Corner-based Square Fiducial Marker Detection for Hand-manipulated AR Objects

Breawn Schoun
Computer Science and Engineering
University of Colorado Denver
Denver, Colorado, USA
breawn.schoun@ucdenver.edu

Hawkar Oagaz
Computer Science and Engineering
University of Colorado Denver
Denver, Colorado, USA
hawkar.oagaz@ucdenver.edu

Min-Hyung Choi
Computer Science and Engineering
University of Colorado Denver
Denver, Colorado, USA
min.choi@ucdenver.edu

***Abstract*—Square fiducial markers are commonly used in Augmented Reality (AR) applications to affix AR content to a particular location in the real world. Unoccluded, these markers are quickly and easily identified, and AR content is realistically displayed in real time. However, because most square fiducial marker libraries use a thresholding-based method of detection, small edge occlusions often prevent markers from being found, or cause inaccurate estimations of marker pose. Both of these scenarios result in visual disturbances in the AR content. This is particularly problematic for hand-manipulated AR objects, where markers will suffer frequent edge occlusions by fingers. In this paper, we propose an alternative method of detecting single square fiducial markers where only two diagonal corners of the marker must be visible for detection. Our proposed method finds and classifies corners in the image, pairs candidate diagonal corners based on their gradient directions, and then attempts to find a homography between a standard template and corners in the image that may belong to a marker. Identification of potential markers is done using a commonly-used square fiducial marker identification algorithm. This method detects markers under partial edge occlusions at a rate of up to 2.48x that of a popular square fiducial library, detects and localizes square fiducial markers in isolated frames, and is fast enough to be used for real-time applications.**

***Index Terms*—Augmented reality, marker, occlusion, corners**

I. Introduction

Marker-based Augmented Reality (AR) is a way to affix virtual content to a particular location in space, making for engaging, stable and realistic AR experiences. A variety of marker designs and detection algorithms have been proposed, with one of the most widely-used designs being square fiducial markers such as ArUco [1], [2] and AprilTag [3], [4]. These markers are composed of a black square border surrounding a black and white bit pattern used to uniquely identify each marker, and can be quickly and accurately identified and localized for real-time AR applications.

Because our primary research interest is to create AR games for therapeutic and diagnostic applications, we use square fiducial markers frequently. In our research, hand-manipulated items, such as small wooden cubes, are affixed with markers to create dynamic and interactive passive haptic AR objects. This intended use places significant restrictions on marker size, layout, and encoding scheme; the markers must be visible even when affixed to a small surface, and several uniquely-identifiable markers are required. Marker detection needs to run in real time to create a realistic experience for the user, and the algorithm needs to be robust against edge occlusions by fingers that are commonplace in our applications.

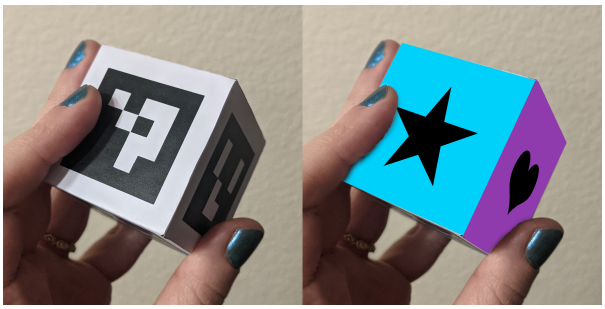

Fig. 1. Example of ideal output when overlaying AR content onto an object.

The requirement that has been most difficult to satisfy using existing marker schemes is robustness to edge occlusions. A frequent problem with our applications is that a user will hold an object in such a way that a finger just slightly overlaps a marker edge, causing problems with marker detection and localization. Because popular marker libraries such as ArUco use a thresholding-based approach to find markers, these minimal occlusions cause the marker to either not be detected or cause the marker corners to be estimated incorrectly, resulting in bad estimations of the position and orientation of the marker [1]. Both cases cause the projected AR content to be visually disturbing to the viewer; the AR content will either completely disappear to show the underlying marker, or the AR content will be jittery and erratic.

The problem of partial occlusion of markers has been explored in the existing literature. AprilTag's original image gradient-based algorithm could handle some partial edge occlusions [3], at the cost of increased computation time and reduced identification accuracy, resulting in this method's abandonment in future AprilTag versions [4]. ArUco, which has poor edge occlusion handling for individual markers, works around the problem of edge occlusions by using marker "boards" that allow the pose of occluded markers to be inferred from the pose of un-occluded markers [5]. However, this method has limited utility for applications involving small

978-1-6654-0347-4/21/$31.00 ©2021 IEEE
DOI 10.1109/ICIR51845.2021.00014

target objects. Fractal markers, where square fiducial markers are embedded within one another to improve detection at varying distances and allow edge occlusions, also have limited utility for small target sizes, and additionally have an inadequate number of uniquely-identifiable markers [6]. Contextual approaches that rely on previous detections have been proposed, such as using a Kalman filter to estimate marker poses [7] or tracking points of interest between frames [6], [8]. Though these methods can be complementary to marker detection algorithms, situations where users maintain an edge-occluding grip on an object for an extended period of time or rotate the marker out of view of the camera create significant gaps in information, resulting in increasingly erroneous pose estimations over time. Improving the rate of detection in cases of slight occlusions would provide these methods with more frequent information and create a more seamless experience. For these reasons, we have developed an approach to square fiducial marker detection and localization that combines the strengths of existing methods and expands upon them to meet our unique application needs, particularly the allowance of partial edge occlusions.

We have developed a square fiducial marker detection method called AR Direction Vector Related Corners (ArDVRC) that searches for markers by corners instead of using a thresholding-based method. The core idea underpinning this method is that detected corners in an image that belong to a marker will have a specific relationship with one another; our method specifically looks for relationships between detected corners that may indicate that they are diagonal corners of a marker. If two corners that may be a diagonal pair are found, the algorithm estimates the intersection points of the corners' edges to approximate the other two possible outer corners. This quadrilateral and the corners contained within are then matched to a standard corner type template of internal and external marker corners using a nearest neighbor search [9], and a homography between the template and image corners is derived using a RANdom SAmple Consensus (RANSAC) algorithm approach [10]. Marker candidates are identified using the standard ArUco identification method [1], [2].

The primary contribution of this paper is the development of a novel method of square fiducial marker detection that can identify single markers under partial occlusion in real time, and that works on individual frames in isolation rather than relying on context from previous marker detections. We evaluate the ArDVRC method by comparing it with ArUco's method of detection and localization, running both on a generated synthetic dataset of 110,000 images and real recorded video data. Our results demonstrate that our method is capable of running in real time, is comparable to ArUco's method in terms of accuracy for unoccluded objects, and performs significantly better than ArUco's method in cases of edge occlusions. Though our application is targeted towards hand-manipulated AR objects, the main contribution of our technique can be utilized for broader interactive applications. ArDVRC sample code and data is available on GitHub [11].

This paper is structured as follows: we will first discuss related work, and then provide detailed information about our method. We will then discuss our experimental design, our results, our method's limitations, and future work.

II. Related Work

Square fiducial marker detection algorithms have been proposed that have some resilience to edge occlusions. AprilTag's original marker detection algorithm searched for possible outer marker edges in an image using a gradient-based method, and was able to handle some cases of edge occlusions [3]. However, as mentioned in their subsequent work [4], the first version suffered from errors during the identification step, particularly during cases of partial occlusion. In addition, it was costly in terms of computation time, and users seldomly used the partial occlusion feature. For these reasons, the authors switched to an adaptive thresholding method in AprilTag 2 and did not attempt to handle cases of partial occlusion. Despite its drawbacks, the original AprilTag algorithm inspired ArDVRC's method of corner directionality extraction, which uses an image gradient-based method similar to that used by the original AprilTag algorithm.

Other marker schemes have been proposed that are able to handle a significant amount of occlusion. Runetags, a circular tag composed of several dots, can handle up to 70% occlusion [12]. However, the framerate is too low for real-time applications, limiting the utility of this method.

Several solutions have been devised to work around the occlusion problem of individual markers. One robust method is the use of marker "boards", or a multiplicity of markers in a specific arrangement that can be used to infer the pose of other markers [5]. While this solves the problem for many use cases, it has less utility when the target has specific size requirements. If the target cannot be made to be larger, the individual markers must be made smaller, to the point where they may not be adequately detected by a camera.

A recent innovation in square fiducial markers is the Fractal marker [6], which is a marker with smaller markers embedded within. This paradigm helps to alleviate distance issues, where the edges of a marker may be occluded at near distances, or where smaller markers may not be recognized at further distances. This is an appealing prospect when considering that users of our system may move an AR object closer or further away from the camera. However, this method is still restricted by the target size. When the largest marker has an occluded edge, the inner markers could be too small or too far away to be recognized. Additionally, Fractal markers do not yet have several unique identifiers, and for our application we want to have multiple AR objects. Though this paradigm isn't a good fit for our application in its current form, this paper also introduces a corner classification method that classifies corners by their arrangement of black and white pixels, which is used to track the marker between frames when the marker is occluded. Our algorithm uses a corner classification method that is based on the method described in this paper, but improves upon it to not only more accurately classify corners, but to additionally extract corner directionality.

Contextual information from previous frames can be used to track markers in conditions where the marker cannot be found [6], [8], and can also be used as input to a Kalman filter, which can estimate the pose of a previously detected marker in future frames [7]. These methods can be used to fill in the gaps of missing marker detections and to filter out erroneous poses. Though these methods are often advantageous, these methods become erroneous when points of interest move out of sight of the camera, or during extended periods without any detections. In our specific application where the user is manipulating a 3D object in their hands, the marker may be rotated out of view of the camera, or a user may grip the AR object in a particular way that occludes the marker for a significant amount of time. To avoid increasingly incorrect estimations due to these conditions, it would be beneficial to improve marker detections under these circumstances to decrease reliance on contextual methods.

In summary, a variety of methods have been proposed to deal with the problem of edge occlusions. Yet, each has individual limitations that make it less than ideal for use for hand-manipulated AR applications.

III. Method

A. Custom marker dictionary

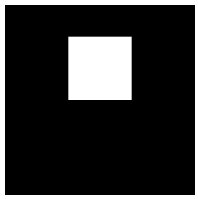

Fig. 2. ArUco marker 17.

Due to the size constraints of our markers and objects, the markers we use need to be clearly visible at varying distances. For that reason, ArUco's 4X4 marker dictionaries are appealing; the marker has the fewest bits of all of ArUco's marker dictionaries, meaning the black and white squares are as large as possible, yet many unique IDs can be generated. However, some markers in ArUco's 4X4 marker dictionaries are visually simplistic and can be confused with naturally-occurring shapes in the environment, resulting in false positives. Marker number 17, for example, is simply a white square, and is frequently falsely identified in indoor environments.

The reason marker number 17 is part of the ArUco 4X4 dictionary is because it has a high intra-marker Hamming distance (the minimum Hamming distance between all four rotations of the marker) and a high inter-marker Hamming distance (the minimum Hamming distance between the marker and every rotation of every other marker in the dictionary). To put it simply, each ArUco marker was designed to be distinguishable from all other markers and itself for all rotations of the marker, and marker 17 fits this criteria.

In addition to marker number 17's tendency towards false positives, marker number 17 also has the fewest number of corners possible, which is 8 (4 outer corners and 4 inner corners). Because ArDVRC is based on finding the outer and inner corners of the marker, we want the marker to have as many corners as possible to improve our chances of detection. By that metric, marker number 17 is the worst-case scenario. We want each of the markers in the dictionary to not only be optimized for high inter- and intra-marker Hamming distances, but to be optimized for entropy as well.

For these reasons, we created a custom dictionary that is a subset of ArUco's $DICT\_4X4\_100$ dictionary. We chose to use a subset of an ArUco dictionary because ArUco dictionaries are already optimized in terms of intra- and inter-marker Hamming distances. By selecting a subset of these markers, we only stand to improve the inter-marker Hamming distances while selecting the markers with the highest entropy.

To select our marker subset, we first iterate through each marker in the dictionary and calculate the number of internal and external corners and the intra-marker Hamming distance. We then select thresholds for these marker qualities, choosing our threshold values to produce a target of 50 markers. Towards this goal, we chose markers with a minimum intra-marker Hamming distance of 6 or greater, and a minimum corner count of 17 or greater, resulting in a marker set of 64 markers. The minimum inter-marker Hamming distance of our chosen set is 3. We use this custom corner-optimized and Hamming distance-optimized dictionary in place of the $DICT\_4X4\_100$ dictionary for both ArDVRC and ArUco detection algorithms to ensure an equal comparison.

B. ArDVRC marker detection algorithm

1) Corner classification and direction estimation: The first step in the ArDVRC algorithm is to locate and classify corners within the image. To locate corners we use the Harris corner detection algorithm [13] and a corner peak detection algorithm [14]. Once the corners are located within the image, we classify the corners and estimate each corner's edge directions.

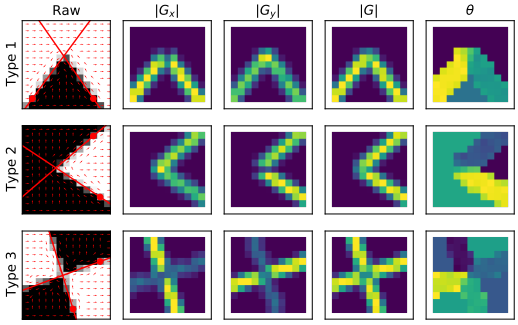

Fig. 3. Image gradient-based approach used to find the two direction vectors. The image is then divided into four quadrants for corner classification.

Inspired by the corner classification technique used in the Fractal markers paper [6], we classify the corners in the image by whether they are mostly white (type 1), mostly black (type 2), or a checkerboard pattern (type 3), and reject corners that do not fit into these categories. An illustration of each of these types of corners can be seen in the first column of Figure 3.

Our method differs from the Fractal method in that we use an image gradient-based approach to classify corners instead of counting black and white pixels, making our method more resilient against misclassification. Additionally, we use the image gradients to estimate corner directionality.

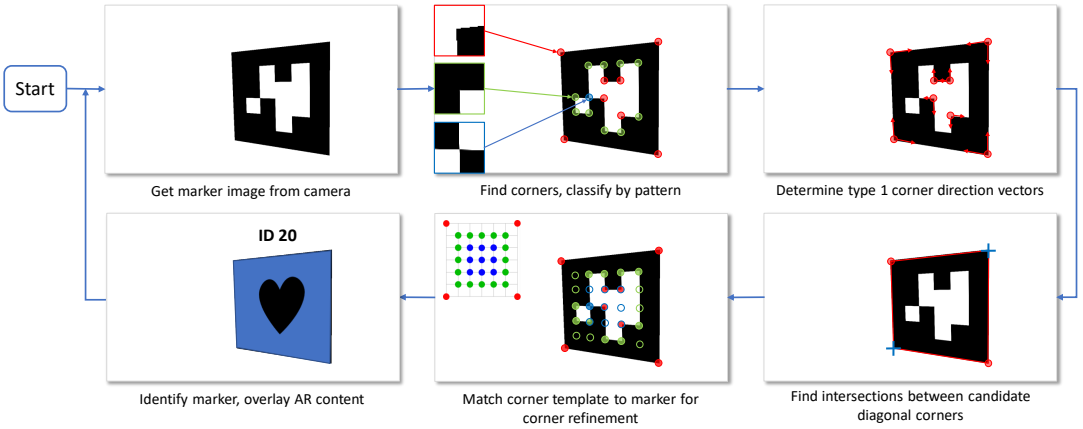

Fig. 4. Method overview.

Classification and direction estimation begins by selecting a small region of interest (ROI) around each corner in the image. We apply a Sobel operator to the ROI to get the image gradients in the x and y directions, and then find the gradient magnitude and direction of each pixel. We take the absolute value of the x and y gradients, shown as $|Gx|$ and $|Gy|$ in Figure 3, and find the maximum value along the border of both $|Gx|$ and $|Gy|$. The θ angle value at these two pixel locations provide direction vectors that are used to draw the lines shown in Figure 3. For valid corners, the two lines should closely align with the edges in the ROI.

The ROI is then segmented into four quadrants along these two lines, and normalized between 0 and 1. Each quadrant of pixels is averaged to determine if the pixels are mostly white (> 0.5) or mostly black ($<= 0.5$). This reduces the information in the ROI to a series of four binary values, the arrangement of which indicates the corner type. All rotations of the sequence $\{1, 1, 1, 0\}$ indicate a type 1 corner, all rotations of the sequence $\{0, 0, 0, 1\}$ indicate a type 2 corner, and all rotations of the sequence $\{1, 0, 1, 0\}$ indicate a type 3 corner. Any corner that is not categorized as one of these three types is labeled 0 and considered unknown.

The information gained about corner type and directionality guides our search for markers within the image by helping determine which corners pairs may have a diagonal relationship.

2) Corner direction extrapolation: Once we've classified the corners and calculated their direction, we can search for candidate markers within the image. We do this by searching for two type 1 corners that may be diagonal outer corners of a marker. Good candidate corner pairs will have direction vectors that have singular intersections with each other; as illustrated in Figure 5, each vector from corner A intersects with a vector from corner B, but no vector from corner A intersects both vectors of corner B (and vice-versa).

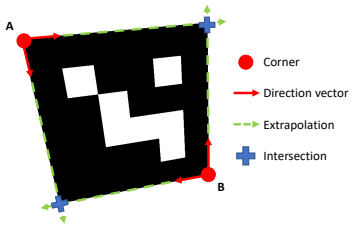

Fig. 5. A corner pair with singular intersections. Each vector crosses one vector from the other corner.

To find good candidate corner pairs, we extrapolate the direction vectors to find the intersections between all type 1 direction vectors. This is accomplished by tiling type 1 vectors vertically and horizontally in two matrices, and then calculating the intersections of the two matrices. This results in a symmetric matrix composed of 2D points for vectors that intersect and NaN values for vectors that do not.

Because there are two direction vectors per corner, the intersections of two corners can be represented as a 2X2 matrix containing some combination of 2D points and NaN values. A singular intersection for this 2X2 matrix will have two 2D intersection points and two NaN values, arranged in a checkerboard pattern. If the two type 1 corners have singular intersections, the corners are combined with their intersection points to create an array of four corners, which will be used as a starting point for template matching.

3) Template matching: All square fiducial marker corners that have a solid black border adhere to a standard corner type pattern that can be seen in Figure 6. The outer corners of the marker are all of type 1, are always present and always have a consistent direction. The corners one level inward are all type 2, are not always present and do not have a consistent direction. Corners further inward can be of any type and have any direction, and may or may not be present.

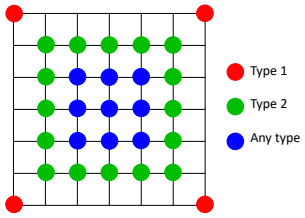

Fig. 6. Marker corner template.

We relate this template to the candidate corners by finding a homography that aligns the template with the candidate corners. We then use Picoflann's KDTree implementation to do a nearest neighbor search between the transformed template points and the image corners [9]. Three KDTrees are used, with one for each template type. A RANSAC algorithm is used to find a homography between the template and the corner points [10], and the outer corners are then used as input to ArUco's identification method.

IV. Experimental Setup

Experiments were carried out with both synthetically-generated data and real data. There are two reasons why we chose to generate synthetic data: first, the ground truth information (ID and position of the ArUco marker) is not known for every frame of real data without painstaking analysis of each frame. By generating the data, we have exact ground truth information with which we can compare our algorithm and the ArUco algorithm. Second, generated data can be manipulated precisely to create the exact conditions that we want to test our algorithm against, such as different blur or lighting effects.

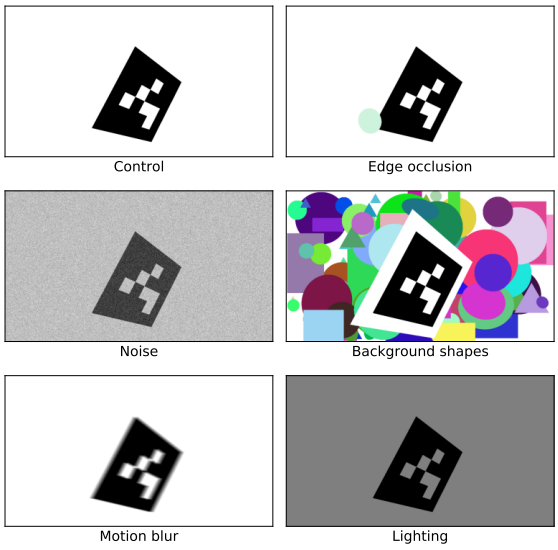

Fig. 7. A sample of the generated synthetic dataset.

We additionally tested our algorithm using recorded videos to ensure that our algorithm would work on real-world data. Though there's no ground-truth data available for these videos, we're able to count marker detections for both ArDVRC and ArUco for comparison, and can qualitatively assess the marker detection behavior between the two methods.

A. Synthetic generated data

A synthetic control dataset of 10,000 images of individual markers was generated that randomizes the ID, pose and size of the marker within the image. In order to simulate a variety of realistic conditions that may be encountered during normal operation, five common circumstances for hand-manipulated AR content were simulated in the dataset: finger occlusions, sub-optimal image quality, noisy background environments, motion blur, and reduced lighting conditions. These real-world conditions were synthesized by adding noise, blur filters, background shapes, and foreground shapes, and by manipulating the image contrast of the control dataset. Each of these conditions had their own dataset of 10,000 images. Performance was assessed by determining (a) whether the algorithm is able to correctly identify markers within the image, (b) how close the corner estimates are to the ground truth information, and (c) whether the algorithm is capable of running in real time. We tested our algorithm on our control dataset and these five simulated conditions (a total of 60,000 images). We then compared our results and ArUco's results to the ground truth data.

To further examine performance for occluded markers, we generated an additional 50,000 marker images with edge occlusions. This data was used to compare each method's performance for different percentages of occlusion. In total, our synthetic dataset consisted of 110,000 images.

Data generation was done in Python, and was accomplished using the OpenCV library [15] and Scikit-Image [14]. These images were stored as TIFF files with all of the ground truth information stored as metadata within each image. Figure 7 shows samples of our control images and the different manipulations that were applied to the images.

B. Real data

We have created an AR cube using the first 6 markers from our custom high-entropy dictionary (IDs 0-5), and recorded 3 minutes-worth of video clips of this cube being manipulated by hand, purposefully introducing slight to significant finger occlusions across the markers. We process these videos using ArUco and ArDVRC, and draw the detected IDs onto the cube. This data is analyzed by considering the number of detections for both methods on the same data, and also visually assessing the projection behavior for both methods.

V. Results

A. Synthetic generated data

The following subsections examine the results from the synthetic data. Each subfigure in Figure 8 shows the identification rate and error rate for that set of 10,000 images, as well as a histogram of the corner counts and their position error. It should be noted that for each of these figures, the y axis is on a logarithmic scale to make the results more clearly visible.

1) Control data: Figure 8 (a) shows a comparison of ArDVRC and ArUco accuracy on the control dataset. The control dataset is indicative of how each algorithm behaves under optimal conditions where no blur, noise, shadows, or occlusions are present. The results show that both methods perform quite well on this dataset. The ArUco algorithm slightly out-performs ArDVRC by correctly identifying 99.31% of the dataset while ArDVRC correctly identified 98.61%. ArUco additionally has a slightly lower error rate for corner positions.

2) Edge occlusions: The edge occlusion dataset, composed of marker images with a single elliptical edge occlusion, is meant to simulate the presence of a finger occlusion on the marker's edge, and is the data of most interest to us for our application. Figure 8 (b) shows that ArDVRC performs significantly better than ArUco in cases of edge occlusions. For this dataset, ArDVRC successfully identified 70.14% of the markers in the dataset, while ArUco identified 28.23% of the markers and had a higher error rate than ArDVRC. In other words, the ArDVRC method was able to identify 2.48x more markers than ArUco, and estimated the marker corner

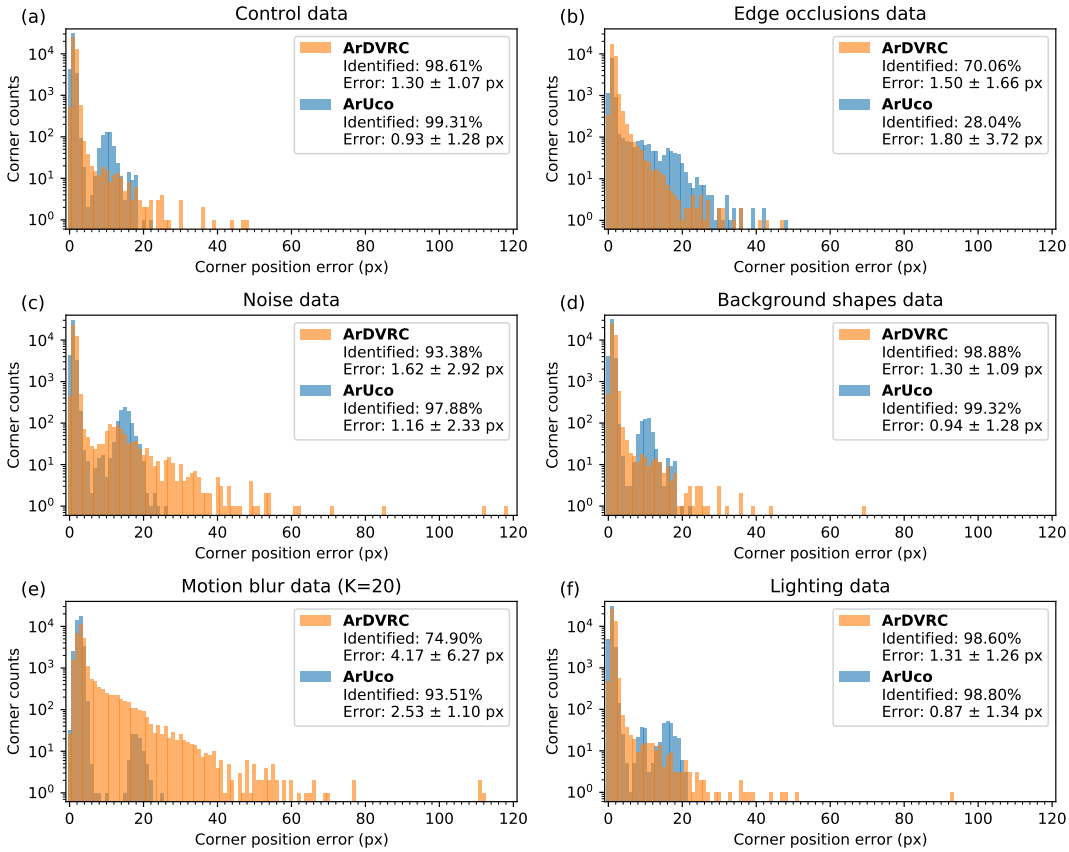

Fig. 8. Results from the control data and the five different manipulations of that data. Subfigure (b) shows that ArDVRC has a significantly higher detection rate and higher corner accuracy than ArUco for partially-occluded markers.

locations more accurately as well. It is worth noting that for this dataset, the locations, sizes and shapes of the ellipses in the images are random, and therefore some occlude the marker to such a degree that the marker couldn't possibly be uniquely identified. If we modified this dataset so that all 10,000 images could theoretically be identified, we would likely see the identification rate improve for each method.

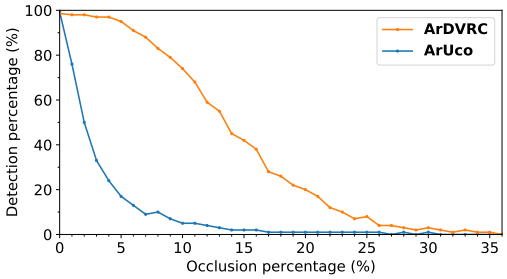

Fig. 9. Detection rate vs. percent occlusion for ArDVRC and ArUco.

We additionally examined the effect of occlusion percentage on the detection rate for each algorithm using a dataset of 50,000 partially-occluded marker images. The results of this comparison can be seen in Figure 9, which shows that while ArUco has significantly reduced detections for 5% occlusion, ArDVRC has a more gradually-declining detection rate, demonstrating its robustness against edge occlusions.

3) Noise data: The noise dataset simulates reduced image quality by adding speckling in the foreground. The results displayed in Figure 8 (c) show that ArDVRC is less resilient to noise than ArUco, but that it still performs well. ArUco identified 97.88% of the markers in this dataset while ArDVRC identified 93.38%. As with the control data results, ArDVRC had a slightly higher corner position error than ArUco.

4) Background shapes: The background shapes dataset simulates a busy background environment. Because ArDVRC is corner-based, we wanted to see if the presence of a significant number of corners would cause problems such as false detections or reduced identification rates. Interestingly, both ArUco and ArDVRC improved their identifications slightly on this dataset compared to the control, as seen in Figure 8 (d), and the error rate for both remained largely the same. There were additionally no false detections for either method.

5) Motion blur: This dataset simulates fast head or hand movements by the user. ArDVRC's and ArUco's performance on motion blur data is heavily dependent on the amount of blur in the image. Figure 8 (e) shows results for a blur kernel of 20 pixels, but we additionally calculated results for blur

kernels of 10 and 30 pixels. For a blur kernel of 10 pixels, both algorithms perform well. However, for a kernel of 30 pixels, ArDVRC detects only 4.45% of the dataset while ArUco is able to identify 75.84%. The accuracy of both degrade as the blur becomes more significant, but ArUco outperforms ArDVRC in instances of high motion blur. This is further discussed in the Discussion section.

6) Lighting: Poor lighting conditions were simulated by reducing the contrast of the control images by 50%. ArDVRC and ArUco both performed well on this dataset, with identification rates of 98.6% and 98.8%, respectively. ArDVRC had a slightly higher corner position error than ArUco.

B. Real data

Three minutes of video of a person manipulating a black and white marker cube was processed using both ArUco and ArDVRC. Figure 10 shows stills from the video sequence. The number of cube detections for both methods were counted and compared. In total, ArUco detected the cube 6383 times while ArDVRC detected the cube 8816 times, meaning that ArDVRC had 1.38x as many marker detections as ArUco. Though this rate is lower than the edge-occluded synthetic data, this is expected because the marker did not have edge occlusions in every frame of the video. Qualitatively, both the ArUco and ArDVRC corner detections looked to be aligned with the cube. This shows that for real-world conditions involving frequent partial occlusions, the ArDVRC method is able to detect markers at a higher rate than ArUco, which is key to providing a more seamless AR experience.

C. Real-time performance

Dataset	Avg. runtime [ms]
Control	7.943 $\pm$ 5.048
Noise	16.883 $\pm$ 6.027
Motion blur	8.163 $\pm$ 5.139
Shapes	37.055 $\pm$ 14.038
Occlusions	7.573 $\pm$ 5.665
Lighting	7.341 $\pm$ 5.167

Because we intend to use this software for real-time games and therapeutic activities, our goal was to keep the algorithm runtime less than 30 ms per frame. We've mostly made it to this goal except for the background shapes data, as shown in the table above. It's worth noting that the runtime of this algorithm is a function of image size (corner detection and image gradients) and number of corners found within the image. As shown in Figure 7, the background shapes image has the most corners of all of the datasets. The more candidate corner pairs are found, the higher the computational workload. However, this method lends itself well to parallelization, and we were able to parallelize the code to the point where the bottleneck became OpenCV's RANSAC function. The above table shows the average runtimes for each dataset.

VI. Discussion

The results show that ArDVRC and ArUco have their own strengths and weaknesses depending on the context, but that ArDVRC significantly outperforms ArUco in ability to detect markers under partial occlusion. This is because ArUco's detection algorithm uses a thresholding-based approach to find markers, and will determine marker candidates based on whether the contours of a shape in the thresholded image approximate a quadrilateral [1]. Because edge occlusions will often disturb the shape of the marker in the thresholded image, even mostly-visible markers will often not be considered as possible candidates; those that are considered and correctly identified often suffer from erroneous corner estimations, which in turn affects the pose of the AR content. Removing the requirement that the entire square be visible in favor of a technique based on visible corner information increases the number of detections under partial occlusion and improves the marker corner estimations for partially-occluded markers.

As shown in the the Results section, ArDVRC had 2.48x more detections for the synthetic partially-occluded data than ArUco, and had 1.38x more marker detections than ArUco for real video data with frequent partial occlusions. Additionally, we've shown that ArDVRC can handle higher occlusion percentages than ArUco. This demonstrates that the ArDVRC method could be a viable alternative to thresholding-based methods of marker detection, especially when the application is expected to encounter frequent edge occlusions. These increased detection rates could help alleviate some of the visual disturbances arising from problems of partial occlusion.

Figure 8 shows a few cases of high pixel error for the ArDVRC method in different datasets. This is because occasionally the direction vectors for a corner are not well-aligned with the marker edge, particularly in cases of blur and image noise, and therefore the intersections of the vectors are significantly far from where they should be. Because the initial corner estimate is poor, the template matching step isn't able to match corners appropriately and the poor initial estimate is used for the identification step. Usually markers with poor corner estimates are rejected during the identification step because the candidate marker image has significant flaws, but on rare occasions a marker is identified. These occurrences can likely be reduced by ensuring a minimum number of matched corners during the template matching step, or by using an identification algorithm that takes internal corners into account.

Neither ArUco nor ArDVRC performs exceptionally well for high-blur situations, and the results show that ArDVRC fails sooner than ArUco. This is because as the blur increases, the corner pixels are more difficult to locate and the corner directions become less defined. However, an adaptive thresholding technique like the one used for the ArUco algorithm will likely still be able to find the square shape of the marker, and will therefore be more robust to blur. A potential way to alleviate performance issues in high blur situations for ArDVRC would be to fall back to the ArUco algorithm when the ArDVRC method fails to detect a marker.

The ArDVRC algorithm currently relies on ArUco's method of identification that warps the candidate corners to a square ROI, segments the image into bits and identifies the marker based on the extracted information. We are interested in ex-

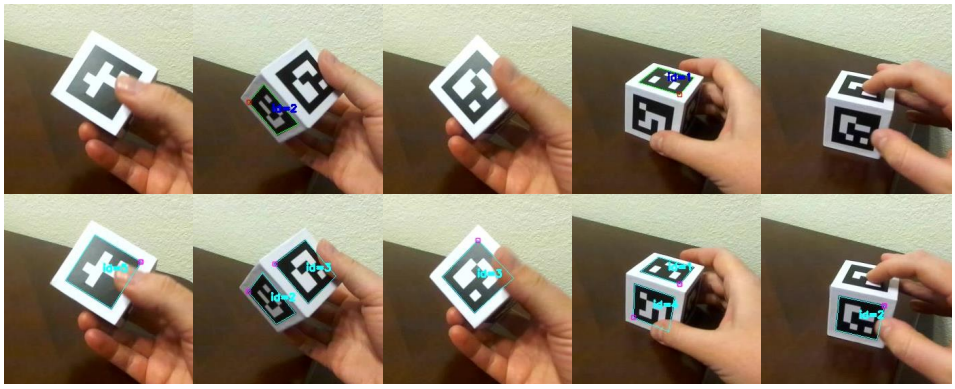

Fig. 10. Video sequence stills comparing ArUco detection (top) and ArDVRC detection (bottom).

ploring whether the corner types, template-matched locations and direction vectors provide enough information to uniquely identify the markers, or if a corner-based method may be able to match the visible portion of an occluded marker to possible candidates. This could provide further robustness against partial occlusions in situations where the marker corners may be found but the marker is rejected at the identification step.

The ArDVRC template matching step uses the RANSAC algorithm to find the best homography between the template points and the image points. While this works well overall, RANSAC is currently the greatest bottleneck in the program. We would like to explore the use of PROSAC [16] as an alternative to RANSAC in future works, as it is generally much faster than RANSAC and should produce similar results if the corresponding corners are thoughtfully ordered.

Finally, the data showed that in many cases, ArDVRC had slightly higher corner estimation errors than ArUco. This difference in error is generally one or two pixels, and is small enough that in most cases it will not be noticeable when drawing the outline of the marker onto an image. However, more research is needed to determine what effect, if any, these slightly increased error values have on the estimated marker pose, and therefore the projection of AR content.

VII. Conclusion

In this paper we have proposed a novel method of square fiducial marker detection and localization by finding diagonal marker corners using corner classification and direction vector extrapolation. We have demonstrated this method's ability to outperform popular square fiducial marker detection methods in cases of frequent partial edge occlusions, and have demonstrated detection rates up to 2.48x that of thresholding-based detection methods for partial edge occlusions. This approach can be used to help alleviate common problems with existing square fiducial detection and localization algorithms that frequently do not detect markers with partial edge occlusions, or that provide erroneous marker corner estimates under these conditions. Our algorithm works for single images or video sequences, is able to run in real time, and shows promise as an alternative method for marker-based AR to provide more seamless interactive AR experiences.

References

[1] 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, 2018.

[2] S. Garrido-Jurado, R. Munoz-Salinas, F. J. Madrid-Cuevas, and R. Medina-Carnicer, "Generation of fiducial marker dictionaries using mixed integer linear programming," *Pattern Recognition*, vol. 51, pp. 481–491, 2016.

[3] E. Olson, "Apriltag: A robust and flexible visual fiducial system," in *2011 IEEE International Conference on Robotics and Automation*. IEEE, 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)*. IEEE, 2016, pp. 4193–4198.

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

[6] F. J. Romero-Ramire, R. Munoz-Salinas, and R. Medina-Carnicer, "Fractal markers: a new approach for long-range marker pose estimation under occlusion," *IEEE Access*, vol. 7, pp. 169 908–169 919, 2019.

[7] H. C. Kam, Y. K. Yu, and K. H. Wong, "An improvement on aruco marker for pose tracking using kalman filter," in *2018 19th IEEE/ACIS International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing (SNPD)*. IEEE, 2018, pp. 65–69.

[8] F. J. Romero-Ramirez, R. Muñoz-Salinas, and R. Medina-Carnicer, "Tracking fiducial markers with discriminative correlation filters," *Image and Vision Computing*, p. 104094, 2020.

[9] R. Muñoz-Salinas and R. Medina-Carnicer, "Ucoslam: Simultaneous localization and mapping by fusion of keypoints and squared planar markers," *Pattern Recognition*, p. 107193, 2020.

[10] M. A. Fischler and R. C. Bolles, "Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography," *Communications of the ACM*, vol. 24, no. 6, pp. 381–395, 1981.

[11] B. Schoun, H. Oagaz, and M.-H. Choi, "Ardvrc," 2021. [Online]. Available: https://github.com/ucd-graphics-vr-lab/ArDVRC

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

[13] C. G. Harris, M. Stephens *et al.*, "A combined corner and edge detector." in *Alvey vision conference*, vol. 15, no. 50. Citeseer, 1988, pp. 10–5244.

[14] S. van der Walt, J. L. Schönberger, J. Nunez-Iglesias, F. Boulogne, J. D. Warner, N. Yager, E. Gouillart, T. Yu, and the scikit-image contributors, "scikit-image: image processing in Python," *PeerJ*, vol. 2, p. e453, 6 2014. [Online]. Available: https://doi.org/10.7717/peerj.453

[15] G. Bradski, "The opencv library," *Dr. Dobb's Journal of Software Tools*, 2000.

[16] O. Chum and J. Matas, "Matching with prosac-progressive sample consensus," in *2005 IEEE computer society conference on computer vision and pattern recognition (CVPR'05)*, vol. 1. IEEE, 2005, pp. 220–226.