Detection of Aruco markers using the quadrilateral sum conjuncture.

José Ferrão[1], António J. R. Neves[1,2], Paulo Dias[1,2]

DETI[1] and IEETA[2]
University of Aveiro
3810-193, Aveiro, Portugal

Abstract. Fiducial Markers are heavily used for pose estimation in many applications from robotics to augmented reality. In this paper we propose an algorithm for the detection of aruco marker at larger distance. The proposed algorithm uses the quadrilateral sum conjecture and analyzes the sum of the cosine of the internal angles to detect squares at larger distances. Experimental results showed that the proposed solution is able to improve the detection distance when compared to other methods that use similar marker format while keeping equivalent pose estimation precision.

Keywords: Robotics, computer vision, markers detection, aruco.

1 Introduction

Absolute pose estimation is a common problem in robotics. One possible solution to tackle this problem is the use of odometry information (coming from encoders and Inertial Units). However, this solution suffers significantly from drift error. An alternative commonly used not only in robotics but also in areas like augmented reality is the usage of markers to estimate an absolute position of the camera. A visual marker is something that can be easily distinguished from the rest of the ambient and have characteristics that allow it to be easily detected and identified.

The motivation for this work was the need for a marker-based solution to easily identify charging stations for robots and correcting odometry drifting using the pose estimated by the marker detector. However, one of the main problems found when experimenting already existing solutions (ROS Aruco [1] and Alvar [2]) was that they are not able to identify the markers from far away or under hard perspective distortion making it hard to discover the charging station.

In this paper we propose an algorithm for the detection of aruco markers. Our approach uses the quadrilateral sum conjecture to improve the detection distance of these markers.

This document is divided into 7 sections, being the first one this introduction. The second section presents the base concepts related with marker encoding and detection. Section 3 explains the proposed algorithm while Section 4 discusses pose estimation. Section 5 presents experimental results and Section 6 contains result analysis and conclusion.

2 Aruco Markers

Aruco markers are geometrically square, they have a black border and an inner grid that is used to store a numeric identifier in binary code.

To identify the marker, a dictionary is used [3]. The dictionary defines a set of rules used to calculate the marker identifier, perform validation and apply error correction.

We use the original aruco dictionary [1], that uses bits from the marker 2nd and 4th columns to store the marker identifier in natural binary code. The remaining bits are used for parity checking. Figure 1 represents the first four markers in this dictionary.

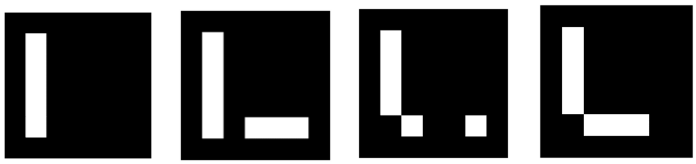

Fig. 1. Aruco markers with ID 0, 1, 2 and 3.

A signature matrix is used to validate the marker. Each row of this matrix encodes a possibility of 2 bits. An aruco marker is valid only if each row is equal to one of the rows of the signature matrix. This forces the marker to have only one valid rotation. Table 1 represents the signature matrix used in this dictionary.

Value	Data				
0	1	**0**	0	**0**	0
1	1	**0**	1	**1**	1
2	0	**1**	0	**0**	1
3	0	**1**	1	**1**	0

Table 1. Signature matrix, used to validate aruco markers.

By analyzing the signature matrix, it is possibly to verify that it is not enough to guarantee that there is only one possible rotation for each marker. In the Figure 2, we can see the marker 1023 that is horizontally symmetric.

Fig. 2. Aruco marker with ID 1023.

3 Detection algorithm

The detection algorithm was implemented using the OpenCV library, since it provides a large set of image processing algorithms. Figure 3 shows the steps applied to detect and identify markers.

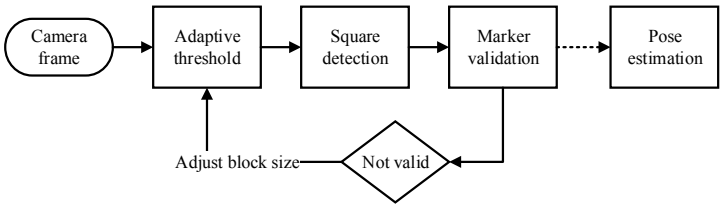

Fig. 3. Diagram of the proposed algorithm.

The algorithm starts by applying adaptive threshold [4] to transform the image into a binary map. This algorithm consists in calculating, for each pixel, a threshold value using the histogram of its neighborhood. It is usually indicated for situations where it is possible to observe multiple lighting conditions. Figure 4 represents the results obtained from adaptive thresholding.

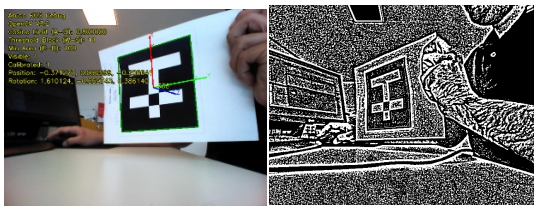

Fig. 4. Adaptive threshold result.

To determine the best threshold for the block size (neighborhood size), several block sizes were tested on each frame, being the threshold chosen the one that maximizes the number of markers found. The obtained parameter is tested also when there are no markers visible in order to avoid false positives. In Figure 5 it is possible to observe two cases where different block sizes were chosen allowing the algorithm to adapt and detect marker with different size.

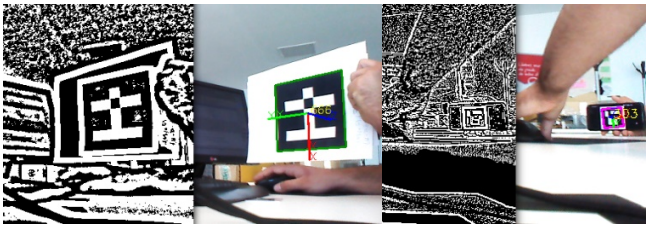

Fig. 5. Comparison of different threshold block sizes.

After having a binary image, we perform square detection. The algorithm starts by obtaining the contours using a border-following algorithm [5]. After obtaining the contours, Douglas-Peucker algorithm [6] is used for contour simplification.

Using the previously detected contours, we use the Quadrilateral Sum Conjecture as a criterion to detect squares. The Quadrilateral Sum Conjecture tells us the sum of the angles in any convex quadrilateral is 360 degrees (Figure 6).

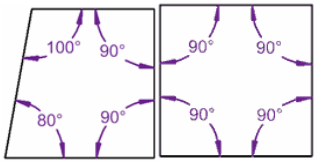

Fig. 6. Internal angles of convex quadrilaterals.

Even under perspective distortion a square is always a convex quadrilateral. This will be our first criteria. It is also possible to verify that the sum of the cosine of all inner angles equals in a convex quadrilateral is always close to zero. Considering this as our second criteria, we consider as valid square if the sum of inner angle between all corners is under a defined threshold (close to zero).

To filter eventual noise on the image, a third criteria was added, considering that all contours composing a geometric object with an area bellow a defined threshold will be discarded.

These three criteria allow to properly filter squares even under heavy distortion from the contour list. Figure 7 represents the obtained result for a maximum sum of cosine of 0.25 and a minimum area of 100px.

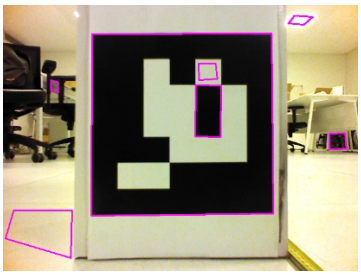

Fig. 7. Result of square detection algorithm.

Perspective distortion is removed from the detected squares, then they are set to a 7x7 matrix using linear interpolation and threshold is applied using the Otsu's Binarization algorithm [7]. At this point, we obtain a matrix with the marker data on it. Figure 8 represents the matrix obtained after the binarization process.

Fig. 8. Marker reading result.

The marker data is now validated using the signature matrix. If the data is not valid that means that it does not correspond to an aruco marker. Markers might be detected in any orientation. The algorithm tests the data with different rotations (90º, 180º, 270º) and, if the marker is not recognized for any rotation, it is then discarded.

4 Pose estimation

For pose estimation we use an iterative mode using Levenberg-Marquardt [8] optimization, implemented on the method `solvePnp` from OpenCV.

To obtain the camera position regarding to the markers, they need to be registered into the program. A marker is represented by its identifier and a real-world pose (position and rotation).

Corners obtained from all visible known markers are used to estimate the camera pose. In Figure 9 we can observe the origin referential and camera pose estimated using the corners detected from two visible makers.

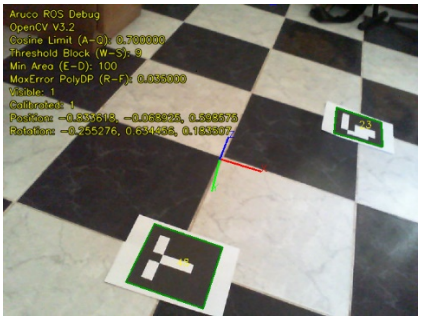

Fig. 9. Marker detection result

5 Algorithm evaluation

We created a laboratory setup to compare the proposed solution with the most used existing solutions. Two test markers were printed, one Aruco maker and one ARTag maker with an exactly size of 20cm. The camera was placed on top of a box and the marker was aligned with the camera, as best as possible. The marker was then held on a box using transparent tape.

A measuring tape with a millimeter scale was used to measure the distance between the box and the markers. An image was taken for each distance tested and the markers were moved 30cm each time until none of the algorithms was able to detect the marker. Figure 10 represents some samples of the testing data used.

Fig. 10. Samples of the testing data.

To measure the tolerance of the detector to perspective distortion, a second laboratory experiment was created. A marker was placed on a box and the camera was positioned 2.0 meters away from the marker. The marker was rotated around itself in steps of 10º from 0º to 80º. Camera calibration was performed using the traditional chessboard pattern and the calibration matrixes obtained were stored and used for the tests.

Table 2 presents the results obtained for marker detection. In this table, the calculated distance between the camera and the marker is presented. It is also presented the error between the real distance and the obtained by the algorithms. It is possible to observe that the proposed solution obtained a considerable longer detection distance when comparing the other algorithms.

Distance (m)	Proposed solution		ROS Aruco		Alvar	
	Distance (m)	Error (%)	Distance (m)	Error (%)	Distance (m)	Error (%)
0,3	0.309	3.004	0.315	4.79	0.303	0.838
0,5	0.501	0.136	0.505	1.013	0.509	1.832
0,7	0.699	0.101	0.705	0.663	0.707	0.974
0,9	0.895	0.61	0.904	0.486	0.908	0.862
1,1	1.094	0.544	1.111	0.976	1.11	0.94
1,3	1.287	0.974	1.312	0.944	1.307	0.524
1,5	1.479	1.425	1.515	0.973	1.505	0.352
1,7	1.679	1.264	1.707	0.414	1.708	0.472
1,9	1.864	1.91	1.926	1.348	1.913	0.655
2,1	2.054	2.229	2.109	0.433	2.107	0.334
2,3	2.283	0.757	2.336	1.544	2.332	1.355
2,5	2.467	1.33	2.539	1.545	2.514	0.548
2,7	2.679	0.787	2.786	3.073	2.726	0.944
2,9	2.84	2.097	2.95	1.682	2.9	0.016
3,1	3.032	2.229	3.142	1.349	3.16	1.907
3,3	3.237	1.961	3.343	1.299	3.358	1.713
3,5	3.459	1.181	3.581	2.263	3.534	0.963
3,7	3.647	1.446	3.791	2.413	3.741	1.104
3,9	3.859	1.058	4.013	2.82	4.014	2.849
4,1	4.091	0.218			4.212	2.662
4,3	4.249	1.19			4.393	2.108
4,5	4.452	1.077			4.595	2.059

4,7	4.643	1.236			5.017	6.322
4,9	4.808	1.923			5.054	3.04
5,1	5.116	0.31				
5,3	5.154	2.836				
5,5	5.566	1.18				
6	5.712	5.049				
6,6	6.304	4.703				
7,2	6.922	4.021				
7,8	7.695	1.367				
8,4	8.618	2.534				
9	8.639	4.181				

Table 2. Results obtained for marker detection.

Table 3 represents the accuracy of each method considering the medium accuracy of all measures inside of the detection range. It is possible to observe that the proposed algorithm has similar precision when comparing to the other algorithms tested.

	Average error (%)		
	Proposed solution	ROS Aruco	Alvar
0-390cm	1,2	1,7	1,1
0-490cm	1,3		1,5
0-900cm	1,8		

Table 3. Accuracy comparison between marker detectors.

Table 4 presents the calculated distances obtained after the marker rotation. We can observe that the proposed method performed better than the other two algorithms used for comparison, obtaining lower errors.

	Proposed solution		ROS Aruco		Alvar	
Rotation	Dist. (m)	Error (%)	Dist. (m)	Error (%)	Dist. (m)	Error (%)
0	1.981	0.980	2.013	0.631	1.986	0.727
10	1.983	0.872	2.032	1.583	2.017	0.858
20	1.985	0.741	2.031	1.537	2.011	0.564
30	1.989	0.528	2.029	1.415	1.974	1.308
40	2.001	0.040	2.029	1.425	1.971	1.493
50	2.002	0.121	2.03	1.468	2.028	1.357
60	2.004	0.213	2.033	1.639	1.974	1.322
70	1.993	0.341	2.029	1.421		

Table 4. Results obtained for maker rotation.

6 Conclusion

Both the proposed algorithm and the two algorithms used for comparison use a similar approach for marker detection. All of them apply adaptive threshold, detect squares, refine corners, apply distortion correction, and decode the marker data. The method proposed in this document used a different approach for square detection that allowed the detection of markers at longer distances.

We compared our algorithm with two state of art algorithms (ROS Aruco and Alvar) and verified that our algorithm detects the marker up to 9m, when compared with 4m and 5m for the other algorithms respectively. It represents a 44% increase in detection distance, with similar precision values.

Our method also presents more tolerance to perspective distortion, obtaining better precision results when detecting rotated markers.

To further improve the algorithm, a corner refinement method with subpixel accuracy could be added improving the marker corner position estimation.

An implementation of the algorithm described in this document can be found at `www.github.com/tentone/aruco`.

7 References

1. S.Garrido-Jurado, R.Muñoz-Salinas, F.J.Madrid-Cuevas, M.J.Marín-Jiménez, Automatic generation and detection of highly reliable fiducial markers under occlusion. Pattern Recognition Volume 47 Issue 6 2280–2292, June 2014.
2. Sanni Siltanen, Theory and applications of marker-based augmented reality. VTT Science 3.
3. S.Garrido-Jurado, R.Muñoz-Salinas, F.J.Madrid-Cuevas, R.Medina-Carnicer, Generation of fiducial marker dictionaries using Mixed Integer Linear Programming. Pattern Recognition 51 Issue C 481-491, March 2016
4. Mehmet Sezgin, Bulent Sankur, Survey over image thresholding techniques and quantitative performance evaluation. Journal of Electronic Imaging 13(1), 146–165, January 2004.
5. Satoshi Suzuki, Keiichi Abe, Topological structural analysis of digitized binary images by border following. Computer Vision. Graphics, and Image Processing, 32-46, Vol. 30, Issue 1, September 1985.
6. Shin-Ting Wu, Adler C. G. da Silva, Mercedes R. G. Márquez, The Douglas-peucker algorithm: sufficiency conditions for non-self-intersections. J. Braz. Comp. Soc. Vol.9, Issue 3, Campinas, April 2004.
7. Nobuyuki Otsu, A Threshold Selection Method from Gray-Level Histograms. IEEE Transactions on Systems, Man, and Cybernetics, 62-66, Vol. 9, Issue 1, January 1979.
8. Kenneth Levenberg, A method for the solution of certain non-linear problems in least squares, Quarterly of Applied Mathematics, 164-168, Vol.2, Issue 2, July 1994