Passive Pose Estimation of ArUco Markers with Optic Flow

William Carty, Samuel Mahle, Paul Gibert
Fall 2020 CS 4476 Computer Vision: Class Project
Georgia Tech

Abstract

The recent boom in AR/VR applications is giving way to many unique tracking problems in computer vision, particularly those that are framed around passive components. This project analyzes the effectivity of enhancing ArUco marker tracking using optic flow, based on a passive stylus tracking application proposed by Oculus researchers. We show that ArUco combined with optic flow can be used to detect more ArUco markers throughout frames than just with ArUco alone.

Teaser Figure

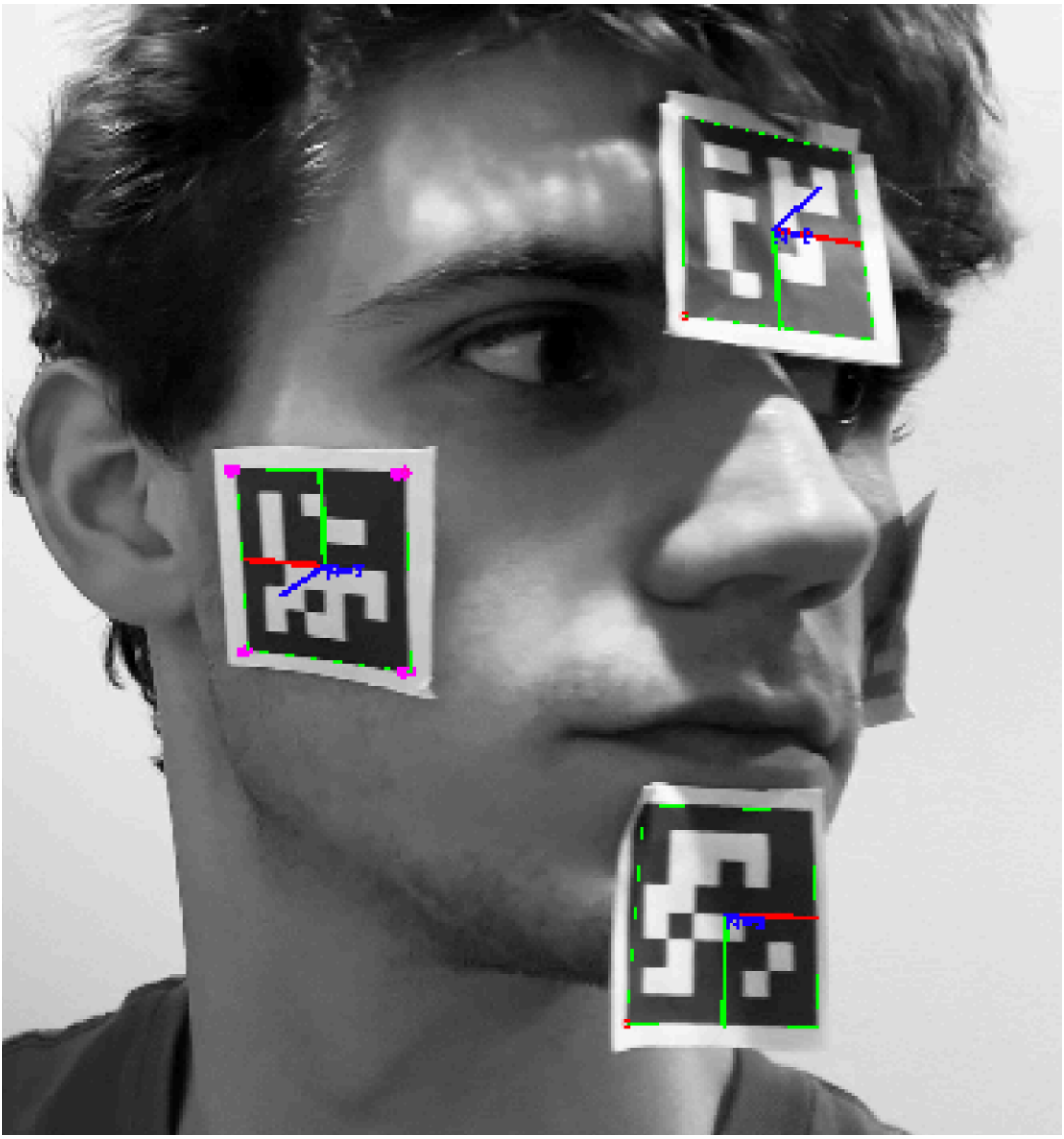

Figure 0. Proposed tracking system detecting ArUco marker poses on the surface of a human face.

Introduction

A growing demand in AR/VR applications has pushed researchers in computer vision fields to explore many new problems, particularly object tracking. It is common for AR/VR applications to use active components such as LED arrays [2][4][6] to improve the trackability of an object, however, battery power restrictions have resulted in many researchers exploring passive systems. In 2020, Oculus experimented with hand tracking in place of battery powered controllers and officially implemented the feature in a release of their Oculus Quest 2 virtual reality headset [8]. In addition, Oculus published a tracking system that combined ArUco markers with optic flow and dense pose alignment to estimate the pose of a passive pen stylus [5]. This project builds off Oculus's work in the former by bringing the ArUco and optic flow stages into a Python environment instead of C++ and demonstrating the system on arbitrary objects other than a pen.

Approach

The system combines ArUco marker tracking using the ArUco python wrappers shipped with OpenCV and OpenCV's pyramidal LK optical flow tracker [2] to track ArUco 2D binary markers in video frame. The tracker is implemented in two stages. First, ArUco is used to estimate

the corners of any markers in the frame. Second, we use optic flow to attempt to locate any markers that ArUco failed to detect. The stages are described in depth below.

ArUco Marker Tracking

ArUco is a popular marker tracking application, known for its use and accurate tracking of 2D binary markers. To create a passive trackable object, we print, cutout, and attach any of the 6x6 ArUco markers from the set shown in Figure 2.

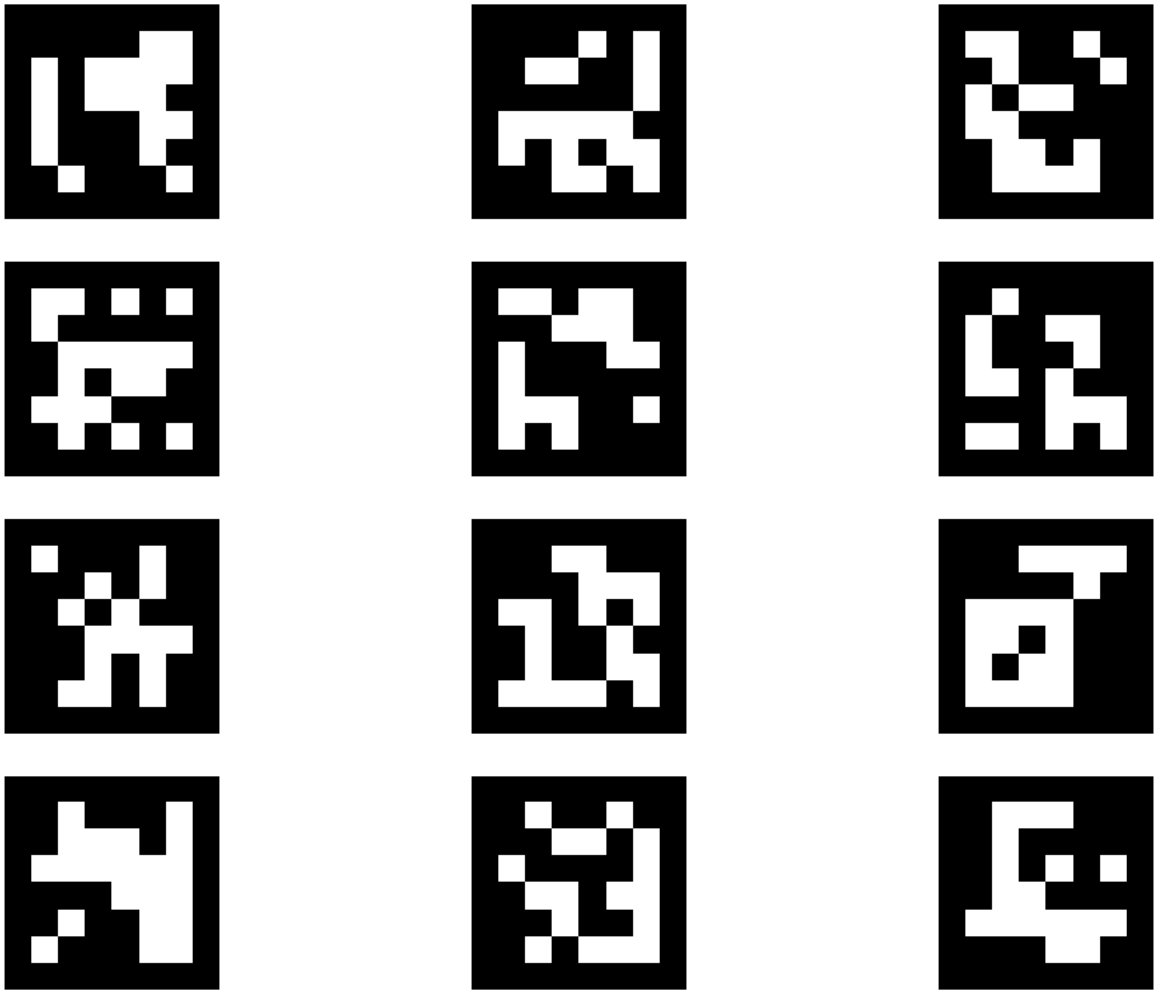

Figure 2. Set of 6x6 ArUco markers recognizable by our tracker

The first stage of the tracker uses ArUco to extract the image coordinate corners of any detectable markers in the frame and reads the 2D barcode to determine the id of each detected marker. This is accomplished by thresholding the image, extracting all four-sided polygons, and uses an intensity voting system to determine or reject the 2D barcode [1]. The

benefits of choosing ArUco as the base of our tracking system are its speed and simplicity. Oculus researchers demonstrated that the library can successfully detect markers in real-time [5] and OpenCV comes with the ArUco built into its python package.

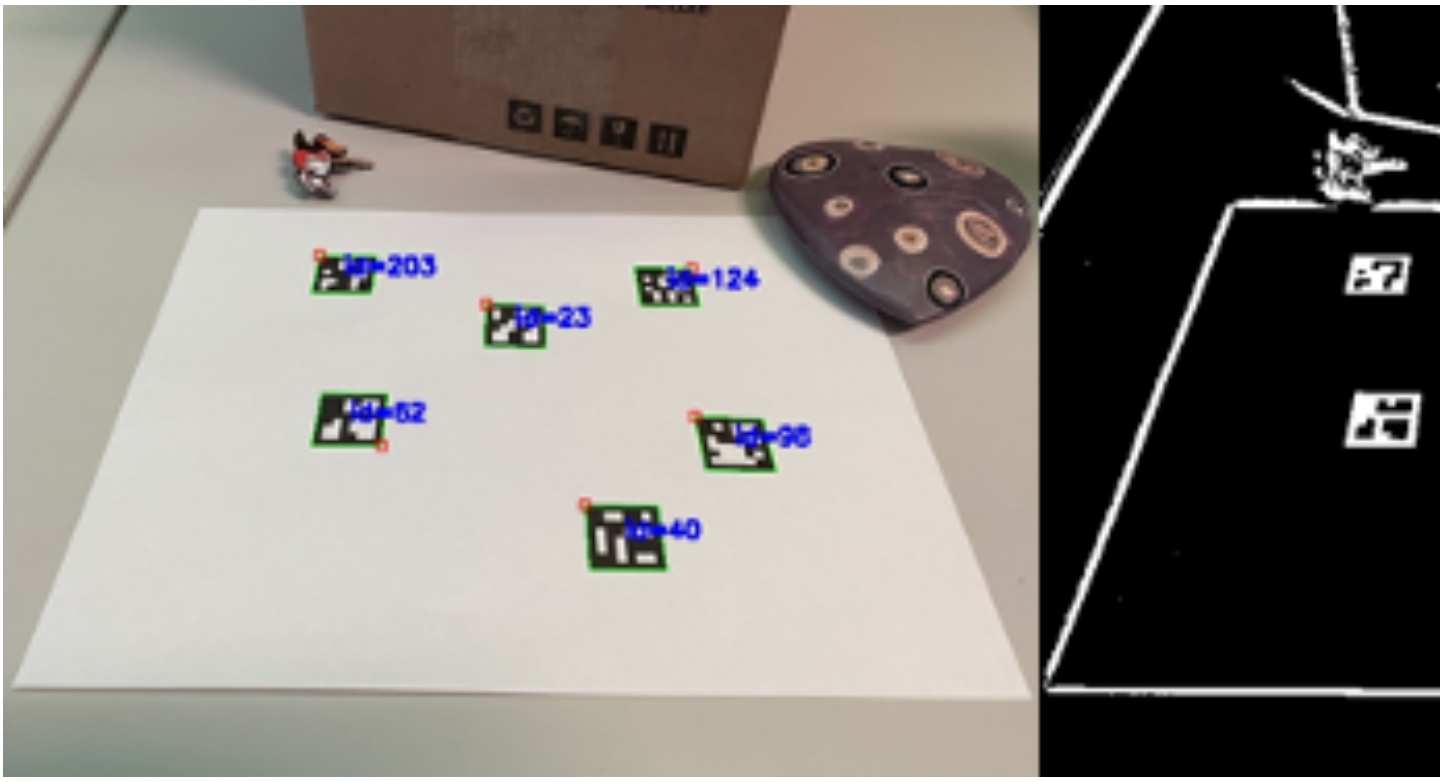

Figure 3. ArUco performing the thresholding step on a scene with 6 markers. The thresholding step is shown on the right and the identified markers are highlighted on the left. This image is taken from the OpenCV ArUco documentation [1].

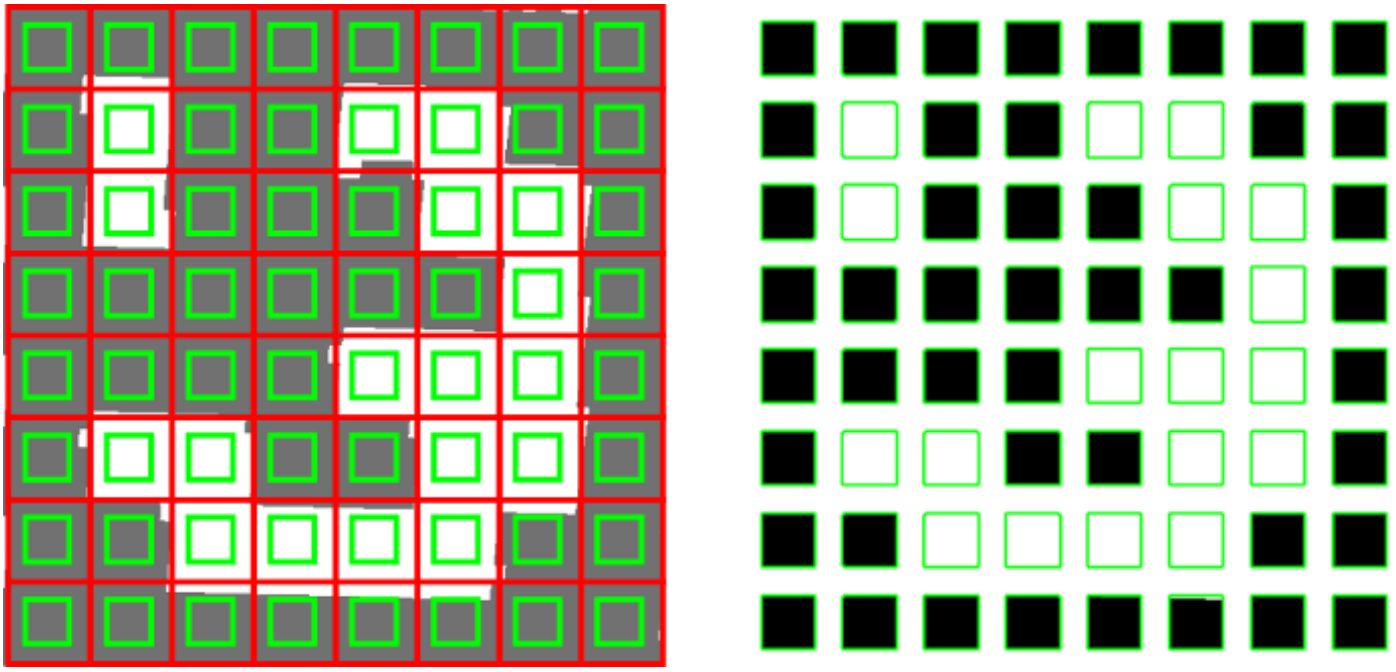

Figure 4. ArUco extracts four-sided polygons from the frame and splits each shape into the expected grid size. A voting system is used to determine the value of each cell to identify the marker. This image is taken from the OpenCV ArUco documentation [1].

Inter-frame Corner Tracking with Optic Flow

As noted in Oculus's pen tracking application, motion blur often results in ArUco failing to detect all markers in each frame [5]. Thus, we introduce the second stage of the system which uses OpenCV's pyramidal LK optical flow tracker to attempt to make up for the loss [3].

We chose this optic flow algorithm because it had a fast implementation in OpenCV and was successful employed in Oculus's pen tracking application in tandem with ArUco [5]. Using optic flow, we detect marker corners where ArUco failed by storing the detected markers from the previous frame. Let $\boldsymbol{P}$ be the set of markers detected in the previous frame and $\boldsymbol{C}$ the set of markers detected in the current frame by ArUco. For each marker $\boldsymbol{m}$ where $\boldsymbol{m} \in \boldsymbol{P}$ but $\boldsymbol{m} \notin \boldsymbol{C}$, get the points of each corner of $\boldsymbol{m}$ from the previous frame and compute the flow at each corner to determine its new location in the current frame. If this can be done successfully for all four corners of $\boldsymbol{m}$, then we have detected an additional marker in the current frame.

Approximate Pose Estimation

After detected as many markers as possible post ArUco and optic flow steps, we turn to ArUco one final time to extract the pose of each marker and complete the tracker. In a previous step, we use ChArUco checkerboards, shown in Figure 5, to calibrate an iPhone camera. These intrinsic and extrinsic parameters along with the corners of each marker are passed into an ArUco pose estimation method which gives us the six degrees of freedom orientation of each marker in space.

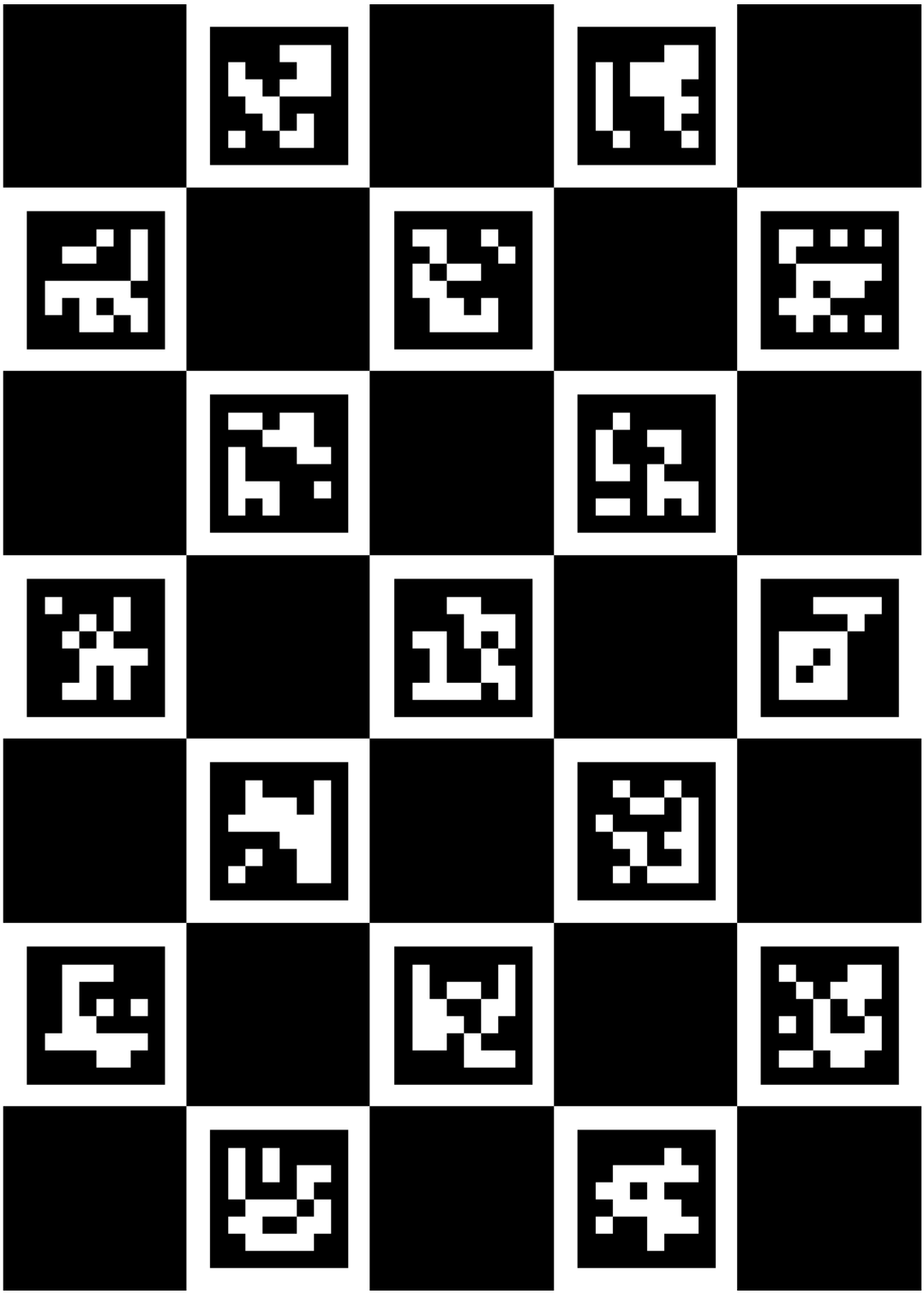

Figure 5. The ChArUco board used to calibrate the iPhone camera. The “Ch” stands for checkerboard and is a

popular ArUco technique for camera calibration that is more accurate than the common checkerboard camera calibration method.

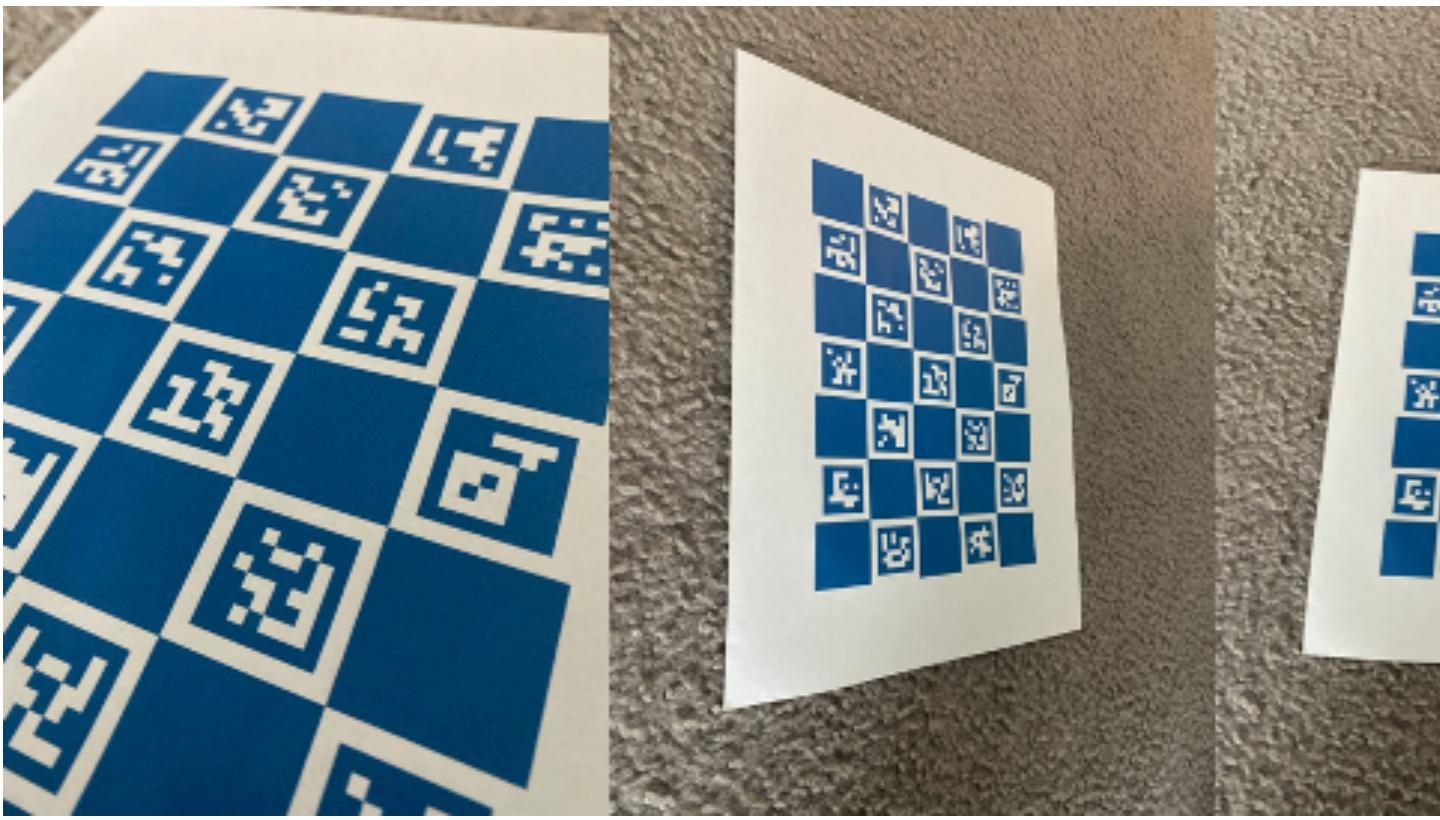

Figure 6. The ChArUco board was photographed at twelve different angles, including these four, for use in ArUco's camera calibration methods. The boards were printed out as dark blue instead of black because our lab printer ran out of black ink. Dark blue proved satisfactory in calibrating the camera accurately.

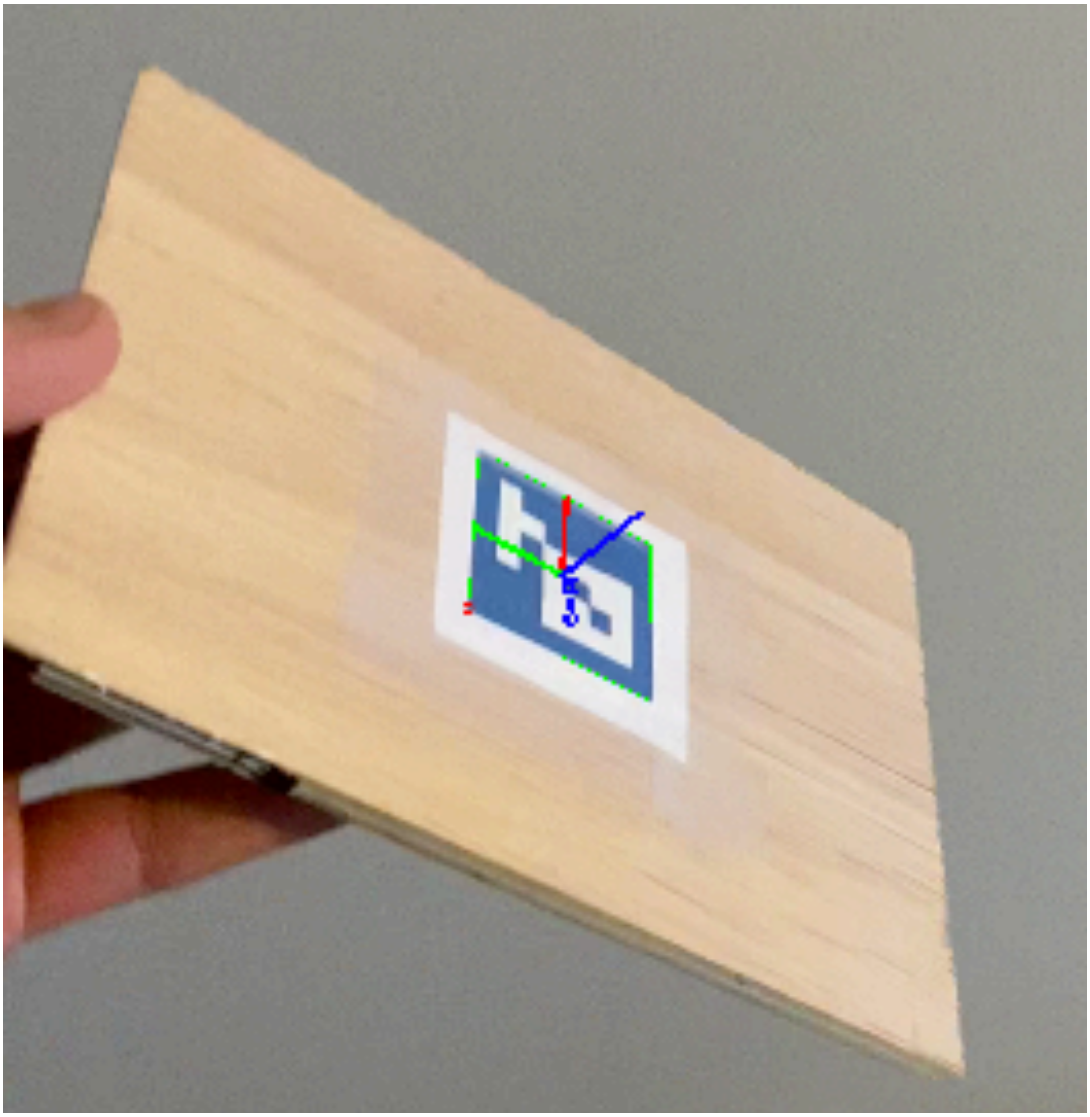

Figure 7. Our tracker estimating the pose of an ArUco marker taped to a wooden board. The marker is outlined with a green frame and x, y, and z axes are drawn at its origin in 3D space indicating its pose.

We faced a few minor obstacles as we designed and developed our system. The primary obstacle was camera calibration. We attempted a traditional checkerboard approach, but our printer ran out of black ink and OpenCV failed to detect any other color. This served as our motivation for using a ChArUco board instead which proved detectable in other colors besides black. Note that all markers in the experiment were also printed in dark blue for the same reason.

Experiments and Results

To test our tracker, we designed and performed two experiments where markers attached to a brush were tracked in space. The brush construction is illustrated in Figure 8. The default or recommended parameters were used to configure ArUco's marker detection and OpenCV's optic flow algorithms for these experiments.

Figure 8. The brush used in the tracker experiments. Two ArUco markers are taped to its back.

Experiment 1: Translation

In the first experiment, we filmed 17 videos of the brush moving between a start and finish line at different speeds (see Figure 9). The brush was moved by hand with the markers facing the camera. This was

to capture the effectiveness of the tracker at detecting translation. We trimmed each video so that the brush is never outside the start and finish lines and moved the brush at a constant speed. We counted the total number of frames of each video and used this to determine the speed of the brush (less frames means the brush was moving faster). We measured two metrics: the total frames where at least one marker was detected and the total markers detected in the video (up to double the total frames in the video with two markers). We attempted to track the markers once using only ArUco and a second time with ArUco and optic flow. This allowed us to determine the effectiveness of combining optic flow with ArUco. The results of this experiment are illustrated in Figure 10.

Figure 9. Moving the brush between the start and finish lines in experiment 1. Detected markers are highlighted in green with axes drawn on them. The corners of the right marker are highlighted in pink to indicate that that marker was detected using optic flow.

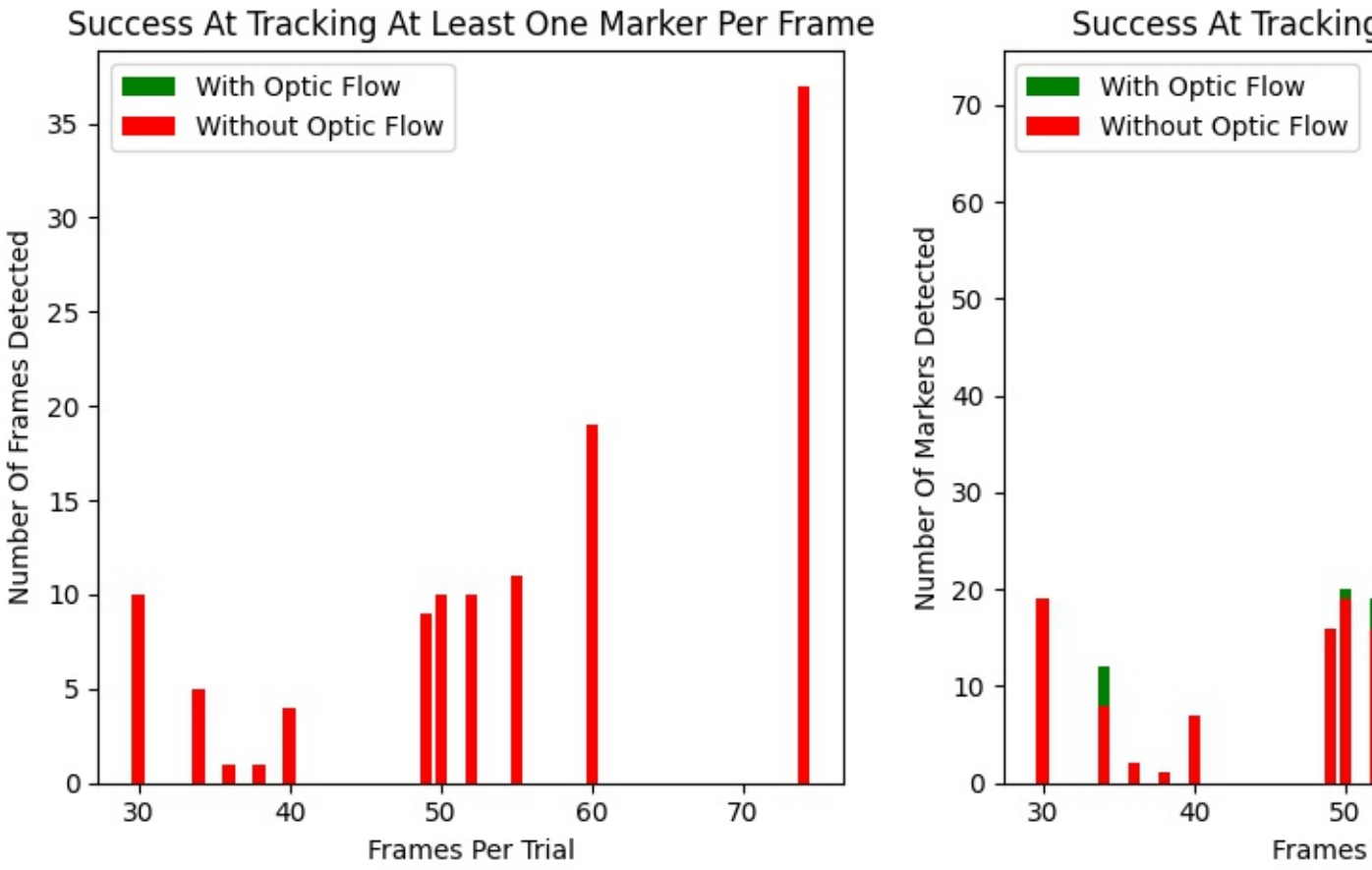

Figure 10. The results of the translation experiment. The left chart plots the number of frames where at least one marker was detected while the right plot shows the total number of markers detected.

As seen in Figure 10, optic flow was not beneficial to the tracker in finding at least one marker per frame. However, the data shows that optic flow does allow the tracker to pick up more markers per video. In the longer

videos, the use of optic flow permits the tracker to detect an additional ~20 markers in the 70+ frame video and an additional ~5 markers in the 60 frame video which is a notable improvement.

Experiment 2: Rotation

The second experiment closely mimics the first. We use the same brush and markers from the first experiment, but instead of translating the object between two points, we rotate it 180 degrees around its handle and away from the camera and then 180 degrees back (see Figure 11). 20 videos were taken for this experiment. The brush is rotated by hand at constant speed, and the total number of frames per video is assumed to indicate the angular speed of the object. We use the same two metrics from the first experiment to determine the effectiveness of adding optic flow to the tracking pipeline. The results of this experiment are shown in Figure 12.

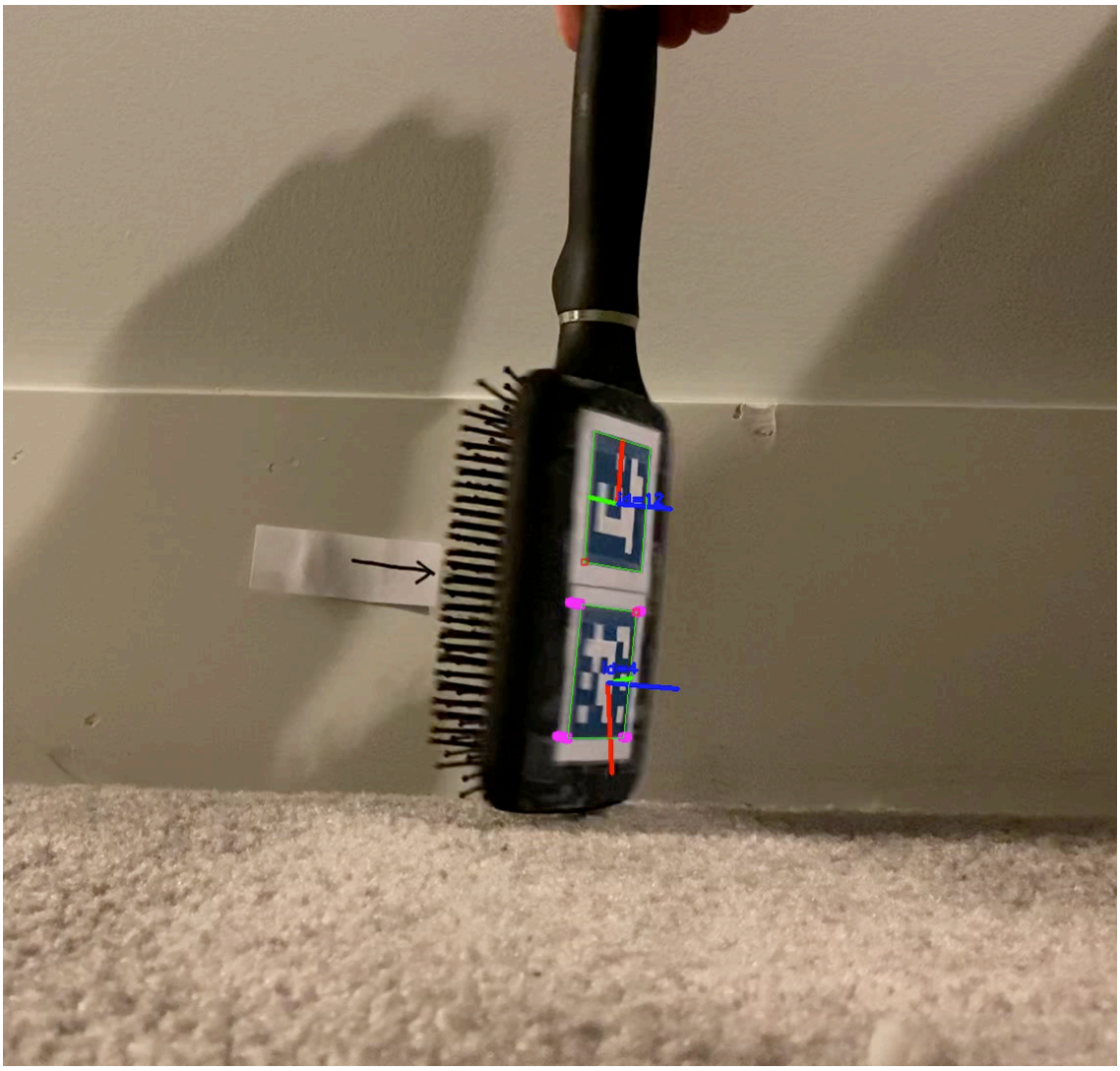

Figure 11. Rotating the brush 180 degrees back and forth. Detected markers are highlighted in green with axes drawn on them. The corners of the right marker are highlighted in pink to indicate that that marker was detected using optic flow.

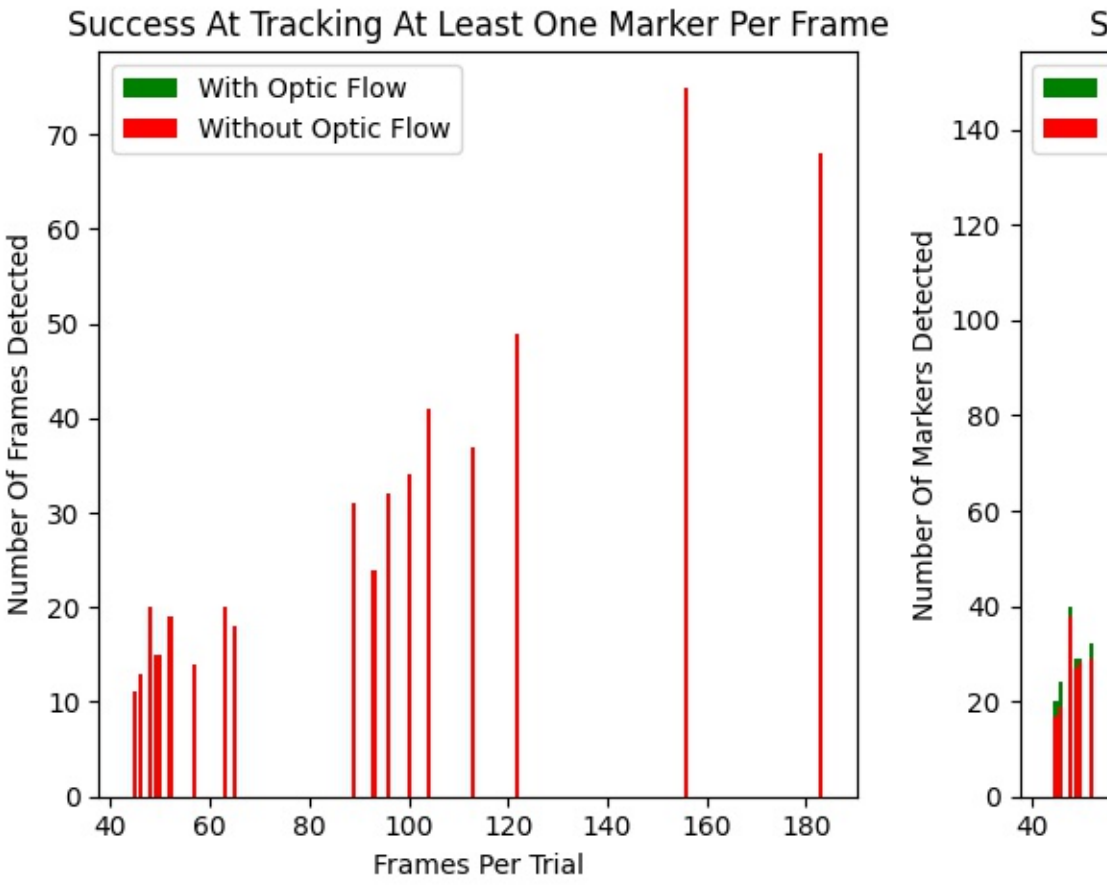

Figure 12. The results of the rotation experiment. The left chart plots the number of frames where at least one marker was detected while the right plot shows the number of markers detected total.

We saw no benefit to adding optic flow to the tracking pipeline in detecting at least one marker per frame. However, we again see benefit when counting the total markers detected. Longer videos, where the object is rotating slower, show an increase in the number of markers optic flow is able to help detect. The ~155 frame video had as many as

~15 extra markers found during its duration, while other videos had 5-8 extra markers detected according to the right plot in Figure 12.

Trends

Both experiments demonstrate that adding optic flow to the tracking pipeline allows the tracker to detect more markers. This was expected as it was demonstrated in the Oculus paper this system is based on [5]. However, we unexpectedly didn't grab any extra frames as a result of adding optic flow. We suspect this is due to low light levels during filming. We also could have implemented parts of the Oculus paper that account more for motion blur using the velocity and acceleration of the pose to help the tracker capture more frames. However, the Oculus paper did note that, "Unfortunately, the initial pose p 0 computed using PnP is too jittery to use in tracking the pen tip" [5] which was observed in the tracking videos and likely the result of not capturing extra frames. In general, the data shows that for slow translations and rotations of the object, adding optic flow to the tracking pipeline can result in a notably higher rate of marker detection.

Qualitative Results

Figure 13. A coffee mug with three markers attached to it. The highlighted markers have been successfully detected and pose drawn. The lower marker was found with optic flow as indicated by the glowing pink corners. We see that the more rotated right marker was unsuccessfully detected.

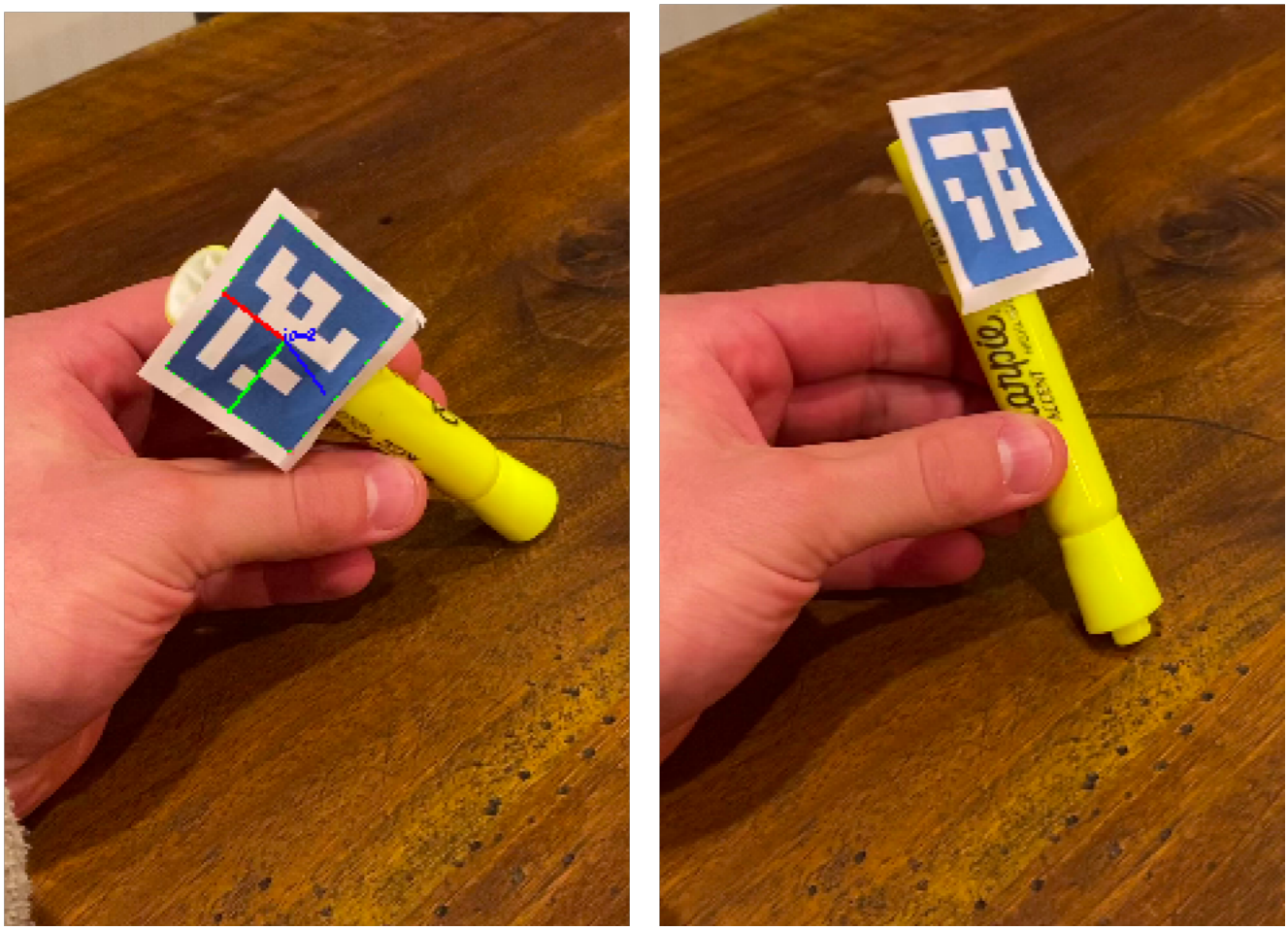

Figure 14. A success case (left) and failure case (right) of tracking a marker affixed to the end of a highlighter. These images demonstrate that markers angled away from the camera are harder to track.

Figure 15. Extracting the pose from two markers placed on the same flat surface. This demonstrates how cleverly placed markers can be used to determine the pose of arbitrary objects, in this case the top of the box.

Figure 16. An example of tracking markers attached to an oddly shaped object. We see again that markers angled away from the camera are harder to track versus those that are head on.

Conclusion and Future Work

This paper shows that optic flow can be used to enhance ArUco passive marker tracking on arbitrary objects. This is significant as there is a growing demand for passive tracking application in AR/VR related

fields. The system described in this paper is based off the Oculus C++ implementation of a passive pen stylus tracker [5]. Future work could include further implementing this paper in Python and continuing to expand the model to other arbitrary objects in addition to those proposed in the paper. We also believe the tracking system proposed by Oculus and partially implemented in this paper would significantly benefit from a motion blur elimination step. In practice, motion blur proved to significantly detract from the accuracy of the tracking system and there are several proposed methods for removal [9].

References

[1] Detection of ArUco Markers. OpenCV Documentation, https://docs.opencv.org/master/d5/dae/tutorial_ArUco_detection.html

[2] Jaehyun Han, Seongkook Heo, Hyong-Euk Lee, and Geehyuk Lee. 2014. The IrPen: A 6-DOF Pen for Interaction with Tablet Computers. IEEE Computer Graphics and Applications 34, 3 (2014), 22–29.

[3] Jean-Yves Bouguet. 2001. Pyramidal Implementation of the Lucas Kanade Feature Tracker: Description of the Algorithm. Intel Corporation 5, 1-10 (2001), 4.

[4] Peter Rubin, “Inside Oculus’ Quest to Design an Invisible VR Controller,” Wired, December 6, 2016, https://www.wired.com/2016/12/oculus-touch-design/

[5] Po-Chen Wu, Robert Wang, Kenrick Kin, Christopher Twigg, Shangchen Han, Ming-Hsuan Yang, and Shao-Yi Chien. 2017. DodecaPen: Accurate 6DoF Tracking of a Passive Stylus. ACM Symposium on User Interface Software and Technology.

[6] Seongkook Heo, Jaehyun Han, Sangwon Choi, Seunghwan Lee, Geehyuk Lee, Hyong-Euk Lee, SangHyun Kim, Won-Chul Bang, DoKyoon Kim, and ChangYeong Kim. 2011. IrCube Tracker: An Optical 6DOF Tracker based on LED Directivity. In Proceedings of ACM Symposium on User Interface Software and Technology.

[7] Sergio Garrido-Jurado, Rafael Muñoz-Salinas, Francisco José Madrid-Cuevas, and Manuel Jesús Marín-Jiménez. 2014. Automatic Generation and Detection of Highly Reliable Fiducial Markers Under Occlusion. Pattern Recognition 47, 6 (2014), 2280–2292.

[8] Shangchen Han, Mark Richardson, and Robert Wang, “Making Technology Feel Natural,” Tech@Facebook. October, 21, 2020, https://tech.fb.com/making-technology-feel-natural/

[9] Shuang Zhang, Ada Zhen, Robert L. Stevenson. 2019. Deep Motion Blur Removal Using Noisy/Blurry Image Pairs.