closely at what causes the detectors to fail to detect tags. In Figure 8 we can see each detector's recall (with parameters set for maximum recall) plotted against the tag distance. The difference in recall between the detectors is mostly due to differences in the distance at which the detector begins failing to detect tags. It can be seen that the AprilTag 3 detector performs better than both the AprilTag 2 and ArUco detectors at detecting small tags.

## V. Conclusion

This paper proposes a fiducial system capable of generating and detecting fiducial tags in a wide variety of layouts. We introduce useful layouts including a higher data density square tag, circular tags, and recursive tags. We show an example of a recursive tag used for quadrotor localization above a landing site.

Our improvements to AprilTag, including both the new tag families and the performance improvements described in subsection III-C, have been incorporated into the open-source AprilTag distribution, maintained by the APRIL lab at the University of Michigan (https://april.eecs.umich.edu).

## References


[1] C. Blut, A. Crespi, D. Mersch, L. Keller, L. Zhao, M. Kollmann, B. Schellscheidt, C. Fülber, and M. Beye, "Automated computer-based detection of encounter behaviours in groups of honeybees," *Scientific Reports*, December 2017.

[2] M. P. Nemitz, M. E. Sayed, J. Mamish, G. Ferrer, L. Teng, R. M. McKenzie, A. O. Hero, E. Olson, and A. A. Stokes, "Hoverbots: Precise locomotion using robots that are designed for manufacturability," *Frontiers in Robotics and AI*, vol. 4, p. 55, 2017. [Online]. Available: https://www.frontiersin.org/article/10.3389/frobt.2017.00055

[3] E. Olson, "Apriltag: A robust and flexible visual fiducial system," in *2011 IEEE International Conference on Robotics and Automation*, May 2011, pp. 3400–3407.

[4] J. Wang and E. Olson, "Apriltag 2: Efficient and robust fiducial detection," in *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Oct 2016, pp. 4193–4198.

[5] C. Nissler, S. Buttner, Z. C. Marton, L. Beckmann, and U. Thomasy, "Evaluation and improvement of global pose estimation with multiple AprilTags for industrial manipulators," in *IEEE International Conference on Emerging Technologies and Factory Automation, ETFA*, 2016.

[6] D. Tang, T. Hu, L. Shen, Z. Ma, and C. Pan, "AprilTag array-aided extrinsic calibration of camera–laser multi-sensor system Background," *Robot. Biomim*, vol. 3, 2016.

[7] J. Wang, C. Sadler, C. F. Montoya, and J. C. L. Liu, "Optimizing ground vehicle tracking using unmanned aerial vehicle and embedded apriltag design," in *2016 International Conference on Computational Science and Computational Intelligence (CSCI)*, Dec 2016, pp. 739–744.

[8] C. Feng, Y. Xiao, A. Willette, W. Mcgee, and V. R. Kamat, "Towards Autonomous Robotic In-Situ Assembly on Unstructured Construction Sites Using Monocular Vision," in *Proceedings of the 31st ISARC*, 2014, pp. 163–170.

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

[10] M. Fiala, "Artag, a fiducial marker system using digital techniques," in *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, vol. 2, June 2005, pp. 590–596 vol. 2.

[11] S. Garrido-Jurado, R. Muñoz-Salinas, F. Madrid-Cuevas, and M. Marín-Jiménez, "Automatic generation and detection of highly reliable fiducial markers under occlusion," *Pattern Recognition*, vol. 47, no. 6, pp. 2280 – 2292, 2014.

[12] F. J. Romero-Ramirez, R. Muñoz-Salinas, and R. Medina-Carnicer, "Speeded up detection of squared fiducial markers," *Image and Vision Computing*, vol. 76, pp. 38–47, aug 2018.

[13] J. DeGol, T. Bretl, and D. Hoiem, "Chromatag: a colored marker and fast detection algorithm," in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 1472–1481.

[14] B. Atcheson, F. Heide, and W. Heidrich, "CALTag: High precision fiducial markers for camera calibration," in $15^{\text{th}}$ *International Workshop on Vision, Modeling and Visualization*, Siegen, Germany, Nov. 2010.

[15] F. Bergamasco, A. Albarelli, E. Rodolà, and A. Torsello, "Rune-tag: A high accuracy fiducial marker with strong occlusion resilience," in *CVPR 2011*, June 2011, pp. 113–120.

[16] J. Sattar, E. Bourque, P. Giguere, and G. Dudek, "Fourier tags: Smoothly degradable fiducial markers for use in human-robot interaction," in *Fourth Canadian Conference on Computer and Robot Vision (CRV '07)*, May 2007, pp. 165–174.

[17] R. Bencina, M. Kaltenbrunner, and S. Jorda, "Improved topological fiducial tracking in the reactivision system," in *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05) - Workshops*, Sept 2005, pp. 99–99.

[18] 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, pp. 157–173, May 2008.