Accelerated Fiducial Tag Detection Using Efficient Search Algorithm with Trajectory Forecasting

Rushil Saraf ¹ and Tony Smoragiewicz ²

¹Saratoga High School
²Affiliation not available

October 31, 2023

Abstract

Fiducial tags have attracted increased attention and use in the robotics community due to their relative ease of detection and pose estimation. Previous methods to detect these tags search the image pixel spaces in their entirety. With the advent of higher resolution video such as 4K and increased camera frame rate, however, it has become increasingly challenging to obtain real time tag detection, especially using lower-end computing devices.

We present a novel algorithm that accelerates the detection speed of these fiducial tags by forecasting the future trajectory of the camera, devising a prediction for the upcoming tag location in the camera frame, and significantly reducing the image search space corresponding to the prediction. Based on the data from several test trials, we tuned the algorithm parameters and compared it to the conventional full search method in multiple real-world test experiments using AprilTags. In addition to the quantitative results, we present a qualitative analysis on the optimal application scenarios for our new algorithm. Our method has demonstrated between a three to four times increase in detection speed and is robust enough to handle intermittent tag occlusion.
Accelerated Fiducial Tag Detection Using Efficient Search Algorithm with Trajectory Forecasting

Rushil Saraf  
Saratoga High School  
20300 Herriman Ave, Saratoga, CA 95070  
sarafrushil@gmail.com

Tony Smoragiewicz  
Northeastern University  
360 Huntington Ave., Boston, MA 02115  
smoragiewicz.t@northeastern.edu

Abstract—Fiducial tags have attracted increased attention and use in the robotics community due to their relative ease of detection and pose estimation. Previous methods to detect these tags search the image pixel spaces in their entirety. With the advent of higher resolution video such as 4K and increased camera frame rate, however, it has become increasingly challenging to obtain real-time tag detection, especially using lower-end computing devices.

We present a novel algorithm that accelerates the detection speed of these fiducial tags by forecasting the future trajectory of the camera, devising a prediction for the upcoming tag location in the camera frame, and significantly reducing the image search space corresponding to the prediction. Based on the data from several test trials, we tuned the algorithm parameters and compared it to the conventional full search method in multiple real-world test experiments using AprilTags. In addition to the quantitative results, we present a qualitative analysis on the optimal application scenarios for our new algorithm. Our method has demonstrated between a three to four times increase in detection speed and is robust enough to handle intermittent tag occlusion.

I. INTRODUCTION

The goal of this project was to increase the speed at which visual fiducial markers could be accurately detected. Visual fiducial markers have been useful tools for the purposes of pose estimation and object tracking in areas such as robotics and augmented reality. With reduced computations, devices with lower processing power could obtain real-time detection while higher end devices would see boosted performance and increased resource availability.

A. Background

Localization is one of the most challenging requirements of autonomous mobile robots. At the heart of the problem, a robot must possess knowledge of its 3D position and orientation relative to environmental objects around it. Critically, navigation and key functionalities of these mobile systems hinge upon accurate localization estimates, obtainable in a variety of ways.

One of the most useful methods for robot localization is wheel odometry, where motion and consequently position are calculated from wheel rotational data given by sensors. However, due to accumulated motion measurement errors, the position error given by odometry integration grows with time [1]. In many cases, a more absolute localization is desirable: a robot’s position should or can not be derived from its known starting point.

Visual odometry is a popular choice for absolute pose estimation. Data in the form of images or video from a camera is analyzed in order to calculate a robot’s location in relation to its environment [2]. Easily detectable and standardized landmarks purposefully placed in the surroundings can make this task considerably easier. The role of visual fiducial tags is exactly that: provide easily recognizable, artificial features in a natural environment. Outside the purpose of robot localization, they have been utilized in augmented reality applications, motion capture, and more.

Fig. 1: Example detection of an AprilTag fiducial marker

ARTag [3] and AprilTag [4] (used in this study) are all popular choices for visual fiducial tags. Due to their known scale and position, these tags can provide a full 6 DOF position estimate and will be the primary focus of investigation in this paper.

B. Problem Formulation

The detector intrinsic methods to increase speed come with costs in other areas. With AprilTags, for example, increasing the quad decimate parameter sacrifices detection distance, while increasing nthreads requires extra CPU cores [5]. A non-detector specific method of accelerating detection speed is by down scaling the resolution of the image. However, this too comes with a downside of smaller tags being less likely to be
Fig. 2: Overview of efficient search: First, the future camera trajectory is estimated using previous frame data. Then, the tag is projected from the world reference frame into the predicted camera reference frame. This projection is scaled according to accuracy data from previous frames. The AprilTag 3 detector is finally run on the reduced image search space.

detected [6]. In the use cases of high resolution video or larger detection distance, these methods would not be particularly well suited.

Some other methods have been proposed to increase fiducial detection speed without modifying the intrinsic detector. These include involve wisely scaling input images and using kernel correlation filters [7] [8].

We present a method that selects regions of the image data deemed to likely contain the marker, prioritizing search in those areas first. This comes at no cost of detection distance or limited tag size searchability and solely uses information from the camera and tag, making it suitable for a wide variety of applications. In the next section, we go into greater depth into the workings of our region selection algorithm.

II. ALGORITHMS

At a high level, the tag forecasting algorithm has four steps.

- The immediate future pose of the robot is calculated using a regression model utilizing previous positional data acquired from the tag.
- The tag corner points are projected from the world frame onto the future image frame based on the predicted next position of the camera.
- The projection is then rectangularized and scaled relative to the projection accuracy in previous frames.
- The input image is cropped according to the final prediction and the standard tag detection algorithm is run on the reduced search space.

To ensure tags are never missed, the algorithm will search the whole image if the tag is not located in the prediction. We detail the specifics of the four steps below.

A. Trajectory Forecasting

The rotation matrix and translation vector of the tag, which transform it from the world frame to the image plane, are extracted using the known coordinates of the fiducial corners within the world and camera reference frames. As shown in Eq. [1], the solvePnP function in the OpenCV library is used to find the solution minimizing the reprojection error for $\mathbf{R}_w$ and $\mathbf{t}$ given the camera intrinsic matrix [9].

$$
\begin{bmatrix}
  u \\
  v \\
  1
\end{bmatrix} =
\begin{bmatrix}
  f_x & 0 & c_x \\
  0 & f_y & c_y \\
  0 & 0 & 1
\end{bmatrix}
\begin{bmatrix}
  \mathbf{R}_w \\
  \mathbf{t}
\end{bmatrix}
\begin{bmatrix}
  X_w \\
  Y_w \\
  Z_w \\
  1
\end{bmatrix}
$$

After inverting the transformations as in Eq. [2] the new rotation matrix and translation vectors are then converted into the 6 dimensional vector representing the 3D pose of the camera: $\mathbf{q} = (x, y, z, \phi, \theta, \psi)$.

$$
\mathbf{wT}_c = \begin{bmatrix}
\mathbf{wR}_c & -\mathbf{wR}_c \mathbf{t} \\
0 & 1
\end{bmatrix}
$$

Based on the desired algorithm parameters, the previous $n$ poses are stored as described along with the previous $n$ values for the elapsed time since the start of execution. We use a least
where $a_0$, $a_1$, and $a_2$ are the coefficients of $a_2t^2 + a_1t + a_0$; $r_n$ is the $n$th previous elapsed time since start of execution, and $r_n \in (x_n, y_n, z_n, \phi_n, \theta_n, \psi_n)$.

To obtain the estimated future pose of the robot, a separate set of coefficients is determined for each of the 6 positional degrees of freedom. Then, $t_0$, the time between the current input frame and the start of execution, is substituted into each of the six functions to acquire the prediction.

For our study, we chose to use a constant acceleration model, so the regression polynomials were degree two.

Due to the nature of planar pose estimation, situations can occur where more than one pose can be physically plausible, especially at farther tag distances [10]. This ambiguity can occasionally result in an incorrect pose being solved for. For this reason, the algorithm has a protection condition to reject any euler angle changes physically impossible for the robot to reach within the time between frames. To account for measurement noise, we set this threshold at 10 degrees. If a pose is deemed incorrect, we use the last correct pose as the next recorded data point. There have been proposed solution methods which can better remove this ambiguity [11]. However, these were not investigated in this study.

**B. Tag Projection**

Given this predicted future pose, we obtain the future transformation matrices. The `projectPoints` function is used to procure the image coordinates of the next predicted locations of the four tag corners, by now solving for $u$ and $v$ in Eq. [1]

**C. Projection Adjustment**

This projection cannot immediately be used as the search space as it leaves a small margin of error for the prediction to capture the entirety of the true tag corner image locations. As such, we scale the projection space according to the formula in Eq. [3]

$$\text{scale factor} = 1 + k(1 - \text{accuracy}) \quad (4)$$

Accuracy in this scenario is a number from 0 to 1 which measures the quality of the previous marker location prediction. We quantify this using the Intersection over Union (IoU) metric as shown in figure 3. Essentially, the lower the accuracy of the previous forecast, the larger the scale factor will be for the next one to compensate.

$k$ is one of the algorithm parameters that can be adjusted by the user. It enlarges the tag region guess to account for uncertainty in camera movement. We will discuss our experimentation with this parameter in the results section.

**D. Reduced Image Search**

Finally, the image is cropped according to specification and the April Tag 3 detector algorithm is run on the reduced area. If the tag is detected, then the robot pose data will be extracted and the program will repeat again from step one. If the tag is not detected in the reduced search space, the marker detection algorithm will be run on the entire image as a fail safe. Likewise, the robot pose data will be obtained and the algorithmic cycle will start again.

**III. Experiments**

To test the performance of our selective search algorithm, we conducted multiple physical tests using a smartphone camera mounted onto a Roomba Robot Cleaner. The camera was set to record video with a resolution of 1920 x 1080 pixels and at 60 frames per second. Throughout all the tests, the robot travelled at a constant speed of approximately 0.3 m/s.

The robot was set to travel on 11 preselected paths three times each for a total of 33 runs. In 6 of the paths, the robot would drive in a straight line from the left to the right side of the room, maintaining a roughly constant distance from the front wall. These 6 "horizontal" paths varied in their set distance away from the front wall, from 1.0m to 3.5m in increments of 0.5m. The camera maintained a constant orientation facing the front of the room in all of these paths.

Along the other 5 paths, the robot travelled in a straight line away from the tag, perpendicular to the front wall. These 5 "vertical" paths varied in their set distance away from the vertical line passing through the center of the tag, from 1.0m left of the tag to 1.0m right of the tag in increments of 0.5m. Like the horizontal paths, the camera maintained a constant orientation facing the front wall.

One important note is that due to a slightly bumpy surface, the camera shook noticeably more on the vertical than the horizontal paths. This resulted in the vertical path videos having some frames with enough motion blur that the AprilTag detector could not identify the marker. Effectively these frames...
Fig. 4: The red arrows indicate the 6 unique horizontal paths while the blue arrows indicate the 5 vertical paths. The 4 rotation points are labelled P1-4 in green. Each unique path and point was tested 3 times each for a total of 45. All axis units are in meters and are in relation to the AprilTag set at the origin.

acted as tag occlusions. The data in the results section reflects this as the speed of the efficient search algorithm was lowest in the vertical paths, though still significantly faster than the full search method.

In addition to driving along these 11 paths, the robot was also rotated in place at 4 different points (as shown in figure 5) three times each for a total of 12 rotational runs. For the two points to the right of the tag, the robot was rotated counter clockwise. For the two points to the left of the tag, the robot was rotated clockwise.

The 45 captured trial videos were then trimmed to exclude the starting and ending portions where the AprilTag was not completely in the view of the camera. Furthermore, the videos were down scaled to a resolution of 1280 x 720 pixels due to processing limitations. Multiple programs were then tested on the video dataset using an Intel Core i5-1135G7 CPU @ 2.40GHz processor.

IV. RESULTS

After obtaining the video dataset, we tested the algorithm to find the optimal value of $n$, the number of previous data points used in the regression prediction, and $k$, the dynamic scaling factor. Using these tuned values we then compared the performance of our efficient search algorithm against the full search method.

A. Tuning n

The metric used to evaluate the algorithms with value of $n$ was the average IoU accuracy of the unscaled projections described in Fig 3. The results of testing $n = 1, 2, 3, 4$, and 5 on the video data sets can be found in Fig 5a, 5b, and 5c.

We will first analyze the horizontal path results. Within 2m of the front wall, $n = 5$ achieved the highest prediction accuracy averaging roughly an IoU value of 0.92. $n = 1, 2$ were not far behind. Past 2m, $n = 1$ performed the best followed by $n = 2$ and 5. Overall, increasing tag distance seemed to steadily lower projection accuracy.

For the vertical paths, $n = 1$ yielded the highest accuracy with $n = 2$ and 5 essentially tied right behind. These paths also support the notion that tag distance is inversely proportional to accuracy, as increased lateral distance reduced performance.

Finally, for the rotational paths, $n = 5$ was clearly the best, followed by $n = 4$. Unlike the other paths, $n = 1$ was noticeably behind the accuracy leader for the rotational tests. Again, tag distance significantly reduced accuracy, as points 1 and 3 attained much better results than 2 and 4.

Based on these results, we chose to use $n = 5$ during further algorithm testing due to its overall high accuracy regardless.

![Fig. 5: Resulting frames processed per second by varying n from 1-5 on horizontal, vertical, and rotational paths](image)
Fig. 6: Resulting frames processed per second by varying $k$ from 1-16 on horizontal, vertical, and rotational paths using $n = 5$

Fig. 7: FPS comparison between full search method and proposed efficient search method on horizontal, vertical, and rotational paths

of path. For close distances relative to the tag size, it seems as though using more data points leads to a better prediction. Farther away, we would advise using less previous data points as smaller tags seem harder to accurately track.

B. Tuning $k$

Now tuning for $k$, we use frames processed per second (FPS) to evaluate the speed of the algorithm with varied dynamic scaling factors. The results of testing $k = 1, 4, 8, 12,$ and $16$ with $n = 5$ on the video data sets can be found in Fig 6a, 6b and 6c.

For the horizontal paths, $k = 4$ and $8$ were clearly the best, performing similarly from all distances from the tag with an average FPS of roughly 50, with the exception when the frontal distance was 1m. $k = 12$ followed behind those two.

The results for the vertical paths were the same as the horizontal. $k = 4$and $8$ achieved a similarly high FPS, followed by $k = 12$.

For the rotational tests, $k = 8$ was slightly better than $k = 4$. Based on these results, we chose to use $k = 8$ for our comparison to the full search method. $k = 4$ performed nearly identically except for the rotation paths. For other data sets and values for $n$, other scale factors may yield faster detection speeds.

C. Final comparison

Using our tuned parameters, we run our efficient search against the full search method on our collected video data
sets. The results of testing on the 45 videos can be found in graphs Fig 7a, 7b, and 7c.

Overall, our proposed efficient search algorithm was significantly faster at marker detection than the full search method, regardless of path type or distance. For the horizontal paths, it was on average 4.0 times faster with an average of 57 FPS. For vertical paths, the difference was not as large as horizontal, as it was 3.5 times faster on average, achieving a mean FPS of 51. Finally, the rotational points’ 45 FPS mean was 3.3 times faster than full search.

V. Conclusion

We have described a novel selective search algorithm that significantly improves upon the speed of fiducial marker detection. By forecasting the future trajectory of the camera, our algorithm defines a likely region for future marker position, reducing the number of computations required to search for the tag. We also described our process for tuning the algorithm parameters and our quantitative results from comparing it to the traditional method.

Our method works best with smooth motion and high frame rate video feed. If the camera’s motion is more erratic or lower frame rate video is used, we would advise decreasing \( n \) and increasing \( k \) to maintain accurate tag prediction. In addition, our method can handle intermittent tag occlusion (demonstrated by the results from the vertical path), but it will slightly reduce the maximum speed gain.

While our initial algorithm has shown improvement over the full search method, there is still much do be done to ascertain its limits. Additional testing with faster robot drive speed, non linear paths, lower frame rate, and varied video resolution would help further develop the optimal and worst case scenarios for usage. Additionally, more sophisticated methods like machine learning could be investigated to see if even higher accuracy position predictions could be made and faster detection times could be achieved.

References
