INTRODUCTION TO COMPUTER VISION

ARUCO TAGS FOR DYNAMIC ESTIMATION AND TRAJECTORY PLANNING

May 16, 2022

Anthony Olvera and Xiangyu Yu

Colorado School of Mines

INTRODUCTION

For our project we investigated using aruco markers for pose estimation to calculate certain dynamic estimations of a moving object. An aruco marker is a 2 dimensional square grid of either white of black cells. These tags give a unique identification to the object they are attached to. An nXn tag gives a total of 2^{n^2} unique ids. For example a 4X4 tag size has 16,536 unique tags. In addition, when the tags size and certain parameters of our camera are known we can calculate its position and orientation or pose through techniques in image projection calculations. We developed a custom aruco library for multiple different objects. This library contains useful information about the object such as its size, shape and mass. Given these properties as well as the pose of the object and our cameras frame-rate we estimate object speed, acceleration, momentum and force. We then use augmented reality or AR to monitor and visualize these metrics as well as give important visual indications when the objects kinematic behavior is approaching a safety threshold. We overlay the object outline to give the user a visual representation of the objects size and shape when it may be partially or completely occluded. We overlay an orthophoto of the object ID along with its properties onto the tag using a homography. When the object velocity exceeds a certain threshold we replace this image with a warning signal. We vary the color of the object from green to red using a color map to give a user a visual means of monitoring and safely manipulating an object. Lastly we project the objects velocity and momentum vector as well as its trajectory onto the video stream giving the user a visualization of such intangible mechanical concepts and allow them to monitor the objects movement. We believe such a system could have various applications in robotics and systems design such as obeying acceleration limits when transporting fragile materials, visually monitoring the tilt of an object or safely halting a system that is moving too fast.

MOTIVATION

In recent years the use of robotics and autonomous systems has been growing rapidly as technology continues to advance. Due to this we have more warehouse and factory machinery responsible for transporting, delivering and or manipulating objects. Because these objects can range widely in the required care which must be taking when handling them, it is important that machines have a means of understanding these requirements. For example a machine moving moving an open liquid container needs to pay special attention to acceleration and tilt whereas moving and empty box does not. Machine learning algorithms are limited in what they can classify and they cannot accurately predict certain physical properties of objects such as weight. Our system could be useful on a plethora of autonomous equipment, so long as the item can be tagged with an aruco marker, then a machine can be certain of these physical properties. This in turn will allow a machine to appropriately handle the object during what ever process it is a part of. Ultimately this will lead to increased safety for employees who must work near such machines, less accidental damage which results in large financial losses and increased efficiency as some objects may require minimal attention that can instead be devoted to more care deprived items.

However there are still many human workers in warehouse and factory industries that must transport and manipulate objects, merchandise and cargo manually. Many workers also must work collaboratively with robots or machinery. The augmented reality aspect of this project is intended to benefit a human worker. Many groups that work in complex or dangerous environments may benefit from this technology as it can visually guide them in appropriately handling certain cargo by using an AR headset or a computer or tablet. Depending on the setting this will lead to less accidents which may result in loss of employment of even loss of life. In fact, due to an increase of safety and efficiency these workers can expect easier, safer and more lucrative work throughout their careers.

PREVIOUS WORK

A lot of research and practical approaches investigated a better application of aruco markers int object tracking, movement measurement, detection, gesture recognition, visual mapping and localization, and estimation of exerted contact forces. [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12].

Kam et al. (2018) [7] proposed a pose tracking method using a Kalman filter, which prevent the estimation from being blocked when part of the aruco code is occluded. Even when the aruco code has been blocked temporarily, the pose is still trackable in resistance to noise. Chan et al. (2018) [4] designed a hand-tracking glove which is able to track the pose of the hand and the motion of the five fingers. Aruco markers were used in their research for obtaining the pose of the glove and the Kalman filter helps with the pose stablization. Although IMU (inertial measurement unit) can be adopted for an accurate measurement at a higher sampling rate of translations and rotations, vision-based aruco marker is necessary for absolute measurement. Asselin et al. (2018) [1] pointed out that the optical tracking system always has accuracy issues with aruco markers within a small tracking volume which is highly possible in applications such as computer assisted surgery or surgical training. Their algorithm and system couldn't improve the aruco marker as a tool, but at least stating the potential error associated with this approach. Aruco markers were also used to train the images for pose estimation Crivellaro et al. (2015) [5]. They presented an approach to partially represent the object by detecting some parts and then predict the 3D pose in the form of 2D projections, where aruco markers provides an accurate reference in the convolutional-neural-network training process. Myhre and Egeland (2016) [8] used visual tracking for crane load collision detection. The dynamic model in their approach needs a measurement of load movement. A camera is rigidly attached to the crane and the camera will locate the crane load in the images with an aruco marker attached on the load. The approach used in this paper is exactly the same as our method.

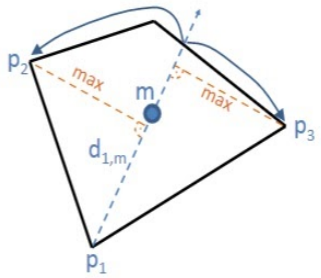
Meanwhile, aruco markers have been applied to navigate drones by their pose estimation functionality. In Sani and Karimian (2017) [9], drones approach an aruco marker using IMU sensors first and then by estimating the distance and pose of the aruco marker, land on the aruco marker to model a navigation process. Aruco markers are also important tools in visual mapping and localization. Bacik et al. (2017) [3] also used aruco markers and their pose estimation to navigate among multiple markers and for real-time mapping and localization. Xavier et al. (2017) [11] provided an algorithm of reconstructing a 3D scene using artificial markers. The markers are used to compute a map of poses, and vertices and edges are formed by the relative transforms. Tordal and Hovland (2017) [10] tracked the relative motion between two vessels by placing aruco cubes on one vessel and a camera on the other. Pose estimation was applied to track the relative motion with the help of a Kalman filter and sensor fusion algorithms. Accuracy within 31mm was achieved in their work. Elangovan et al. (2019) [6] decoded the contact forces exerted by adaptive hands using IMU and aruco markers. In order to provide robust and stable grasp from adaptive hand, a machine learning model was used to capture then train the complex post-contact reconfiguration input from the sensors and the vision-based markers.

In these applications, aruco markers have been applied mainly for estimating poses for localization, navigation, and motion tracking. We are using the aruco markers in a similar way where we will compute the objects dynamic properties and estimate the approximate forces or threshold for stabilization. The previous research provides novel ideas for us to explore what can be done using our current tool, such as the two vessels and drone navigation cases. The generated data can be used for force estimation as well based on [6] although they employed a machine learning model.

TECHNICAL APPROACH

Aruco marker generation and detection

An aruco marker (figure 1) is a square grid of black and white cells that make up a unique ID. These tags can be attached to objects and detected using computer vision techniques. An nXn tag makes up a total of 2^{n^2} unique IDs. These IDs then correspond to data stored in a dictionary for our specific application. We used 4X4 tags which gives a total of 16,536 unique IDs which correspond to 3 unique objects in our custom aruco library. These objects are a square box with mass of 1kg. A pyramid with a mass of 0.5kg and an rectangle with a mass of 0.1kg. To detect the markers we first threshold each image in our video stream using otsu's method for global thresholding which is done by minimizing $\sigma_W^2 = P_1\sigma_1^2 + P_2\sigma_2^2$ where $P_1 = \sum_{i=0}^{k} p_i$ and $P_2 = \sum_{i=k+1}^{L-1} p_i$ which results in a binary image as in figure 2. We then find contours around all the black regions by tracing the outline of all black connected components see figure 3. Once all black regions are detected we determine if it is a quadrilateral using the following algorithm



Algorithm 1: Finding Quadrilaterals

Start at a and walk contour, place p_1 an max distance;
Compute centroid m;
Find corners p_2, p_3 on either side of $d_{1,m} = (p_2, p_3)$;
Find farthest point p_4;
Determine orientation from black corner at $s_1 = (p_1 + m)/2$;

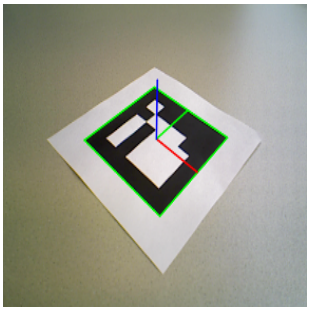


Figure 1: Aruco Marker

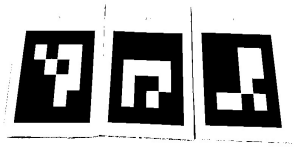


Figure 2: Image after OTSU Thresholding

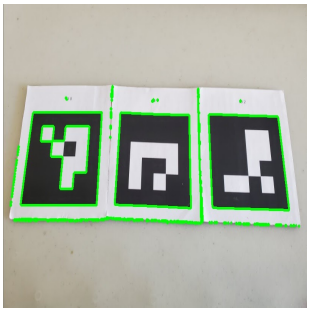


Figure 3: Contours around Black Regions

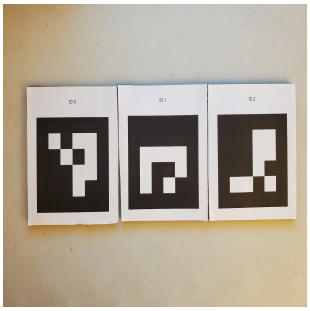


Figure 4: Orthophoto

Once black squares are detected we transform the candidate marker to an orthophoto to determine if it is a valid marker. This is done using a homography. A homography is a 2D-2D transform that maps points from the projection of one plane to that of another. By doing so we transform the image of the tag to an orthophoto. An orthophoto is a photo that has been transformed as if we were looking directly at it (figure 4). Given a point p in camera ones coordinate system the homography from frame 1 to 2 is calculated as follows.

$$\tilde{x_1} = \mathrm{K} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} \mathrm{c}_1 X \\ \mathrm{c}_1 Y \\ \mathrm{c}_1 Z \\ 1 \end{bmatrix} = \mathrm{K} \begin{bmatrix} \mathrm{c}_1 X \\ \mathrm{c}_1 Y \\ \mathrm{c}_1 Z \end{bmatrix}$$

$$\text{also} \begin{bmatrix} \mathrm{c}_1 X \\ \mathrm{c}_1 Y \\ \mathrm{c}_1 Z \end{bmatrix} = K^{-1} \tilde{x_1}$$

$$\begin{bmatrix} c_1X \\ c_1Y \\ c_1Z \\ 1 \end{bmatrix} = \begin{bmatrix} {}^{c_2}_{c_1}\mathbf{R} & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} c_1X \\ c_1Y \\ c_1Z \\ 1 \end{bmatrix} \text{ or } \begin{bmatrix} c_1X \\ c_1Y \\ c_1Z \end{bmatrix} = {}^{c_2}_{c_1}\mathbf{R} \begin{bmatrix} c_1X \\ c_1Y \\ c_1Z \end{bmatrix}$$

$$\text{then, } \tilde{x_2} = \mathrm{K} \begin{bmatrix} c_1X \\ c_1Y \\ c_1Z \end{bmatrix} = K\,{}^{c_2}_{c_1}\mathbf{R}K^{-1}\tilde{x_1}$$

We now parse the tag and check if it matches any objects inside our aruco library. This is done in four separate directions as opposed to only left to right, top to bottom to ensure accuracy in tag detection by integrating redundancy. A four by four tag is represented as a 2X4 matrix of 8-bit unsigned integers. The notation I am using to represent the parse direction is as follows. Left to right, top to bottom is represented as LRTB. Then the data is represented as

$$\begin{bmatrix} LRTB & LRTB & TBRL & TBRL \\ RLBT & RLBT & BTLR & BTLR \end{bmatrix}$$

For example the following tag pattern would be represented as the following matrix.

$$\begin{bmatrix} 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix} \rightarrow \begin{bmatrix} 181 & 50 & 235 & 72 \\ 35 & 173 & 18 & 215 \end{bmatrix}$$

Once we have detected an aruco marker we need to calculate its position and orientation with respect to our camera. This is done using another homography which maps the projection from the plane defined by the aruco tag to our cameras image plane. We can solve for H given the equation below as we have 4-point correspondences between the image and marker. We have h such that

$$x^{`} = Hx \begin{bmatrix} \mathrm{ku}^{`} \\ \mathrm{kv}^{`} \\ \mathrm{k} \end{bmatrix} = \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix} \begin{bmatrix} u \\ v \\ 1 \end{bmatrix}$$

and

$$\tilde{p} = MP = K(r_1 r_2 r_3 t) \begin{bmatrix} X \\ Y \\ 0 \\ 1 \end{bmatrix} = K(r_1 r_2 r_3 t) \begin{bmatrix} X \\ Y \\ 1 \end{bmatrix}$$

where p is the projection point and r_1, r_2 and r_3 are the columns of R and this gives us the pose of the tag with respect to our camera.

Aruco marker dynamic properties

The pose estimation in the above section gives important information about our translation vector. The equation below shows the homography transformation matrix derived in the previous section:

$$(r_1 r_2 r_3 t) = {}^{A}_{B}H = \begin{bmatrix} & {}^{A}_{B}R & & {}^{A}t_{Borg} \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Among the elements in the matrix, ${}^{A}t_{Borg}$ is the translation vector from Bs coordinate system to As coordinate system as shown in Figure 5. In the camera coordinate, the translation vector provides the movement of point B with respect to point A. We can directly use this vector for computation of dynamic properties [10, 8]. However, a reference value is necessary to input into the computation. In our case, we use real aruco marker length as this reference value so that the calculated dynamic properties are also in this unit in terms of length. In addition to the translation vector, we can perform a RQ decomposition of the rotational matrix using OpenCVs library to obtain polar rotations. With the translation vector, we can use the coordinates in each frame to calculate the velocity and acceleration of a moving tag.

We can also get the polar rotation with respect to each axis from the decomposition so that a threshold can be set as a "dangerous" limit which should not be passed.

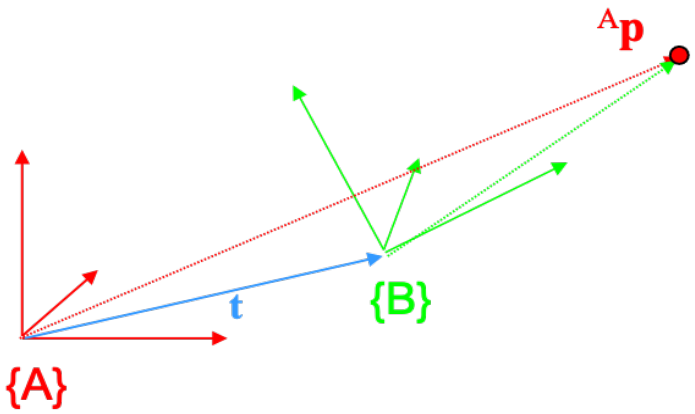


Figure 5: Translation vector from B to A coordinates

Meanwhile, we have attached an object to describe the unique aruco marker that we used, such as liquid container or fragile sculpture. Whenever there is a "dangerous" situation such as too much axis rotation or too high velocity, the container will become red on top and bottom. The trajectory of the aruco marker is also tracked and output to the video as a reference. As shown in 6, the red top/bottom, real time velocity, acceleration are output on the video screen.

Algorithm 2: Marker pose decomposition and dynamic property calculation

Detect aruco marker in the frame;
Estimate pose of the detected marker;
Decompose rotational vector into polar rotations;
Extract coordinate of tag center in camera;
Minus the previous frame coordinate and divided by second per frame

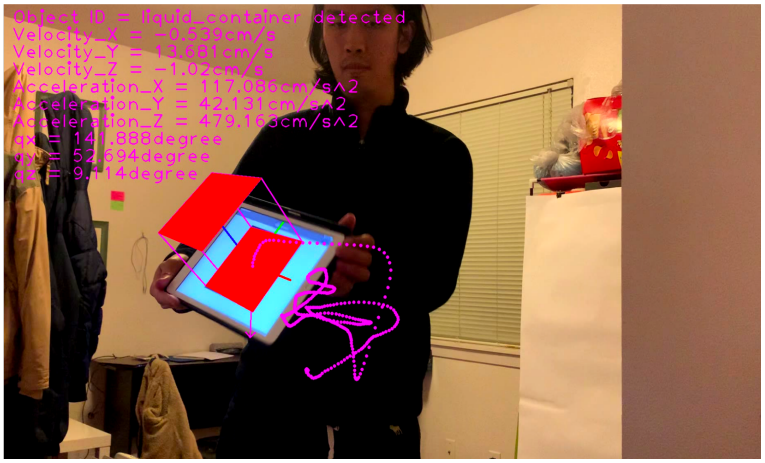


Figure 6: Sample frame of the output video

EXPERIMENTS AND RESULTS

Object encoding by aruco marker

In order to evaluate our solution we placed an aruco tag on an object of known mass and defined a speed limit 0.75mph. The object we used was a set of indoor security cameras in their original packaging. The packages weight was measured using a kitchen counter top scale and weighed 449 grams, 50 additional grams were added to reach our desired half kilogram by attach 20 pennies to the package which weigh 2.5 grams each. This give us a maximum momentum of $0.167 kgm/s$ to not be exceeded. We placed the object on a treadmill and recorded multiple videos at multiple speed with our camera with known intrinsic parameters. It is important to clarify we must assume the treadmills motor controller is accurate and the actual speed is what is indicated. We then read the videos in and evaluated the behavior of our code. We expect to see the video of the object moving at 0.5mph overlaid with object image and green data indicating we are within speed tolerance. Likewise we expect to see red indications and a warning overlay image when the object is traveling at 1 mph as it has exceeded it speed limit.

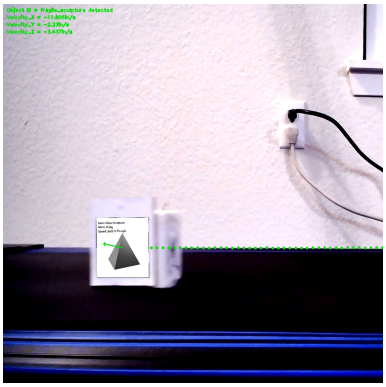


Figure 7: Object moving within speed limit

Figure 8: Object exceeding speed limit

Velocity measurement and benchmark

Using our program to compute the velocity of the package moving on the treadmill, we plotted out the velocity values at two velocities: 0.5 mph and 1 mph. The plots are shown as below in Figure 9. Although the pose estimation gives velocity data with a lot of noise, we can still capture the average velocity in these two cases in the plot. The unit in this figure is inches/sec and the value transformed into mph would be 0.55 mph and 1.13 mph which are very close to the treadmill velocity set up beforehand.

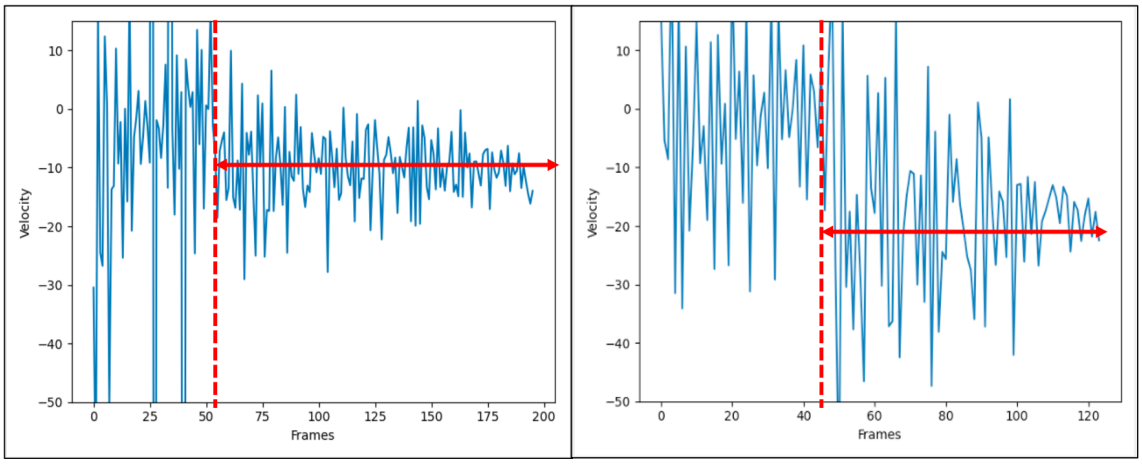


Figure 9: Compare treadmill velocity 0.5 mph and 1 mph

Using a handheld aruco tag, we also measure the objects movement along the same length in two ranges of time: 6 seconds and 12 seconds. We use a ruler as the reference and

move the object as steadily as we can, as shown in Figure 10. The velocity should be doubled in the shorter 6 second video. We also plotted out the real-time velocity in Figure 11: except the low value in the beginning and at the end, the measured velocity shows a relatively stable mean about 5 to 10 (cm/s in this case), which also matched our expectation.

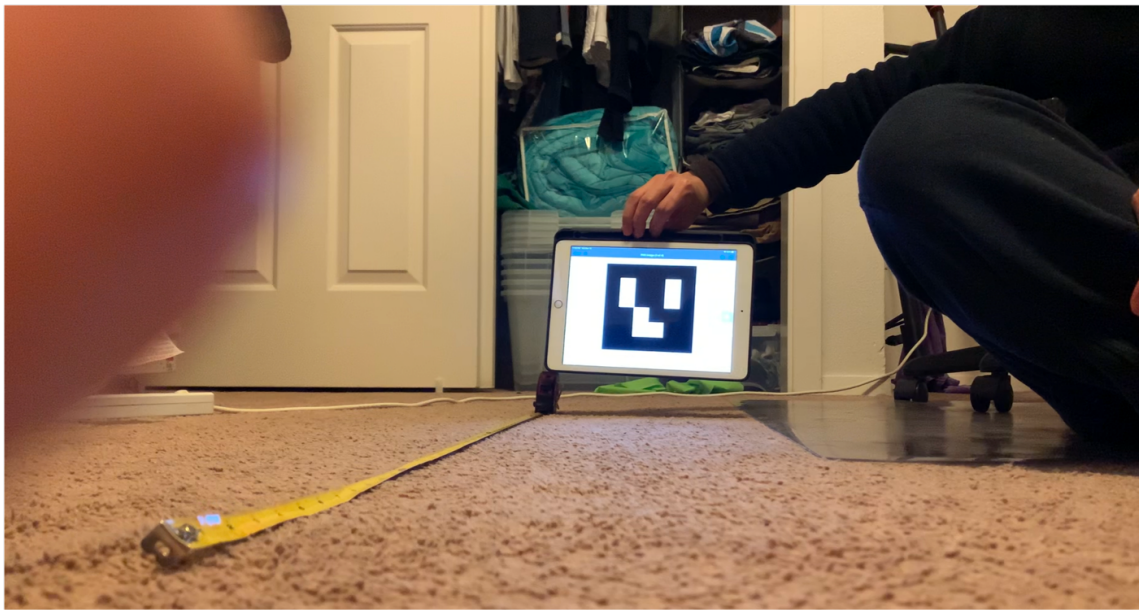


Figure 10: Handhold aruco tag moving with same length in different time span

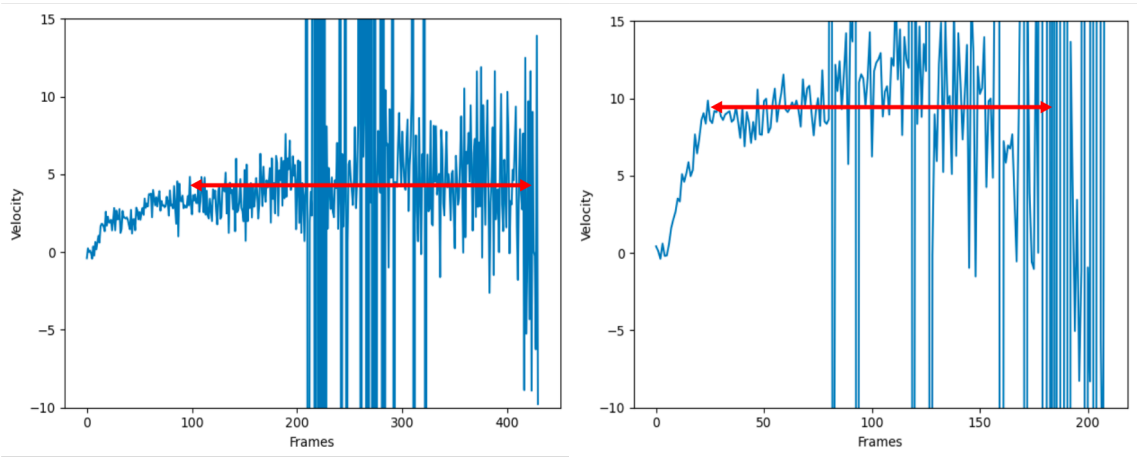


Figure 11: Handhold aruco tag velocity comparison

Table 1: Statistical result of four velocities benchmark

	mean	standard deviation	min	max
handheld velocity slow (cm/sec)	4.47	2.34	0.24	11.89
handheld velocity fast (cm/sec)	10.07	2.80	3.54	19.91
treadmill velocity 0.5 mph (8.8 in/sec)	8.80	3.71	0.19	14.99
treadmill velocity 1 mph (17.6 in/sec)	19.72	9.02	0.98	38.11

Statistical results

For the data in Figure 9 and Figure 11, we performed simple data cleaning process (which ideally should have been done by Kalman filtering) by removing the data out of the desired range. For example, in the first plot of Figure 11, we removed all velocity out of the rage $[0, 15]$ and then collect the mean, standard deviation and variance of the data. The results in table 1 show all the statistics for the four cases we have tested. It can be seen that the larger velocity tends to give higher standard deviation indicating an unstable measurement or pose estimation. We plotted out the distribution for the handheld low velocity case in Figure 12 and the plot shows an acceptable histogram with major contribution at 4-5 cm/sec. Based on the statistical result above, we have 4 true positives and accuracy can be treated as high as 100% although our experimental attempts are not enough to draw such a conclusion.

DISCUSSION

Achievements

Given that we can keep a running average of our velocity estimations we can accurately predict the speed, momentum, acceleration and force of an aruco tagged object given that we know its mass. Because of this we can give appropriate visual cues to a user who may need to manipulate an object with care and sensitivity to its fragility, or may wish to monitor the motion or kinematics of an object. The user can see an image of a tags object as well as its data overlaid onto the tag. They can also see the objects tangent vector and trajectory

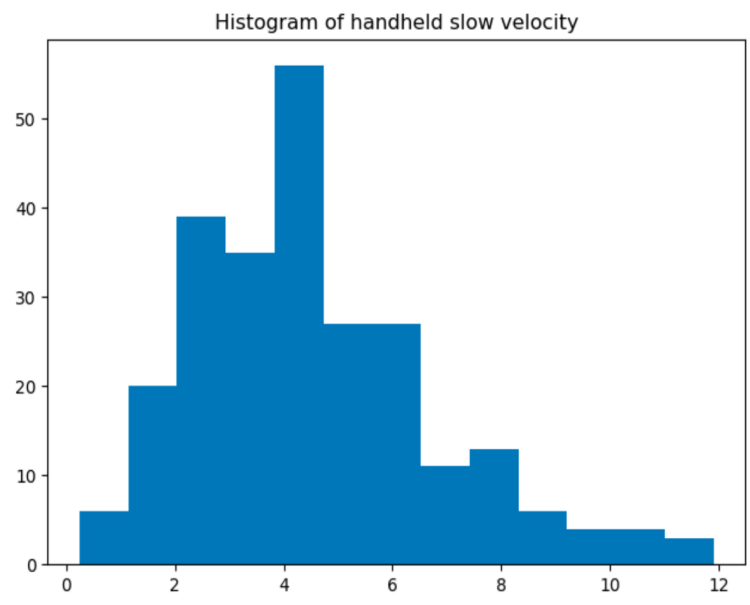


Figure 12: Histogram of handheld low velocity measurement

mapped to a color indicative of the safety level as the object moves with varying speed. Lastly the will see the image replaced with a warning when the object speed or acceleration reaches an intolerable threshold. Developers and engineers can thus use this system in their design to adhere to object transportation and care guidelines in their particular applications.

Limitations

Pose Estimation Noise

In Kam et al. (2018) [7], they also presented the comparison between the pose estimation with and without a Kalman filter. They pointed out that the subtle vibration of markers could affect the pose computed from the image. In order to stabilize the pose estimation, a Kalman filter should be used. But in our case, a Kalman filter was not implemented. Although the noise cannot be removed, we can still observe the expected velocity measurement from the plotting of real-time instantaneous velocity. We have bench-marked the computation by putting a package on a treadmill or moving an handhold tag with a constant speed.

High Velocity

When we are running our program on 1.5 and 2 mph treadmill, the aruco code cannot be captured. This issue might be due to the resolution of the camera. But one of our purpose of this project would be tracking fast moving object, and this limitation needs further improvement, either on recording devices or implementation. The frame image we have obtained at 2 mph has been shown in Figure 13. The aruco marker has been blurred due to the high velocity and camera resolution.

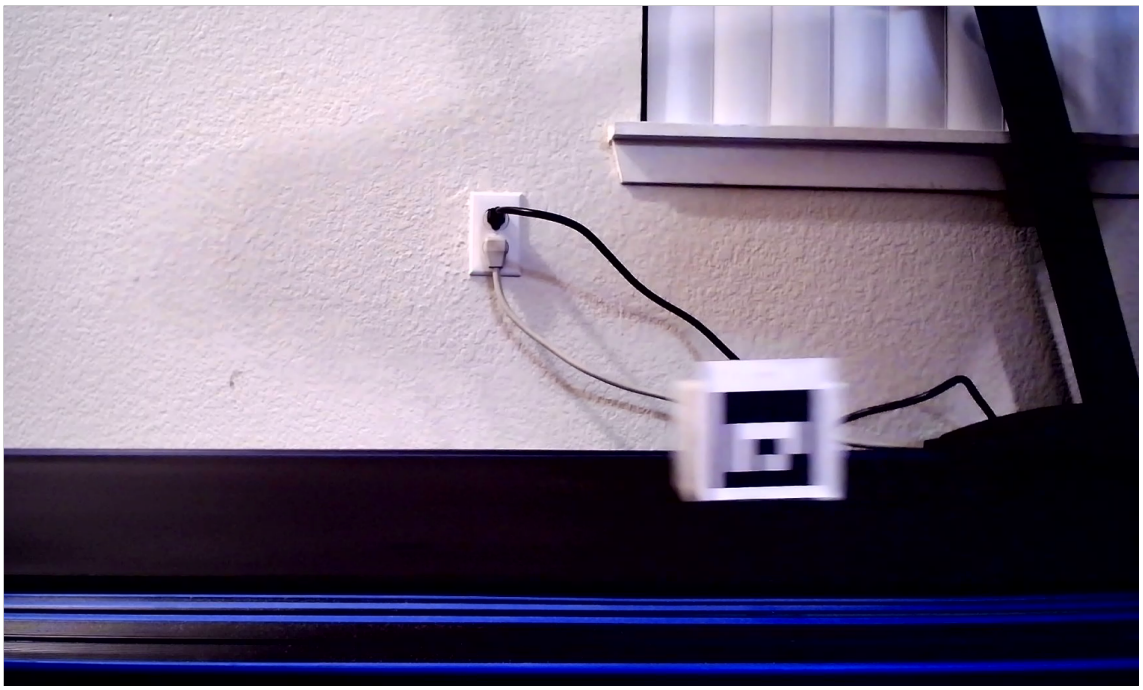


Figure 13: Fast moving object: cannot detect aruco

Aruco Code Occlusion

When aruco tag is partially blocked, our program might not detect the tag, or if the object which a tag is attached rotates to an angle such that aruco tag is not visible any more, our approach can no longer work. In these cases, we might need some estimation method introduced in the references we listed in the previous work section, interpolation or approximation is necessary to give a reasonable pose result.

Bibliography

[1] Mark Asselin, Andras Lasso, Tamas Ungi, and Gabor Fichtinger. Towards webcam-based tracking for interventional navigation. In *Medical Imaging 2018: Image-Guided Procedures, Robotic Interventions, and Modeling*, volume 10576, page 1057627. International Society for Optics and Photonics, 2018.

[2] Danilo Avola, Luigi Cinque, Gian Luca Foresti, Cristina Mercuri, and Daniele Pannone. A practical framework for the development of augmented reality applications by using aruco markers. In *International Conference on Pattern Recognition Applications and Methods*, volume 2, pages 645–654. SCITEPRESS, 2016.

[3] Jan Bacik, Frantisek Durovsky, Pavol Fedor, and Daniela Perdukova. Autonomous flying with quadrocopter using fuzzy control and aruco markers. *Intelligent Service Robotics*, 10(3):185–194, 2017.

[4] Ting Kwok Chan, Ying Kin Yu, Ho Chuen Kam, and Kin Hong Wong. Robust hand gesture input using computer vision, inertial measurement unit (imu) and flex sensors. In *2018 IEEE International Conference on Mechatronics, Robotics and Automation (ICMRA)*, pages 95–99. IEEE, 2018.

[5] Alberto Crivellaro, Mahdi Rad, Yannick Verdie, Kwang Moo Yi, Pascal Fua, and Vincent Lepetit. A novel representation of parts for accurate 3d object detection and tracking

in monocular images. In *Proceedings of the IEEE international conference on computer vision*, pages 4391–4399, 2015.

[6] Nathan Elangovan, Anany Dwivedi, Lucas Gerez, Che-Ming Chang, and Minas Liarokapis. Employing imu and aruco marker based tracking to decode the contact forces exerted by adaptive hands. In *2019 IEEE-RAS 19th International Conference on Humanoid Robots (Humanoids)*, pages 525–530. IEEE, 2019.

[7] Ho Chuen Kam, Ying Kin Yu, and Kin Hong 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)*, pages 65–69. IEEE, 2018.

[8] Torstein A Myhre and Olav Egeland. Collision detection for visual tracking of crane loads using a particle filter. In *IECON 2016-42nd Annual Conference of the IEEE Industrial Electronics Society*, pages 865–870. IEEE, 2016.

[9] Mohammad Fattahi Sani and Ghader Karimian. Automatic navigation and landing of an indoor ar. drone quadrotor using aruco marker and inertial sensors. In *2017 International Conference on Computer and Drone Applications (IConDA)*, pages 102–107. IEEE, 2017.

[10] Sondre Sanden Tørdal and Geir Hovland. Relative vessel motion tracking using sensor fusion, aruco markers, and mru sensors. 2017.

[11] Rodrigo S Xavier, Bruno MF da Silva, Luiz MG Gon, et al. Accuracy analysis of augmented reality markers for visual mapping and localization. In *2017 Workshop of Computer Vision (WVC)*, pages 73–77. IEEE, 2017.

[12] Guoxing Yu, Yongtao Hu, and Jingwen Dai. Topotag: A robust and scalable topological fiducial marker system. *arXiv preprint arXiv:1908.01450*, 2019.