Towards robust 6-DoF detection in uncontrolled lightning enviroments

Alejandro Mora, Edmundo Guerra, Manuel Manzanares, Antoni Grau
Automatic Control Dept.
Technical University of Catalonia, UPC
Barcelona, Spain
{edmundo.guerra, manolo.manzanares, antoni.grau}@upc.edu

***Abstract*—One of the most critical issues that arises when controlling a robot is the necessity to locate itself in the environment. Ranging from industrial processes applications to outdoor mobile robots, landmarks are used to such purpose: by recognizing them, the robots are able to know where they are with the aid of computer vision. Fiducial markers are the cheapest and one of the most common solutions to this issue: 2D planar patterns that embed some information that can be identified using artificial vision techniques. Many different typologies of markers as well as image processing algorithms are being implemented nowadays, using C++ / Python / MATLAB® libraries and ROS as middleware. In this work we have evaluated the robustness of various fiducial marker typologies, documenting the main technical aspects of the used implementations and commenting how each algorithm operates. Aggregated results are presented and discussed, evaluating the different implementations tested.**

***Keywords*— fiducial markers, ROS, pose estimation, performance evaluation.**

I. Introduction

A fiducial marker is a system of planar (2D) markers localized in a certain environment, whose purpose is to be detected by a camera in various applications [1]. It is possible to estimate the camera pose (and in turn, knowing its morphology, the pose of the robot featuring this camera) thanks to these markers, but assumptions about the following factors need to be made [2]:

- View angles: shouldn't be too high with respect to the perpendicular axis of the marker plane, in order to be able to recognize it.
- Occlusions: the recognition algorithm of the markers should be resistant to occlusions, and it should foresee situations in which the fiducial marker information is incomplete (not all the single markers are detected).
- Distance and light conditions: they should be as invariant in time as possible, so that the detection algorithms can work properly.
- However, these conditions are not always fulfilled. Outdoor applications are a very clear example: with a wide range of possible view angles and constantly changing distance and light conditions, it becomes a challenge to be able to complete the pose estimation task because of the difficulties appearing in the computer vision part of the algorithm.

To be able to compare different types of fiducial markers in terms of robustness and performance, up to 11 evaluation criteria are quantitatively measured and contrasted [3], ranging from false positive/negative detection rate, all the way to occlusion and lighting immunity and speed performance. These will be considered to decide which is the optimal fiducial marker class for each specific application.

II. Robotics Middleware for Detection and Measurement

In this work, several fiducial markers available in ROS (Robot Operating System) [4] are studied in order to assess their robustness and performance through a series of well-defined tests. This is vital information for outdoor operations, since the robot (whether it be terrestrial or aerial) must be localized and able to apply adequate control signals at all times, even if lighting conditions (amongst others) are unfavorable. In the end, the implementation has been limited to those typologies who have available ROS packages, which will be specified afterwards in the next section.

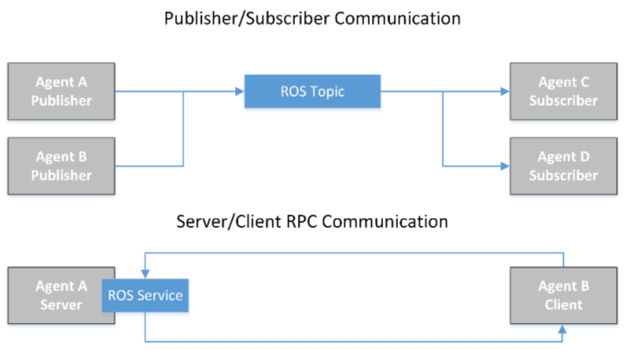

Fig. 1. ROS communication schemas

ROS is based in a client-server architecture, in which the executable files work as nodes that subscribe to certain topics, and publish the output of their operation (see Fig. 1). In the general case, the fiducial marker packages (which contains the previously called executables) require two topics: the camera feedback (type of message: sensor_msgs/Image, published in topic

/<camera_name>/image_raw) and its intrinsic parameters (type of message: sensor_msgs/CameraInfo, published in topic /<camera_name>/camera_info). These nodes are usually included in launch files, which enable the user to run more than one executable at a time. Since custom message types can be defined in ROS, the output of these nodes can vary for different fiducial marker implementations. However, they all share in common a well-known package called tf, which is tasked with publishing the transformation matrices between two given reference frames. The pose of the markers (position + orientation) can be obtained from that topic, and for that reason the performance evaluation tests defined will be based on this information.

III. ROS Implmentation of Fiducial Markers

There are multiple ROS-available implementations of fiducial marker technologies. Figure 2 shows examples of several fiducial marker types that will be studied in this work.

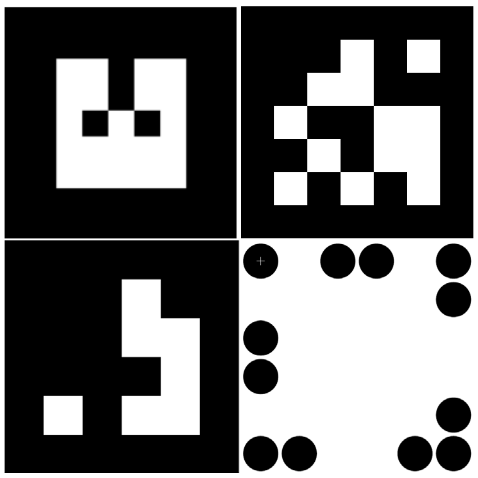
Fig. 2. Sample fiduciary markers. **Top left**: ARTag. **Top right**: ArUco. **Bottom left**: AprilTags. **Bottom right**: PiTag.

A. *ARTag:*

The ar_track_alvar package [5] provides an implementation of the ARTag marker detector and tracker. This package enables the possibility of using cameras with depth information (e.g. the Kinect) to add finesse to the pose estimation, and to set marker bundles to be detected as a single entity.

The outer border of these markers is square-shaped, to provide perspective support and minimal pose jitter by using the corners. Inside, a grid of 6x6 squares represents a 36-bit codification of the marker (black = ‘0’, white = ‘1’), where the first 10 bits are the marker’s ID and the 26 remaining account for error detection and correction, and uniqueness of only four possible marker orientations. The algorithm follows the next steps:

1. Edge detection of the square border (for increased immunity against unknown lighting and photometric calibration, as well as partial occlusion, when compared to thresholding approaches).
2. Finding quads (quadrilateral contours) for distinguishing between the cells of the whole marker.
3. Implicit decoding process using the bitonal grid cells (they can only be black or white, ‘0’ or ‘1’, to avoid thresholding issues).
4. Verification of the marker via different methods, using several bits of the AR Tag reserved for this purpose.

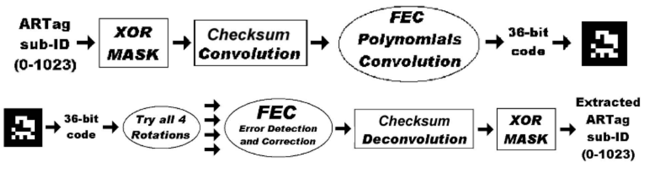

Fig. 3. ARTag block diagram.

B. *ArUco*

A ROS implementation for the ArUco marker recognition algorithm can be found within package fiducials [6]. The process takes 4 steps:

1. Use of the adaptive threshold technique to binarize the image while enhancing the edges, making it easier to identify them using an edge detector.
2. Out of the detected edges, only the fiducial markers are picked.
3. The ID and data contained within the marker core are extracted using perspective transformations.
4. Finally, the corners position is estimated as the intersection of edges. With that information and the size of the marker itself, the pose estimation can be computed making use of OpenCV libraries and assuming the pinhole camera model (as presented previously).

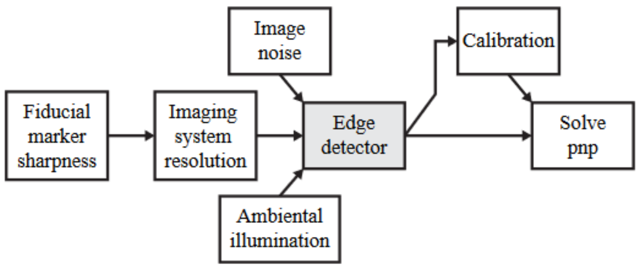

Fig. 4. Aruco block diagram, courtesy of [2].

As in the ar_track_alvar package, fiducials includes a library with valid 7x7 ARUco markers (varieties of other sizes exist too, can be set in the dictionary argument of the node). Online libraries of ARUco markers can be found too.

C. *PiTag and Fast PiTag*

PiTag and Fast PiTag implementations can be found in the cob_fiducials [7] package. Itincludes printable documents for both versions of the PiTag markers models, and a third-party repository that implements creation of custom markers of specific sizes and cross-ratios, called PiTag-generator. The general procedure for both algorithms follow a number of steps:

1. Binarize with an adaptive threshold (increases robustness to illumination changes) and apply an ellipse detector provided by the OpenCV library.
2. Iterate over all pairs of dots, and if two other dots can be found within a fixed distance of the line connecting the two first corner candidates, they will likely be corner points (exploiting straight line invariance or collinearity) (Figure 8, a)).
3. Repeat the process to validate if the candidate found in the previous step was indeed a corner, verifying that the cross-ratio between the dots is 1 or δ (depending on which side of the square). The labeling happens by observing the ordering of the sides, which is always conserved (Figure 8, b)).
4. The fourth corner is sought following the same line-based technique (Figure 8, c)).
5. Finally, test if the points detected and labeled belong to an expected marker by comparing the average cross-ratio between the four sides and comparing it to database values.
6. If the pose estimation of the camera is also desired, the solvePnP function from the OpenCV library is used. However, it is recommended to refine the ellipse centers detection to recover a more accurate pose.

D. ApriTtags

AprilTags can be found under apriltags_ros [8] package. The type, size and ID's of the tags to be found can be configured through yaml files. Other options are configurable through the launcher file at ROS configuration level. A custom tag generator is also available.

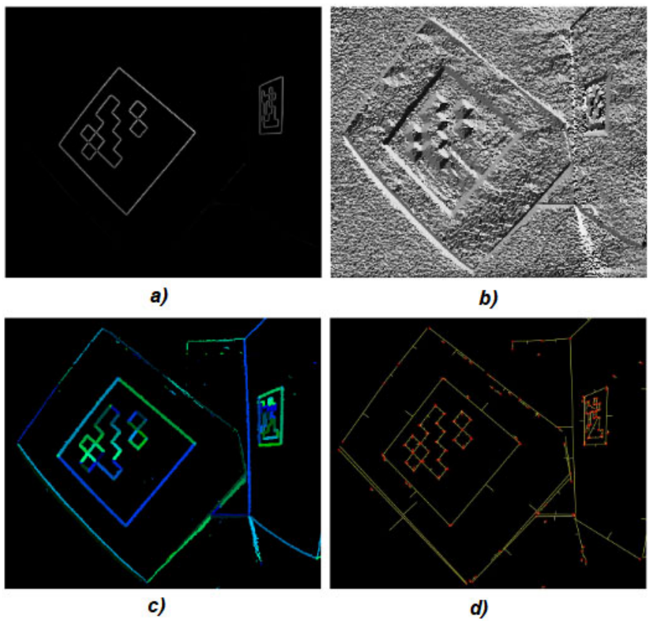

Fig. 5. Sample images for the AprilTag detection procedure.

The detector is divided into two major parts: the tag detection and the coding system. The first one is a graph-based image segmentation algorithm that detects lines by computing the gradient magnitude (Fig. 5,a) and direction (Fig. 5,b) of every pixel and clustering together those with similar characteristics (Fig. 5,c), to finally fit a line segment across clustered pixels using a least-squares weighing method (Fig. 5,d). This process is sensitive to noise in the image, which is the reason why a low-pass filter is used; however, as edges are intrinsically large features, this filtering will not cause appreciable amounts of information loss.

All segments forming a 4-sided shape become "quad" candidates; the threshold applied for identification handles robustness to occlusions and segmentations errors, and in the end leads to a low false negative rate at the cost of a high false positive rate. At this point, it is possible to project the 2D points in the tag coordinate system to the 2D image plane using the Direct Linear Transform (DLT). Position and orientation of the tag can also be determined if the camera's focal length and physical size of the marker are known.

Finally, the payload encoded in the marker can be read bit by bit by computing the tag-relative coordinates of each bit field, applying the DLT to work in the image plane and thresholding the resulting pixels using a spatially-variant intensity level model (to increase robustness to lighting changes). As per the coding part of the algorithm, a 36-bit modified lexicode is implemented, which has the ability to detect and correct an amount of errors equal to the minimum Hamming distance between any two codewords. Because of the necessity of rotation invariance, excessively simple codewords are discarded; the lexicode algorithm can also be expanded to add new constraints.

IV. Benchmark description

The set of tests to be performed to assess the robustness of each of the fiducial marker technologies implemented are based in [1], [2]. The camera used to record the image feed is a Logitech® that publishes images in VGA resolution (640x480 pixels) at a 30 fps rate using a ROS package driver called usb_cam, through the /usb_cam/image_raw topic.

To obtain accurate measurements for the tests, the camera has been calibrated previously with the aid of the camera_calibration package, using a checkerboard of known size (9x6) and known tile size (2,05 cm).

V. Experimental Results

In order to evaluate the different fiducial marker detectors, a ROS node was developed. This node subscribes to the /tf and /image_raw topics to compute the maximum and minimum distances and orientations of the markers with respect to the camera at which they were recognized, as well as the detection rate of the markers during the test and the processing time of the algorithms.

TABLE I. Experimental results

Metric [units]	Fiducial Marker				
	AR Tag	ARUco	Pi Tag	Fast PiTag	April Tags
Marker size [cm x cm]	14 x 14	14 x 14	10 x 10	10 x 10	10,4 x 10,4
Detection rate (lighting) [%]	10,81	68,36	63,5	0	95,51
Detection rate (occlusion) [%]	4,93	26,53	27,4	0	28,48
Max. occlusion [%]	10	15	20	0	15
Min. detection distance [m]	0,23	0,495	0,322	0,516	0,163
Max. detection distance [m]	4,314	4,240	1,708	1,150	4,110
Min. tilt angle [°]	-68,2	-67,1	-58,4	-60,9	-68,8
Max. tilt angle[°]	68,8	69,7	70,6	51,1	73,2
False positive rate	Low	Very low	Very low	Very low	High
False negative rate	Low	Very low	Low	High	Low
Intermarker confusion	Medium	Low	Low	Low	High
Rate (average) [Hz | ms]	1 Hz; 1 s	4,5 Hz; 222 ms	11Hz 90ms	5 Hz 200 ms	8,2 Hz 122 ms
Marker library size (# ID's)	1024	~4000	85 + custom	6 given	400+ custom
Perspective support	Yes	Yes	Yes	Yes	Yes
Immunity to photometric calibration	Yes	Yes	Yes	Yes	Yes

To allow for repetitibility and any additional validation required, the sequences were captured through the *rosbag* tool and then processed. The results obtained for each of the fiducials technologies studied can be found summarized in Table 1.

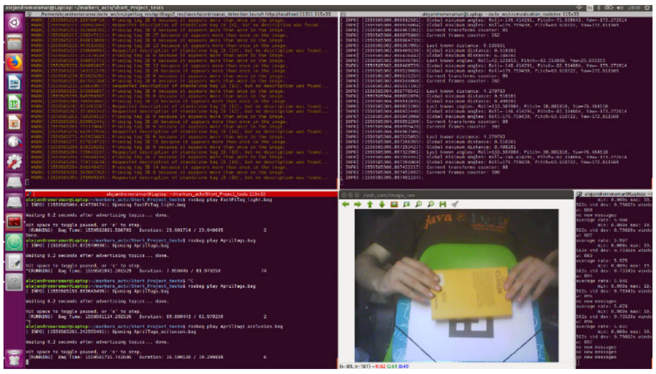

Fig. 6. Detail of the ROS-Ubuntu system setup used to run the experiments, showing a segment of the occlusion experiment. The raw detections of each marker detector running are shown in the top left screen, the custom built evaluation node on the top-right screen, and the rosbag command line to reproduce the camera data in the bottom-left screen.

VI. Discussion & Conclusions

Regarding ARTag, the tests denote a quite good performance: it boasts the best range for allowed orientations of the marker, both minimum and maximum angles, and the best distance operating range (greatest difference between maximum and minimum distance detections). It is, however, the slowest in terms of speed performance (marker detections at 1 Hz), which can be an instant disqualifier in some applications (for example, in aerial robotics) where high-speed performance is a must. Adding this fact to average lighting and occlusion resistance, the conclusions drawn are that ARTag is good for long-range applications, but limited to those where the engineers are in control of the environment.

Regarding ARUco, its performance is very close to ARTag in terms of distance and orientation ranges allowed. However, it is markedly more robust against unfavorable lighting and occlusions, partly due to its higher speed performance (an average of 4,5 Hz | 222 ms, more than 4 times faster than ARTag). Together with very low false positive and negative detections rates, the conclusion is that ARUco could be used in real-time and potentially be the best candidate for medium-long range, outdoor applications. Its library size also makes it attractive for projects of a bigger scope.

Regarding PiTag, its distance range is one of the shortest out of the complete dataset (bear in mind that the marker itself was smaller in comparison with the rest, too). In terms of orientation, though, it performs well, at the same level of ARTag and ARUco. It is the fastest out of all the processing algorithms and presents low rates of false positives and negative, which makes it the best candidate for high-speed, short-medium range applications. As ARUco, it can be scaled to higher dimension projects, in this case due to the fact that custom markers can be generated.

Regarding Fast PiTag, even though its image processing algorithm is fast, the reduced detection rate of markers results in overall worse speed performance in average, as well as in terms of distance and orientation ranges. The 0% detection rate in both the occlusion and light immunity tests point out its poor robustness against unfavorable conditions, which renders its usefulness to medium range, indoor applications.

Finally, regarding AprilTags, it rivals with ARTag for the best detection range in both distance and orientation terms. Even if it automatically discards duplicated ID's in the image, it presents a worrying amount of false positives, making it a necessity to filter out the transforms received (by knowing the ID of the expected markers to be recognized, for example). Since it is the second fastest algorithm out of all the ones considered, and it includes a broad dictionary of markers (around 400 different ID's, distributed among multiple families + the possibility to create new, custom ones), it would be worth to implement an ID-filter for the detections, since it is the best balanced option in terms of performance speed and detection range (between ARUco and PiTag).

Acknowledgement

This research was funded by the Ministry of Economy, Industry and Competitiveness through project DPI2016-78957-R.

VII. References

[1] [1] A. Sagitov, K. Shabalina, R. Lavrenov, and E. Magid, ‘Comparing fiducial marker systems in the presence of occlusion’, in *2017 International Conference on Mechanical, System and Control Engineering (ICMSC)*, 2017, pp. 377–382.

[2] [2] D. C. Popescu, M. O. Cernaianu, P. Ghenuche, and I. Dumitrache, ‘An assessment on the accuracy of high precision 3D positioning using planar fiducial markers’, in *2017 21st International Conference on System Theory, Control and Computing (ICSTCC)*, 2017, pp. 471–476.

[3] [3] M. Fiala, ‘Designing Highly Reliable Fiducial Markers’, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 32, no. 7, pp. 1317–1324, Jul. 2010.

[4] [4] M. Quigley *et al.*, ‘ROS: an open-source Robot Operating System’, in *ICRA workshop on open source software*, 2009, vol. 3, p. 5.

[5] [5] ‘ar_track_alvar - ROS Wiki’. [Online]. Available: www.wiki.ros.org/ar_track_alvar. [Accessed: 11-Jun-2019].

[6] [6] ‘fiducials - ROS Wiki’. [Online]. Available: www.wiki.ros.org/fiducials. [Accessed: 11-Jun-2019].

[7] [7] ‘cob_fiducials - ROS Wiki’. [Online]. Available: www.wiki.ros.org/cob_fiducials. [Accessed: 11-Jun-2019].

[8] [8] ‘apriltag_ros - ROS Wiki’. [Online]. Available: www.wiki.ros.org/apriltag_ros. [Accessed: 11-Jun-2019].