Color-Based and Recursive Fiducial Marker for Augmented Reality

Douglas Tybusch, Gilseone Rosa de Moraes, Osmar M. dos Santos, Andrei Piccinini Legg, Simone Regina Ceolin
Postgraduate Program in Computer Science (PPGI)
Federal University of Santa Maria (UFSM)
Email: {tybusch.douglas, gilseonemoraes, andrei.legg, simoneceolin}@gmail.com, osmar@inf.ufsm.br

***Abstract*—The popularity of applications using Augmented Reality, especially due to the dissemination of smartphones with high processing power, introduces the need for Fiducial Markers that can be detected quickly, with good accuracy and can deal with partial occlusion. Fiducial Markers can have different shapes, sizes, structure and colors, and are inserted into a scene to facilitate the detection and consequent projection of virtual objects. In particular, this paper proposes a new Color-based and Recursive Fiducial Marker (CRFM), which is constructed by square-based geometric forms and uses different colors to produce a recursive structure that was designed to still work under partial occlusion. We describe the CRFM design and how its detection mechanism works. Our evaluation results show that CRFM achieves a good level of accuracy. Moreover, we show that the detection of the CRFM can be as fast as a board of ArUco, where only black and white colors are used.**

I. Introduction

Augmented Reality (AR) is a technological concept [1] that has been around for quite some time, but has recently been giving a lot of attention. In particular, due to the increase of processing power of smartphones and the dissemination of applications for them [2].

One of the key research topics in Augmented Reality include techniques for detecting elements in a given image. Detection enables the projection of virtual objects over the original image. In this work, we focus on the use of Fiducial Markers [3], synthetic elements that are inserted as references for facilitating the projection of virtual objects [4] [5] [6] [7].

Usually, markers are composed of geometric forms, which enables a fast and accurate detection over the image. For instance, square markers provide the possibility of calculating the camera pose from the location of one of its 4 vertices [8]. While using circular markers, the camera pose can be calculated considering the marker contour [9]. In this work, we focus on the use of square-based geometric forms.

More specifically, we propose the use of a Color-based Recursive Fiducial Marker (CRFM). The use of color is key to the definition of a recursive structure in the CRFM. With a recursive marker, we are able to deal with partial occlusion that may occur over the marker. Moreover, we can deal with different detection sizes for the marker, depending on the viewing distance to the marker, we can focus on the detection of different levels of the recursive marker. This provide an optimized detection over traditional (fixed size) markers.

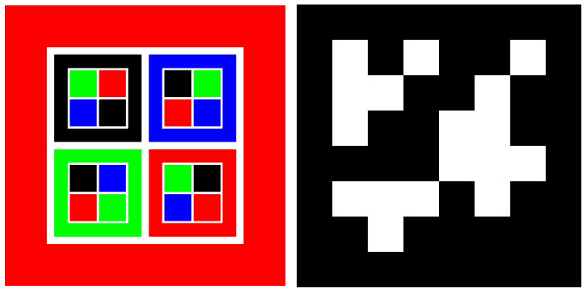


Fig. 1. The proposed marker (CRFM) and a binary marker (ArUco [10]).

This paper is organized as follows. Section II describes related work. In Section III we provide a description of the CRFM proposed in this paper. Section IV shows how to detect the CRFM in an image. Both detection accuracy and performance evaluations are presented in Section V. Finally, Section VI presents final remarks and future work.

II. Related Work

Figure 2 presents Fiducial Markers that are similar to the work described in this paper, differing mainly by not using colors in its structures.

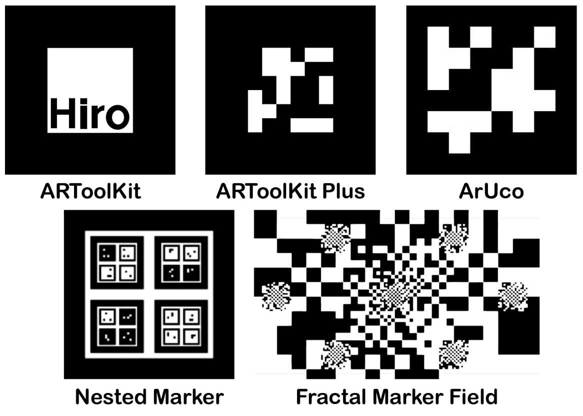


Fig. 2. Fiducial Markers similar to our work.

Many proposed fiducial markers systems consist of using techniques in order to segment and identify patterns encoded within the marker area. ArUco [10] uses a binary identification algorithm. ARToolKit [4] uses image correlation for marker detection, while the ARToolKit Plus [11] also focuses on binary detection.

Nested Marker [12] proposes a structure organized into recursive layers, where each layer allows identification at different distances of the camera, while also allowing identification under a partially occluded marker.

Fractal Marker Field (FMF) [13] is a fractal structure marker system, which aims to solve limitations of markers regarding their scale. This marker enables a larger area for camera detection due to the possible size of its nested markers. It is based on the structure proposed by the Nested Marker [12].

The work of ArUco [10] does not provide a recursive structure, but provides a robust detection method, which addresses partial occlusion by creating different types of boards (each board containing a set of binary markers). Although ArUco does not use color in its markers, we use part of its technique for contour detection and filtering in our work.

There are several studies that propose markers of different types and formats, but there are no studies that use colors, specifically, into the implementation of its marker structure. Also, hierarchical or recursive markers do not appear to be a highly explored topic.

The marker structure and detection algorithm proposed in this work aims to explore the use and detection of colors, as well as recursive structures.

III. Color-based and Recursive Fiducial Marker

In the Color-based and Recursive Fiducial Marker (CRFM) structure, each marker (and its sub-markers) is composed of a main block border color and four internal colored blocks that, together, make up a unique identifier (ID) for a given marker. This relationship between the main block and its children blocks is also called a hierarchy. Each one of the internal blocks of a given marker can also contain another hierarchy, giving a recursive design to the marker. The relationship between the colors that compose a hierarchy are never repeated – each hierarchy maintain its own unique ID. On any given hierarchy, the lower and rightmost block, must have the same color of its main block, indicating its position and orientation. This can be seen in Figure 3 (I) and (II).

A CRFM is generated following a proportionality pattern (Figure 3 (II) and (III)) between the border thickness of the main block, its children and a blank space, which is added to generate extra contrast between blocks (improve the rectangle detection). The thickness of a child block is 45% the thickness of its parent block and the blank space is 20% of the child block thickness. Following this proportion, for every generated hierarchy, we are able to create an area of interest that allows the identification the color of each block in a given hierarchy (Section IV). These proportions were determined using an empiric method, based on an initial marker model, and adapted according to testing results.

CRFM is composed of a recursive structure, having 2 layers (of markers) and 3 levels (of blocks), defined by a hierarchy that has one marker color at the top layer and four markers on

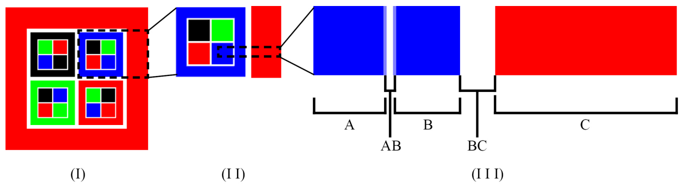


Fig. 3. The [AB] region is equivalent to 20% of region [A] and 10% of region [B]. The thickness of each sub-marker border [B] is 45% the thickness of its parent marker.

its inferior layer (Figure 4). Currently, four colors are used to generate a marker: black, blue, green and red. Considering that each hierarchy is composed of a block and its four children, it is possible to define a marker combining these colors.

In order to find a given target on a captured image, we must first generate its corresponding N-ary tree as depicted in Figure 4. Then, when the detection algorithm is executed, we cross data from the target N-ary tree with the detected markers from the image. The main block color and the sequence of its children colors are not repeated into other hierarchies of target markers.

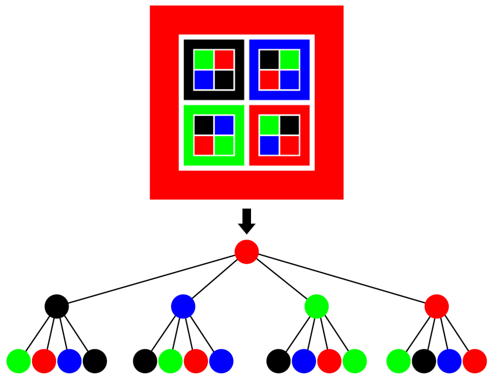


Fig. 4. Corresponding N-ary tree for the CRFM.

Therefore, due to the existence of the N-ary tree, we do not need to find all target markers in an image. It is possible to estimate a pose with only one target (marker or sub-marker), thus, allowing the CRFM to handle partial occlusion situations. This is exemplified in Figure 5, where the detection of only one marker is enough to determine the position in the hierarchy.

The recursive structure also allows the marker to be detected from greater distances (Figure 6). If a marker is observed from a greater distance, it will be able to be detected its outer-most marker (Figure 6 (b)) but, probably, not its inner-most markers. Using the N-ary tree, it is able to map the marker according to its hierarchy. The same idea applies when the marker is observed from a shorter distance but, instead, for the inner-most markers (Figure 6 (c)).

IV. Detection of the CRFM

The flowchart for the detection algorithm used in the CRFM is presented in Figure 8. We can identify three main steps executed by the detection algorithm: (a) candidate detection;

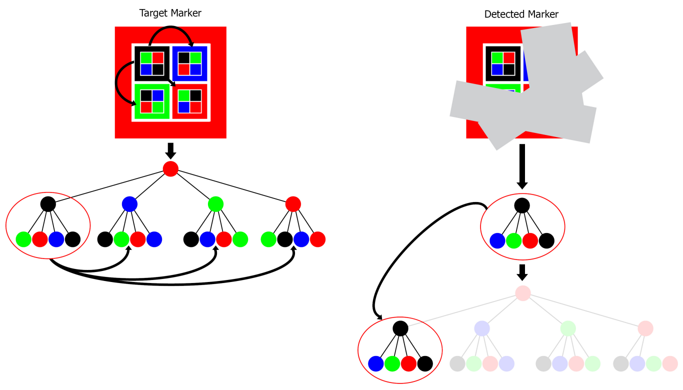


Fig. 5. Target marker (left) and detected marker considering partial occlusion (right).

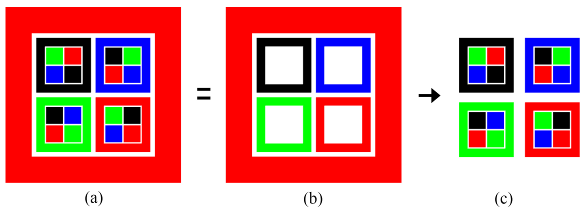


Fig. 6. CRFM with a two-layer hierarchy (a). Possible hierarchy detection from greater (b) and shorter (c) distances to the marker.

(b) color filtering; and, (c) identification. The algorithm begins its process by detecting and filtering candidate contours. After the candidate detection, each candidate is inserted and organized in an N-ary tree, where its corresponding color is analyzed and filtered accordingly. In the last step occurs the identification of the marker. Where a search over the N-ary tree occurs, looking for potential matches from a previously determined target marker tree. These steps are discussed in more details.

A. Candidate Detection

The candidate detection algorithm consists of a slightly modified version of the ArUco detection algorithm, which can be abridged into the following steps:

- Binary-Threshold: Consists on applying a local adaptive thresholding technique [10], which iterates over the gray-scale converted image being analyzed. The number of iterations affects both the performance and reliability for the candidate detection.
- Finding Contours: The Suzuki and Abe contours detection algorithm [14] is used on each thresholded image, generating a number of contours to be further analyzed.
- Filtering Square Contours: After combining detected contours from its multiple thresholds, the contours are then applied to the Douglas–Peucker algorithm [15], in order to reduce the number of vertices. To obtain only rectangular and square shaped contours, it is kept only contours containing four vertices.
- Filtering Close Contours: In order to remove possibly duplicated contours, contours too close to each other are removed, keeping the outer-most one only.
- Generate the N-ary Tree: The remaining contours are then added to an N-ary tree that, during insertion, are organized into a parent-child/sibling hierarchy, according to the rectangles (obtained from the contours) position and size.

B. Color Filtering

After the candidate contours are filtered, only rectangle-shaped contours remain. For every rectangle-shaped contour, we need to determine it corresponding color. In order to determine the color of the rectangle, we analyze an area of pixels inside the rectangle. These steps are described in more details.

1) Generation of Sampling Areas: In order to reduce the processing required by the algorithm, we define a reduced area of the rectangle to be analyzed. The generation of the sampling areas are calculated based on the four vertices of the detected rectangle (Figure 7).

In order to define the sampling areas, we start by delimiting the centroid (PC) of the rectangle. We draw lines originating from each of its vertices (Px), with its intersection point being the centroid (PC). For each one of these lines, we perform two operations. First, we find the central point (P0C1) between the vertex and the centroid (PC). Second, we repeat the previous calculation, now using the previous central point (P0C1) and the starting vertex (P0), resulting the (P0C2) point. We use the calculated Px and PxC2 points in order to generate our resulting samplings areas.

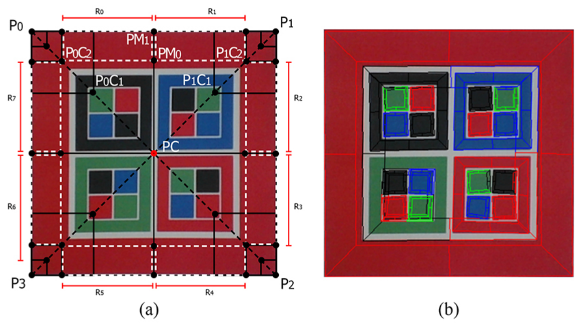


Fig. 7. Definition of sampling areas (a), and the resulting sampling areas (b).

After the sampling area is determined, we convert it into 8 corresponding rectangles. By increasing the number of rectangles, and determining each of their colors separately, we are able to reduce the number of errors in the color detection. These errors are usually caused by light changes, reflexes and shadows over the marker.

2) Raster Sampling Areas: Since the rectangles can be positioned over any orientation in the image, we must raster the selected areas. We implemented the raster algorithm using Bresenham's line algorithm [16] in order to iterate over the sampling areas more efficiently.

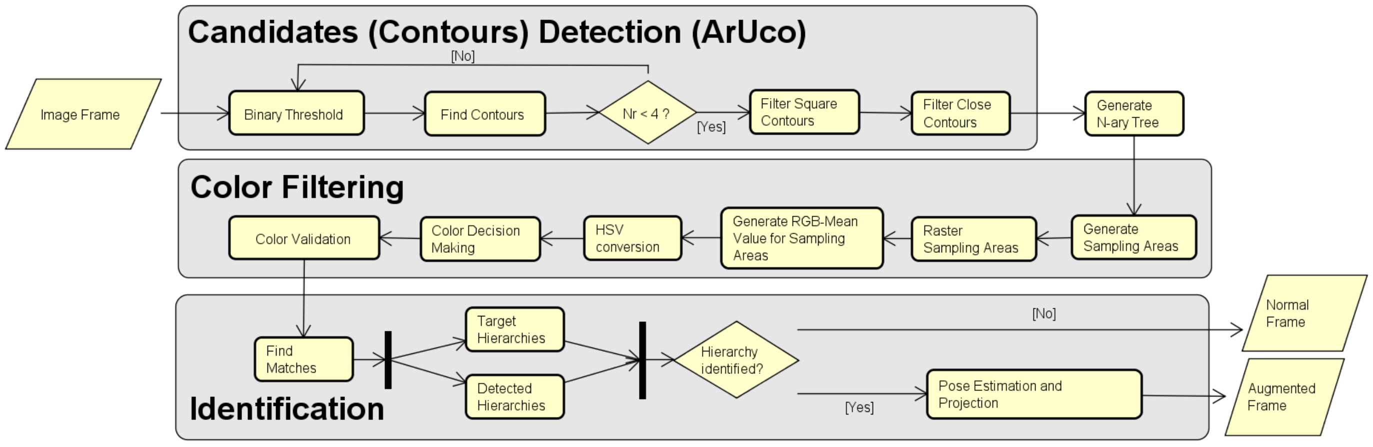


Fig. 8. Flowchart for the CRFM detection.

3) Generate RGB-Mean Value for the Sampling Areas and HSV Conversion: The next step consists of iterating over the pixels of the sampling area and calculate the mean RGB value of the whole region. The average value of the area consists in a single pixel value. This RGB pixel value is converted into an HSV pixel value. We use this technique since handling colors in the HSV format is more robust when determining pixel colors. Since converting from RGB to HSV is a costly operation, we do not convert the whole area, but only the mean value for the area.

4) Color Decision Making: Having determined the HSV pixel value for the mean value of the area, we must analyze its values in order to determine its correct color. An HSV pixel is composed of three values: Hue, Saturation and Value (brightness). To determine the color, we must create a set of intervals into which the HSV values are compared to, ultimately determining its color.

This set of rules was created using a histogram. We capture and analyzed patches of colors under different levels of brightness and noise, then joined these patches together into a single image and generated its histogram. For each analyzed patch, we interpreted the corresponding histogram into rules for each color.

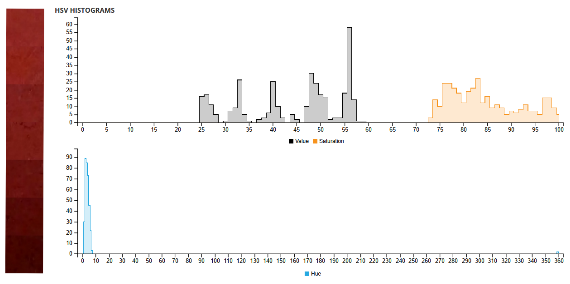


Fig. 9. Red patches sampling (left) and its corresponding histogram (right).

Figure 9 shows that for an HSV value to be considered red, the values must be within the following intervals: Hue [350 to 360] and [0 to 10]; Saturation [70 to 100]; and, Value [20 to 65]. This process has been repeated through all the colors that are used in the CRFM (red, green, blue, black and white).

5) Color Validation: After all rectangles colors have been determined, the color that has the most occurrences is chosen. We also check if at least 5 rectangles, from the set of 8 rectangles, correspond to the same color. Otherwise, we set an undefined color for the block in order to avoid false-positives.

C. Identification and Pose Estimation

This step consists of search over the N-ary trees, finding matches between the target N-ary tree and the detected N-ary tree. The pose estimating uses an algorithm based on the Levenberg-Marquardt algorithm [17] [18].

V. Experimental Evaluation

For the experimental evaluation, we compare both detection accuracy and performance of the CRFM against two boards of the ArUco marker. We used the ArUco marker in the comparison due to its fast performance and high detection accuracy, providing a good benchmark for the CRFM. The markers used in the evaluation are shown in Figure 10. A two-layer CRFM, composed of three levels, was defined (Figure 10 (a)). For comparison purposes, two ArUco boards with 3x3 (9 markers) and 4x4 (16 markers) sizes (Figure 10 (b) and (c), respectively) were generated. All markers have a square shape of 14cm of size.

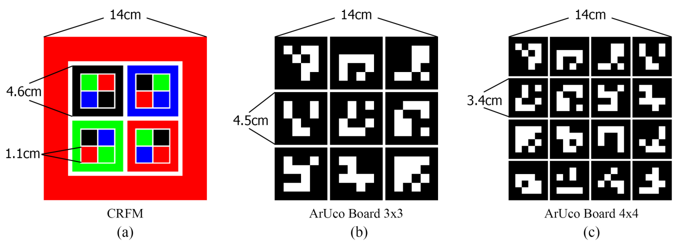


Fig. 10. Markers used in the experimental evaluation.

In order to generate a data-set of images, each marker was printed in an A4-size sheet and pictures of the sheets were

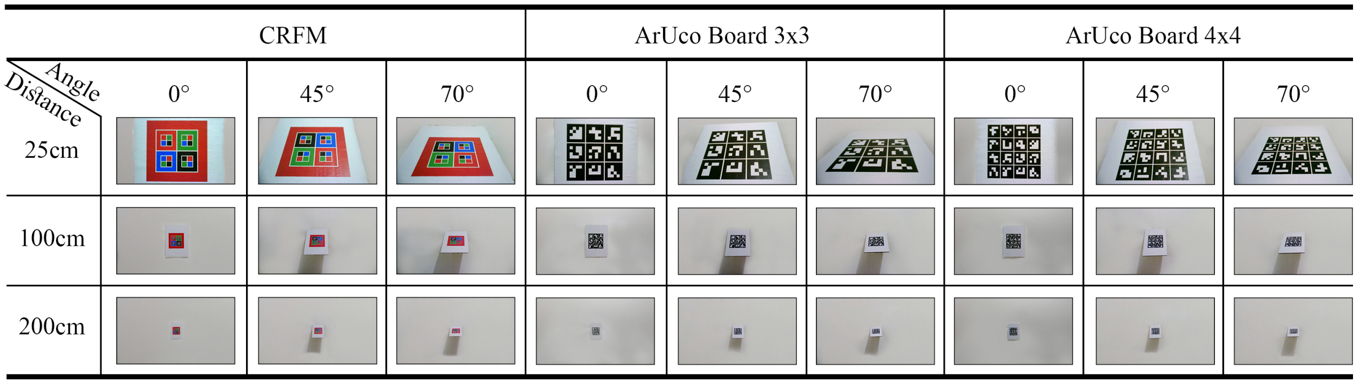


Fig. 11. Sample of images from the data-set.

taken at different distance ranges and angles (samples can be found Figure 12). Both marker position and light conditions were the same for all pictures in the data-set. Pictures were taken using a tripod, and located indoors, with an average luminance (lux) measurement of 200.

Pictures of the sheets (containing the markers) were taken ranging the distance of the camera from 25cm up to 200cm (moving the camera 25cm between each picture) – 8 pictures in total. Moreover, pictures from 3 different angles were also taken 0º (no inclination), 45º and 70º of inclination – generating a total of 24 pictures for each marker. Figure 12 presents a sample of the pictures used in the data-set.

All pictures in the data-set were taken from a Novatek NY99140, 30fps, 1280x720 resolution USB-camera. The evaluation experiments were executed in a laptop with the following specifications: Intel Core i7 6500u dual-core / four threads processor, 8gb 1600mhz DDR3 single-channel memory and Windows 10 64-bit. The CRFM algorithm, as well as the OpenCV [19] library (containing the ArUco algorithm) were implemented and compiled using Microsoft Visual C++ 12.0 compiler.

A. Case Study

Our case study consists of employing the CRFM into a practical Augmented Reality application. This application consists of projecting a 3d model of the AV-LMU-MK6 (Universal Multiple Launcher) military vehicle, over a CRFM marker, verifying the employability of the algorithm into an actual simulation environment for training purposes. This activity is part of the SIS-ASTROS Project between the Federal University of Santa Maria and the Brazilian Army. One of the key scopes of the project is to research and develop tools for training military personnel in the ASTROS system.

Two videos were recorded and processed using the CRFM detection algorithm, with the camera moving from different positions and distances. Samples from the videos are available on the internet [20].

Figures 14 and 13 shows the detection rate considering all the frames from the videos. We recorded an average of 91% to 95% detection rate over the course of the videos. The main cause for failure in detecting the markers was due to fast

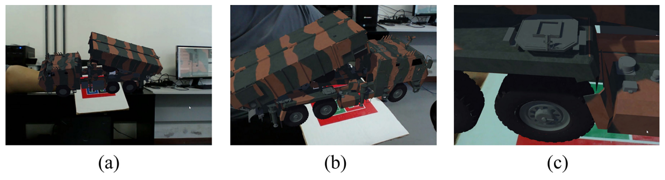


Fig. 12. Algorithm detection and projection at varying distances.

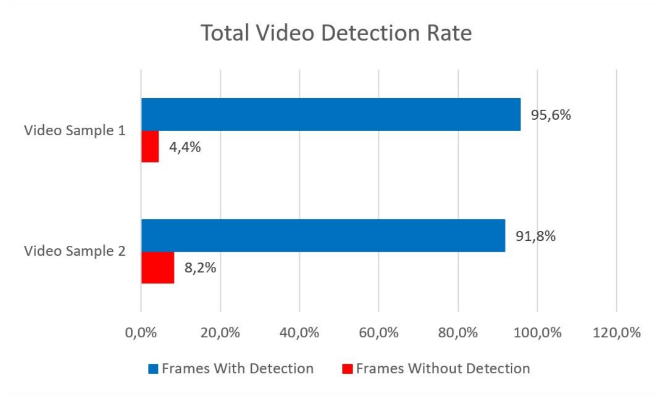


Fig. 13. Detection rate according to each video sample analyzed.

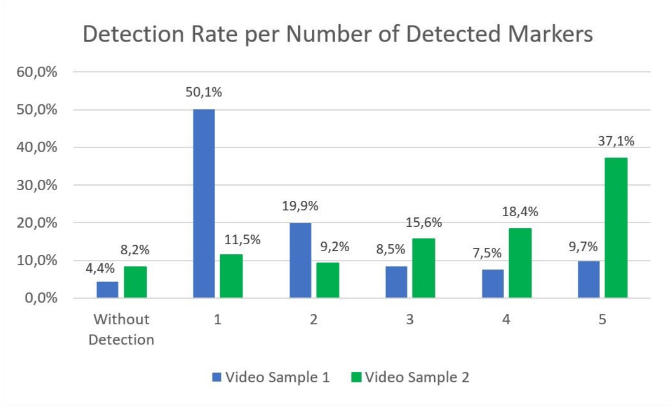


Fig. 14. Detection rate according to the number of markers detected per frame.

movement of the camera (generating blurred images), or the marker being seen from a wide-angle of view.

B. *Detection Accuracy*

For evaluating the detection accuracy for each marker, we considered the following metric. At least one valid hierarchy should be detected in the CRFM and one valid marker in the ArUco boards. We used this metric as it is enough to correctly estimate the projection pose in each marker.

Figure 15 presents an example of the detection accuracy with a picture at 200cm of distance and 70º of inclination. In this example, we can see that the CRFM algorithm is capable of detecting at least one hierarchy, while the ArUco boards have a detection of 7 markers (3x3 grid) and 4 markers (4x4 grid). As the size of the ArUco grid get bigger, the markers become smaller, reducing the detection accuracy of the board.

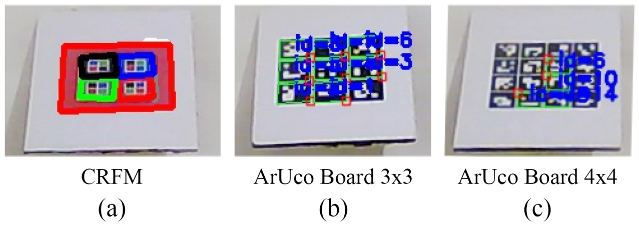


Fig. 15. Example of detected markers at 200cm of distance and 70º of inclination

Figure 16, shows the overall detection accuracy from the markers, according to their distance range and inclination angle. The only case where no detection was possible occurred with two pictures from the CRFM under 70º of inclination. The failure to detect a marker hierarchy can be attributed to the fact that, under higher inclinations, the white border separating the colors in CRFM have their visibility reduced due to certain light conditions, thus, affecting the contour detection algorithm. Such problem can be avoided by either increasing the border thickness or the number of threshold iterations of the algorithm. In the first solution, both design and size of the CRFM is affected. In the latter, more processing time is needed to decode the marker.

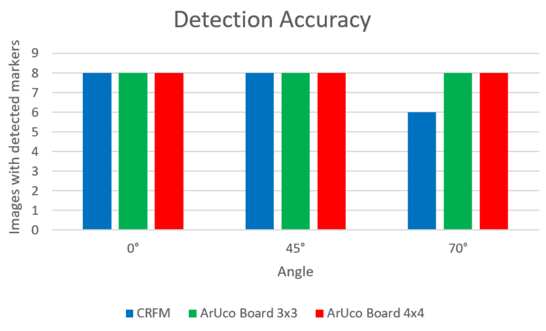


Fig. 16. Detection accuracy for the images in the data-set.

C. *Performance*

For the performance evaluation, all images in the data-set are processed over 300 iterations. The first 100 iterations are ignored in order to generate more stable and reliable results, due to the nature of dynamic clocking speeds of current Intel processors (Intel® Turbo Boost Technology 2.0). The processing time for the performance evaluation measures the time between the start and end of the decoding of the marker in the image. The evaluation does not take into account the required time to load and write images from the storage device.

The average run-time (in milliseconds) for decoding the markers in the data-set are presented in Figure 17 considering the images for 0º (no inclination), 45º and 70º of inclination, respectively, charts (a), (b) and (c). As shown on the charts, it can be deduced that both algorithms performance are affected by distance they are from the marker. We can see that the CRFM has a better performance against ArUco boards in all images of the data-set, except at the images at 25cm with 0º (no inclination), 45º and 70º of inclination. The reason this occurs, is a result of the markers being bigger in terms of pixel quantity. The closer the camera is to the marker, the greater is the necessary processing power required for each marker. CRFM is affected the most, due to its increased processing in order to determine the color of the areas.

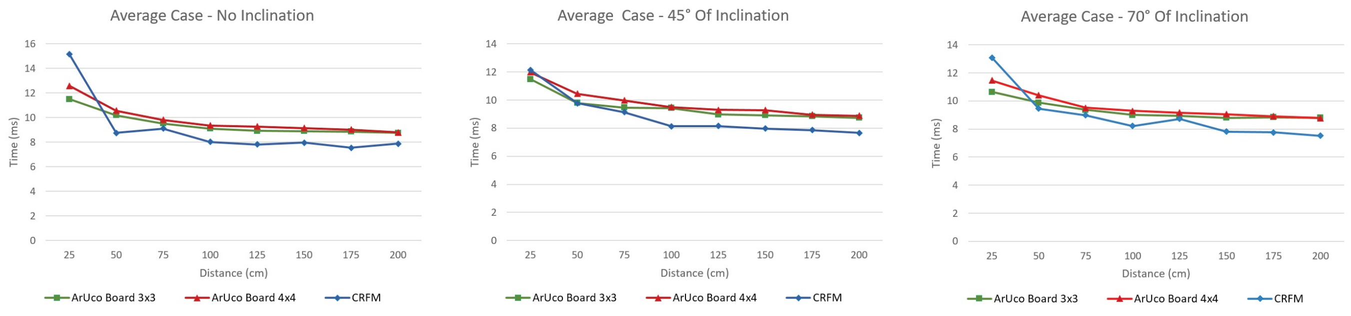
Figure 18 presents the overall performance considering all the images in the data-set. In both best and average cases, CRFM is faster the ArUco boards. Moreover, looking only at the average case, CRFM is 4,88% faster than ArUco 3x3 and 8,04% faster than ArUco 4x4.

VI. Final Remarks and Future Work

The use of fiducial markers for augmented reality enables the quick identification (reducing processing time), which is required for running application in embedded systems, for instance, smart-phones. In this work, we proposed CRFM, a Color-based and Recursive Fiducial Marker. We explored the use of black, red, green and blue colors for the definition of CRFM. However, the idea behind using a color-based marker is to explore other colors that can increase the possible range of hierarchies that can compose the marker. The recursive nature of CRFM enables it use under partial occlusion images. A key component for the fast and accurate projection in augmented reality applications.

When using color for the definition of markers, we have to deal with problems that may not arise over gray-scale markers, since light conditions have a higher impact over the identification and decoding of the marker. In this sense, we used a mixture of RGB and HSV color codifications in order to maintain both processing speed and detection accuracy in CRFM. In our evaluation results, we showed the decoding process for CRFM is generally faster than gray-scale markers, such as ArUco. Nevertheless, the detection accuracy may be worse than gray-scale markers due to light conditions.

As future work, we intend to provide better analysis over the use of CRFM under partial occlusion, including both accuracy and performance details. We also intend to use other colors



(a) (b) (c)

Fig. 17. Average run-time in milliseconds.

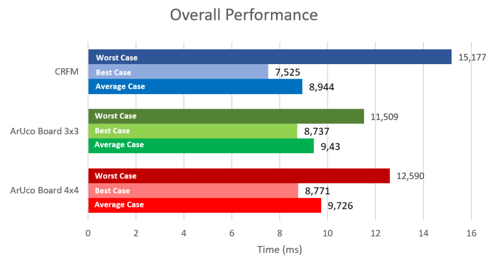


Fig. 18. Overall performance for all images in the data-set.

for the definition of the markers. In this line of work, we are considering the definition of a dynamic algorithms for the analysis and calibration of colors. This is a way of dealing, dynamically, with the problem of light conditions.

VII. Acknowledgements

This work is partially funded by the SIS-ASTROS project in cooperation with the Brazilian Army, and both CAPES and CNPq Brazilian funding agencies.

References

[1] I. E. Sutherland, "A head-mounted three dimensional display," in *Proceedings of the December 9-11, 1968, Fall Joint Computer Conference, Part I*, ser. AFIPS '68 (Fall, part I). New York, NY, USA: ACM, 1968, pp. 757–764. [Online]. Available: http://doi.acm.org/10.1145/1476589.1476686

[2] N. J. Andre, "A modular approach to the development of interactive augmented reality applications." Ph.D. dissertation, The University of Western Ontario, 2013.

[3] J. Sattar, E. Bourque, P. Giguere, and G. Dudek, "Fourier tags: Smoothly degradable fiducial markers for use in human-robot interaction," in *Proceedings of the Fourth Canadian Conference on Computer and Robot Vision*, ser. CRV '07. Washington, DC, USA: IEEE Computer Society, 2007, pp. 165–174. [Online]. Available: http://dx.doi.org/10.1109/CRV.2007.34

[4] H. Kato and M. Billinghurst, "Marker tracking and hmd calibration for a video-based augmented reality conferencing system," in *Proceedings of the 2Nd IEEE and ACM International Workshop on Augmented Reality*, ser. IWAR '99. Washington, DC, USA: IEEE Computer Society, 1999, pp. 85–. [Online]. Available: http://dl.acm.org/citation.cfm?id=857202.858134

[5] M. Billinghurst, A. Clark, and G. Lee, "A survey of augmented reality," *Found. Trends Hum.-Comput. Interact.*, vol. 8, no. 2-3, pp. 73–272, Mar. 2015. [Online]. Available: http://dx.doi.org/10.1561/1100000049

[6] M. Fiala, "Artag, a fiducial marker system using digital techniques," in *Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05) - Volume 2 - Volume 02*, ser. CVPR '05. Washington, DC, USA: IEEE Computer Society, 2005, pp. 590–596. [Online]. Available: http://dx.doi.org/10.1109/CVPR.2005.74

[7] 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. [Online]. Available: http://dx.doi.org/10.1007/s00138-012-0469-6

[8] G. Schweighofer and A. Pinz, "Robust pose estimation from a planar target," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 28, no. 12, pp. 2024–2030, Dec. 2006. [Online]. Available: http://dx.doi.org/10.1109/TPAMI.2006.252

[9] L. Naimark and E. Foxlin, "Circular data matrix fiducial system and robust image processing for a wearable vision-inertial self-tracker," in *Proceedings of the 1st International Symposium on Mixed and Augmented Reality*, ser. ISMAR '02. Washington, DC, USA: IEEE Computer Society, 2002, pp. 27–. [Online]. Available: http://dl.acm.org/citation.cfm?id=850976.854961

[10] 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 Recogn.*, vol. 47, no. 6, pp. 2280–2292, Jun. 2014. [Online]. Available: https://doi.org/10.1016/j.patcog.2014.01.005

[11] D. Wagner and D. Schmalstieg, "Artoolkitplus for pose tracking on mobile devices," 2007.

[12] K. Tateno, I. Kitahara, and Y. Ohta, "A nested marker for augmented reality," in *ACM SIGGRAPH 2006 Sketches*, ser. SIGGRAPH '06. New York, NY, USA: ACM, 2006. [Online]. Available: http://doi.acm.org/10.1145/1179849.1180039

[13] A. Herout, M. Zacharias, M. Dubska, and J. Havel, "Fractal marker fields: No more scale limitations for fiduciary markers," in *Proceedings of the 2012 IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*, ser. ISMAR '12. Washington, DC, USA: IEEE Computer Society, 2012, pp. 285–286. [Online]. Available: http://dx.doi.org/10.1109/ISMAR.2012.6402576

[14] S. Suzuki and K. be, "Topological structural analysis of digitized binary images by border following," *Computer Vision, Graphics, and Image Processing*, vol. 30, no. 1, pp. 32 – 46, 1985. [Online]. Available: http://www.sciencedirect.com/science/article/pii/0734189X85900167

[15] D. H. Douglas and T. K. Peucker, "Algorithms for the reduction of the number of points required to represent a digitized line or its caricature," in *The canadian cartographer*. Cartographica: The International Journal for Geographic Information and Geovisualization, 1973.

[16] J. E. Bresenham, "Algorithm for computer control of a digital plotter," *IBM Systems Journal*, vol. 4, no. 1, pp. 25–30, 1965.

[17] D. W. Marquardt, "An algorithm for least-squares estimation of nonlinear parameters," *Journal of the Society for Industrial and Applied Mathematics*, vol. 11, no. 2, pp. 431–441, 1963. [Online]. Available: https://doi.org/10.1137/0111030

[18] K. Levenberg, "A method for the solution of certain non-linear problems in least squares," *Quarterly of applied mathematics*, vol. 2, no. 2, pp. 164–168, 1944.

[19] Itseez, “Open source computer vision library,” https://github.com/itseez/opencv, 2016.
[20] C. Marker. (2017) Crfm video samples. [Online]. Available: https://www.youtube.com/watch?v=ZXCDb2CQPM0&list=PLt8RqhhEaUtBlw61exlRugcy5JrZD36MD