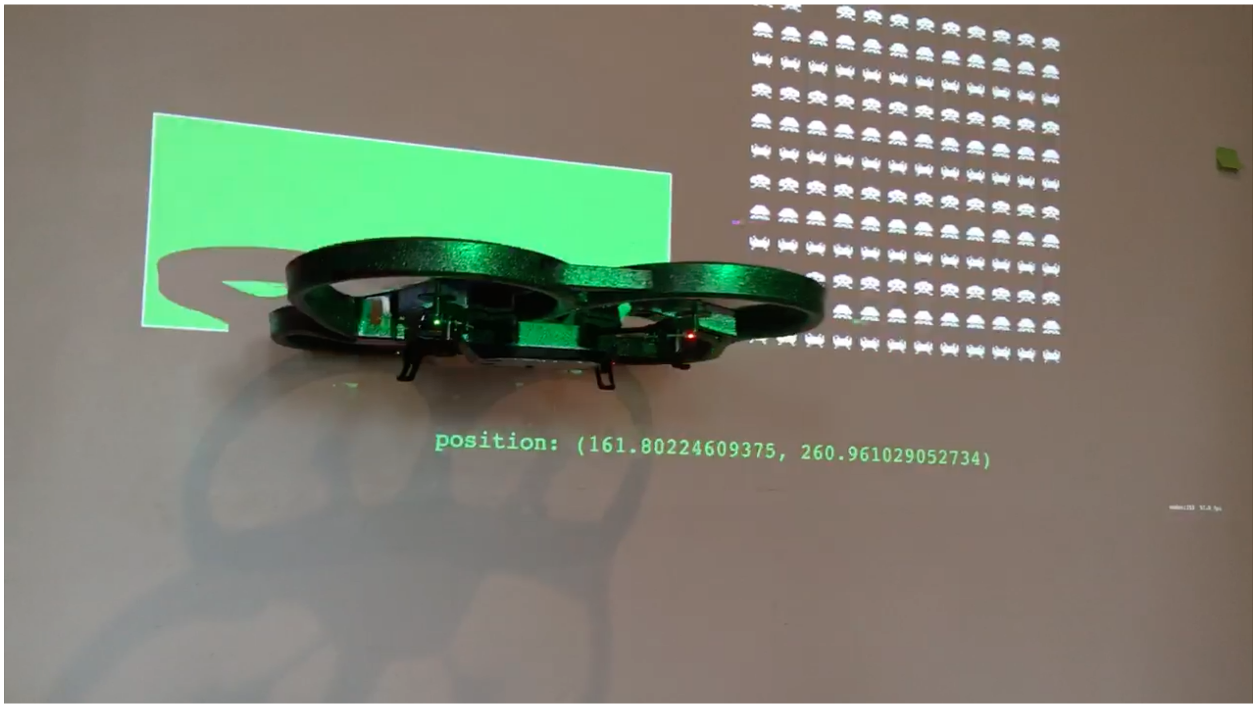
Playing Space Invaders with a Quadcopter in a Mixed Media Interface

Kayhan F. Qaiser



```
position: (161.80224609375, 260.961029052734)
```

Figure 1. Space Invaders played with a quadcopter. Localization shown in green.

1. Introduction

Augmented reality is a form of mixed media where virtual objects are projected into the physical world. This paper focuses on potential roles robots could play in creating novel mixed media interfaces. Our main reference is Mechanical constraints as computational constraints in tabletop tangible interfaces [**PI07**]. We use this as a model for how with tangible mixed media interfaces can create novel experiences. We believe this field of augmented reality has been underdeveloped and that the technologies exist to create simple, useful and fun interfaces.

Specifically we will discuss the implementation and challenges of our variation on the classic arcade game, Space Invaders. In our variation the game is projected onto a wall, but instead of having a projected spaceship the player pilots a physical quadcopter. The quadcopter is tracked using a camera so that its position can be accurately represented in the game. The quadcopter is piloted up and down in order to dodge lasers from the invades and to return fire.

This paper will broadly cover two sections. Firstly we will discuss various methods of implementation of the game that were developed, including vision and tracking. This section will cover challenges in implementing these methods and the effectiveness of each. After this we will provide an analysis of the success of the game and mixed media interface using subjective data gathered from players. We decided to analyze subjective data because of the small sample size of players. We simply do not have a significant sample size, nor were players playing long enough to do an analysis of objective data such as player scores.

We used two main tools to create this project. The first was Apple's Sprite Kit framework for creating the Space Invaders game. We modded an open-source version of the game to suit our needs. The second tool was Opencv which we used for our vision and tracking.

2. Vision and Tracking

2.1. Overview. In order to create an effective mixed media interface having a vision and tracking system is usually a must. Today there are two very popular tools that are used to implement vision and tracking. The first is Microsoft's Kinect and the second is the OpenCV library. Both are outstanding tools but for our purposes we decided that OpenCV was the best choice. The Kinect is a very powerful piece of hardware and software and excels in situations where depth perception or 3D vision is required. For our game we project onto a flat wall and fly a quadcopter just in in front of it. Although depth perception could be used here we decided that it was absolutely necessary because of the relatively close proximity of the quadcopter and the wall.

After researching both technologies we decided to utilise OpenCV because it is lightweight, cross-platform and sufficiently powerful for our needs. OpenCV also gives us the ability to run our tracking algorithms on mobile devices without changing our code-base. Another major concern was ease of integration with other technologies, which OpenCV scores quite high on.

There were two main approaches that were considered for our vision and tracking system. The first was a purely vision approach and the second used a vision simulation hybrid.

Initially we had opted for a pure vision approach for our game. A camera would observe the an image of the game state and try to report any important events. For example, the vision system would be able to track a laser beam across the screen and report if it collided with a craft. A purely vision based approach would have been very powerful and could have been applied to more complicated games. The problems we ran into were essentially trying to capture which entities we actually wanted to track and trying to figure out when they collided with each other. This was initially done by making different entities different colours and filtering those. However that method did not scale well. Implementing a collision tracker without colour information proved to be very hard. Eventually we concluded that a pure vision system was overkill for what we needed to achieve so we went with the simpler alternative.

The second approach was where only the state of our quadcopter was tracked and then modeled inside our game. This method was substantially easier to implement as only one object had to be tracked and any collisions or interesting events were simulated in the game itself. This approach was divided into three sections: calibration, tracking and simulation. The high level steps of our method are as below:

(1) Project fiducial markers and calibrate screen.
(2) Detect quadcopter object and save position, with and height.
(3) Map quadcopter coordinate from the camera space to the game space.

(4) Simulate the quadcopter with a bounding box in the game.
(5) Compute game logic and repeat from 2.

2.2. Calibration. The first step to localizing our quadcopter was to calibrate our coordinate system in real life to our coordinate system in the game. In our game our space ship has only two degrees of freedom however in the world our quadcopter has 6 degrees of freedom. Since our game is 2D we must create a function that maps the x and z coordinates of the quadcopter to that in our game.

Our fist step was to accurately measure the x and z coordinates of our quadcopter. To do that we required a relative position from which to measure from. We decided to use fiducial markers to accomplish this. Fiducial markers are simply easily identifiable markers that can be used to create a relative coordinate system and find objects. Fiducial markers are very popular in computer vision and robotics and have been proposed for and used in applications such as space craft docking[**HM92**] and localization of robots in camera networks[**RMD06**]. We decide to utilize a very simple fiducial marker, a simple red circle. We projected 4 circles on each corner of our screen. The red circle outlined in green shows a successful categorization by our calibration algorithm.

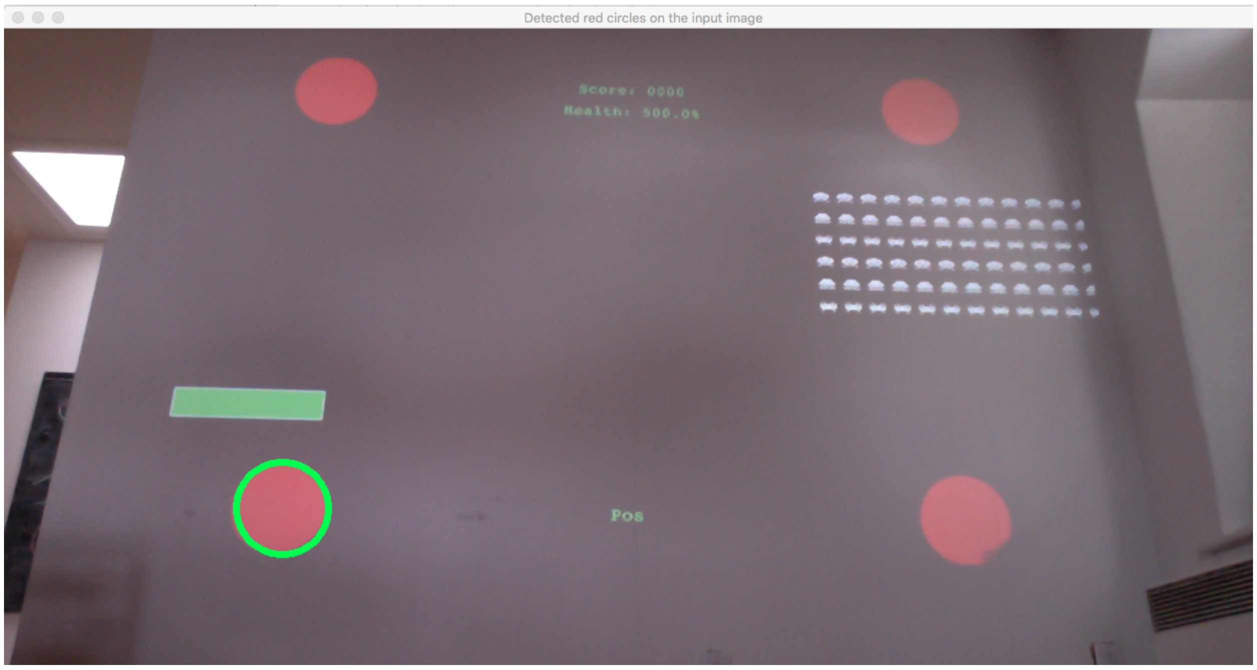


FIGURE 2. Red circles as fiducial markers on corners of projected screen.

Finding such a red circle in a image is a fairly simple vision task, that is one of the reasons why we chose these as fiducial markers instead of more information heavy markers. First we convert our image to the HSV (hue, saturation and value) space. This has the advantage of allowing us to find a colour only using one value, the hue. We then remove all pixels that are not in our hue. After this we blur the image using a simple Gaussian blur to help us avoid false positives. Finally we have to detect which of our red objects are circles. We use the OpenCv function HoughCircles to achieve this. This function has to be tuned depending on your tolerance for what is a circle. For our application we required a medium tolerance as the shape of our projected circles changed significantly depending on the projector and camera angle.

Once all our markers were detected we applied a sorting algorithm to determine which ones were at which coordinates. Our first implementation of a tracking system utilized the naive approach and made the assumption that in the captured image the circles formed a rectangle. This however is rarely the case because our projector projects the image onto the wall at an angle and our camera records this projected image at an angle. The result is a camera that seeing a keystoned image. Keystoning refers to the effect of a square or rectangular image being observed as a trapezoid. Our initial naive approach calculated the relative coordinate of the quadcopter did not solve this keystoning effect but provided a decent result using the slope of the keystone and the screen width computed with the top two markers.

$$Point2f relPoint = Point2f((quadPoint.x - topLeft.x)/screenWidth, \\ 1.0f - ((quadPoint.y - topLeft.y)/screenHeight)); \tag{1}$$

We would like our system to be robust enough so that any projector angle α and camera angle β do not create inaccurate tracking. It is known that any two images of the same planar surface in space are related by a homography, an isomorphism of projective spaces. This can also be thought of as a bijection mapping lines to lines. We must assume a pin hole camera model for this to hold, which we do. Homographies are used to project images at the correct perspective for applications in augmented reality. A simpler example is using a camera to correct for a keystoned projector image [**SSM01**].

In OpenCv we use a perspective transform to map our observed coordinates into our known game coordinates. Using this transformation matrix we are able to map a coordinate in our vision space to that of our game space. We will discuss the experimental results of our calibration and tracking methods in the coming section.

2.3. Tracking. We decided to project the game on a plain white wall so that the quadcopter would easily contrast. We first converted our image to greyscale and then thresholded to only detect darker objects that greats contrast after which we slightly blur our image. After this we used the Canny algorithm[**Can86**] to detect edges. We then find the bounding boxes of all closed loops and computed which one was most likely to be our quadcopter. Any loops or edges which are too small to be our object are removed. Below the tracking algorithm is detecting a piece of paper taped to the wall and objects on the floor and sides of the wall.

2.4. Simulation. The tracking algorithm returns us the most likely object to be our quadcopter. We use the coordinates and size of this object with the calibration steps above to send our game a transformed coordinate and size of the quadcopter. Once this is received by the game it can create a bounding box and keep updating its position. It projects this bounding box in green over the actual quad. When the player moves the bounding box moves and when he fires a laser is shot from the center of the bounding box.

Any collisions between lasers are handled internally with the game logic. If the quadcopter is hit then its health decreases. A large advantage of our decoupling of the game logic and tracking algorithms is that they can be applied to any game. In short any simple game could easily be modified to allow it to be played in a mixed media interface.

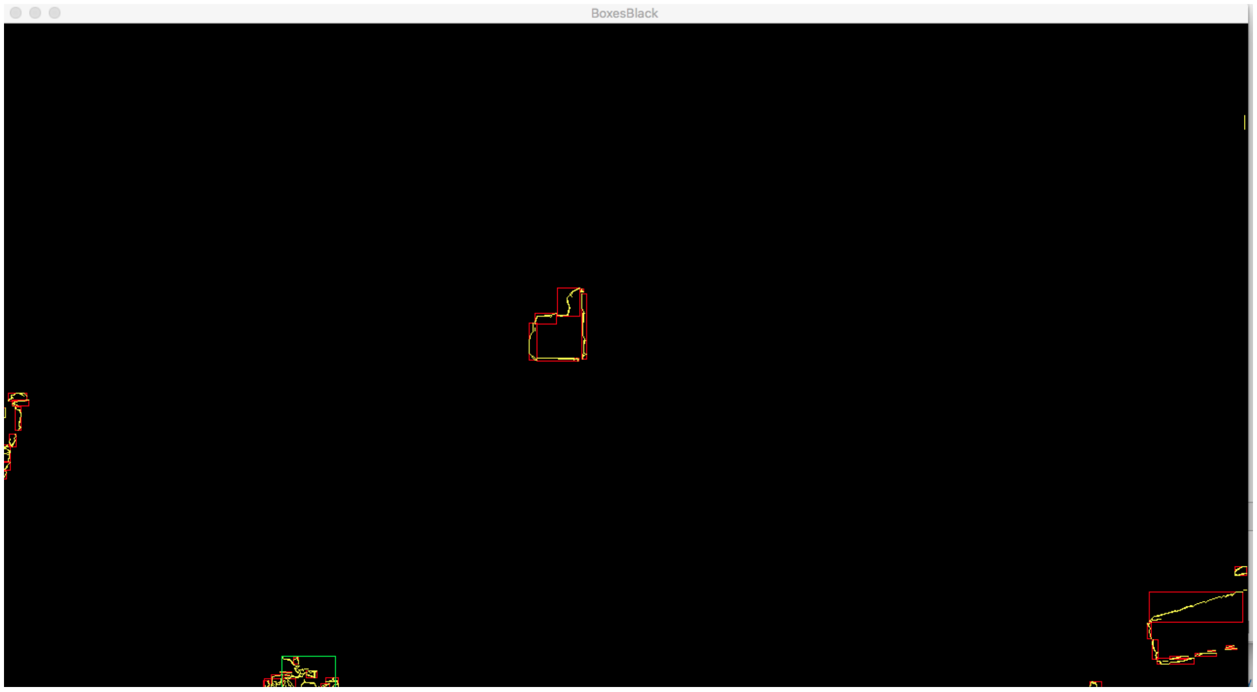


FIGURE 3. Canny algorithm detecting edges overlayed with bounding boxes.

3. Results

3.1. Calibration and Tracking. To judge how accurately each of calibration and tracking methods worked we gathered data related to the position of a rectangular marker at 9 points on the game-play area. For all these approaches the projector angle α was 10° and the camera was placed on the floor looking up at an angle of 20° .

The first naive approach we used was to utilize only two fiducial markers at the top left and bottom right positions of the screen. The width and height of the screen were calculated using these markers so that we could calculate the relative coordinate of the quadcopter between 0 and 1. Figure 5 shows how the position of the detected rectangle differed from the actual rectangle, with the detected rectangle shown in green. We will call this the tracking figure of the calibration method.

We calculated a metric for how accurate the tracking was using the tracking figure. More specifically we compute this metric using the differences between the measured and actual coordinates, where R represents the actual rectangle and M represents the measured rectangle. W is the width of the rectangles and H is the height. We compute the tracking metric both for the x and y coordinates of each method. The closer these metrics are to zero the better our tracking method is. For our first method we had $\eta_x = 0.51$ and $\eta_y = 0.32$.

$$\eta_x = \sum_{n=1}^{9} \frac{|Rx_n - Mx_n|}{9W} \tag{2}$$

$$\eta_y = \sum_{n=1}^{9} \frac{|Ry_n - My_n|}{9H} \tag{3}$$

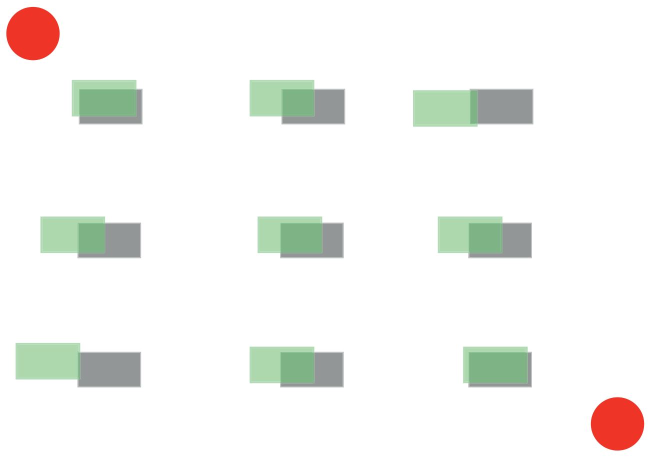


FIGURE 4. Tracking diagram for diagonal markers.

The second method was to use 4 fiducial markers and to calculate the width by subtracting the top two markers. This method was discussed in the Calibration section. Here our accuracy was better with $\eta_x = 0.28$ and $\eta_y = 0.13$.

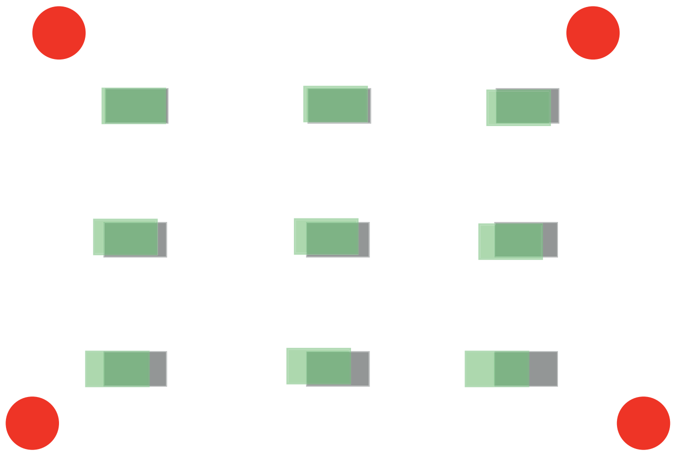


FIGURE 5. Tracking diagram for relative coordinates approach.

Our last method was to use a perspective transform as discussed above to translate the observed point into the game coordinate space. For this method we were able to directly transform the points of the quadcopter so there was no need to calculate a width and height and compute a relative coordinate. This method provided the best results with $\eta_x = 0.09$ and $\eta_y = 0.08$.

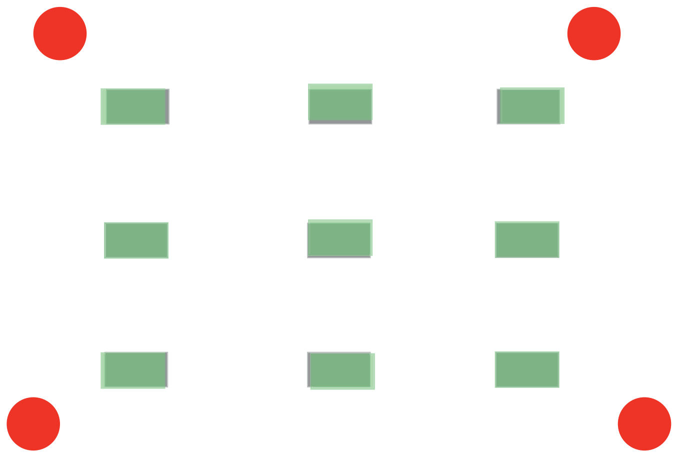


FIGURE 6. Tracking diagram for perspective transform approach.

4. Conclusion

Overall we were happy with the results from our chosen methods. in particular we believe that using a simpler tracking and simulation method instead of a purely vision approach is better for a simple game like Space Invaders.

A major problem of our current system is the fact that if after calibration if the camera is at all moved the tracking algorithm is incorrect. Recalibration is now necessary. A viable solution is to recalibrate at a set timestep, for example at every 10 seconds. However, this is likely to distract from the game play as fiducial markers would have to be redisplayed on the game screen. The current method is not robust enough because of this. We would suggest less invasive fiducial markers, such as projecting a distinct boundary around the screen on the wall and tracking this boundary. Often in mixed media interfaces physical objects are used as fiducial markers[**KB07**]. This approach was considered, however, we concluded that projecting our fiducial markers at known positions in our game space simplified the calibration process.

Another limitation of our current system relates to inaccuracies in tracking at certain positions. Because we make the assumption that our quadcopter is a flat object on the plane of our wall when our camera is placed either very low, looking up, or very high, looking down we receive an inaccurate localization. For example as shown in the figure below when we detect our quadcopter from below we imagine it being higher than it actually is as the line of sight is projected up to the wall. This inaccuracy can be observed in the demo video on the project web page.

Again, we do not want the z coordinate of the projection onto the wall, but we want to true z coordinate of the quadcopter from the point of view of a player looking straight onto the wall. We could attain this value by taking the y coordinates into account and using the projection model. The distance from the wall to the projector could be measured as well as the expected distance from the quadcopter to the projector (this could be also computer on the fly by using the quadcopter's measured width). These coordinates would be used along with the perspective transform to adjust the tracking coordinates for a view straight onto the wall which is what we require for our pseudo 2D game.

Our tracking results were in general satisfactory. We would like to implement a more sophisticated rating systems for identified objects. An easy improvement would be to give object in the main game

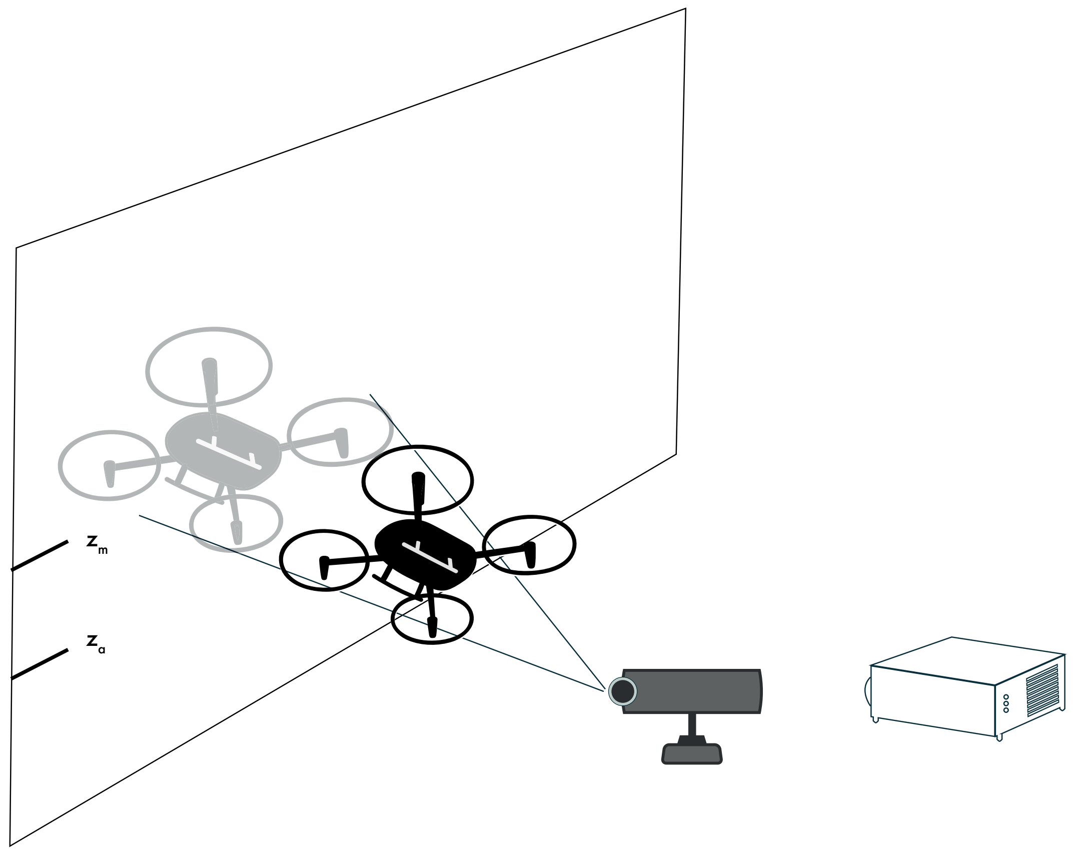


FIGURE 7. Inaccuracy in model of measured z coordinate when camera is too low.

play area a higher weight. This would remove false positives like wall edges. In addition we would like to implement a smoothing so that sudden jumps in the tracked object do not occur. The tracking system also requires additional tuning so that the single quadcopter object is not sometimes perceived as multiple separate objects.

The tracking system we have developed could easily be applied to any other simple game. An example of a possible follow up to this project would be implementing Breakout using a ground robot as the paddle. This system could also be extended for other mixed media interfaces, for example, to display origami instructions on a piece of paper as it is being folded on a flat surface. An algorithmic folding sequence such as that in Scaling Any Surface Down to Any Fraction by Demaine, Demaine and Qaiser[**DDK15**].

A webpage of the project can be found here as well as a video of a demo of the game being played. All the programs that were written for this project can be found on Github here.

5. Acknowledgements

I would like to thank Gareth Dicker for his help sourcing and piloting a drone during testing for this project. I would also like to thank Marc Burns for his advice on perspective transforms. I also would like to extend a huge thanks to Prof. Dudek and Prof. Merger for allowing me to pursue quite an adventurous and open ended project and for letting me get my hands dirty with OpenCv and robots in the real world.

References

[Can86] John Canny, *A computational approach to edge detection*, Pattern Analysis and Machine Intelligence, IEEE Transactions on (1986), no. 6, 679–698.

[DDK15] Erik D Demaine, Martin L Demaine, and F Qaiser Kayhan, *Scaling any surface down to any fraction*, Origami6: Sixth International Meeting of Origami Science, Mathematics, and Education, 2015, pp. 201–208.

[HM92] C. C. J. Ho and N. H. McClamroch, *Autonomous spacecraft docking using a computer vision system*, Decision and Control, 1992., Proceedings of the 31st IEEE Conference on, 1992, pp. 645–650 vol.1.

[KB07] Martin Kaltenbrunner and Ross Bencina, *reactivision: A computer-vision framework for table-based tangible interaction*, Proceedings of the 1st International Conference on Tangible and Embedded Interaction (New York, NY, USA), TEI '07, ACM, 2007, pp. 69–74.

[PI07] James Patten and Hiroshi Ishii, *Mechanical constraints as computational constraints in tabletop tangible interfaces*, Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (New York, NY, USA), CHI '07, ACM, 2007, pp. 809–818.

[RMD06] Ioannis Rekleitis, David Meger, and Gregory Dudek, *Simultaneous planning, localization, and mapping in a camera sensor network*, Robotics and Autonomous Systems **54** (2006), no. 11, 921 – 932, Planning Under Uncertainty in Robotics.

[SSM01] Rahul Sukthankar, Robert G Stockton, and Matthew D Mullin, *Smarter presentations: Exploiting homography in camera-projector systems*, Computer Vision, 2001. ICCV 2001. Proceedings. Eighth IEEE International Conference on, vol. 1, IEEE, 2001, pp. 247–253.

McGill University, Montréal, Québec, Canada
E-mail address: `kayhan.qaiser@mail.mcgill.ca`

Student Number: 260 507 016