ASE 479W: Find the Balloon

Victor Branea

April 10th, 2023

1. Introduction

This lab deals with computer vision. Specifically, completing a number of functions in C++ that recognize and locate the center of a blue or red balloon, and then estimate the balloons' position.

2. Results

3D Location estimation: MATLAB implementation

In order to test the accuracy of the estimate3dFeatureLocation, the testing protocol in camTest.m was iterated 10000 times and the resulting errors between rXI and rXIHat were averaged.

Vector Component	Absolute Error (m)
X direction	0.016616
Y direction	0.009508
Z direction	0.011290

Figure 1 Averaged absolute errors between estimated and true feature positions

As can be seen in the table in figure 1, the averaged absolute error over 10000 iterations is less than 2 cm for each component of the vector.

A useful metric to test the estimate3dFeatureLocation function is how the accuracy changes as the spacing between images increases. By generating a structure M, which is loaded with the location on the camera plane, attitude of the camera, and position of the camera for every instance that the hdCameraSimulator function sees a desired point (in our case [0;0;0.7]), we can generate estimations of the feature's position and find their accuracy. This is done by taking 10 evenly spaced instances, using estimate3dFeatureLocation to estimate the position of the feature, and subtracting the actual position of the feature to get the error.

A suitable range of spacing between images is 1 to 50. The total sample of images (> ~1700 times that the point [0;0;0.7] was seen by hdCameraSimulator) was parsed by this spacing range starting at the 1st, 400th, 800th, and 1200th sample in order to check if the "stage" of the circular path affects the errors.

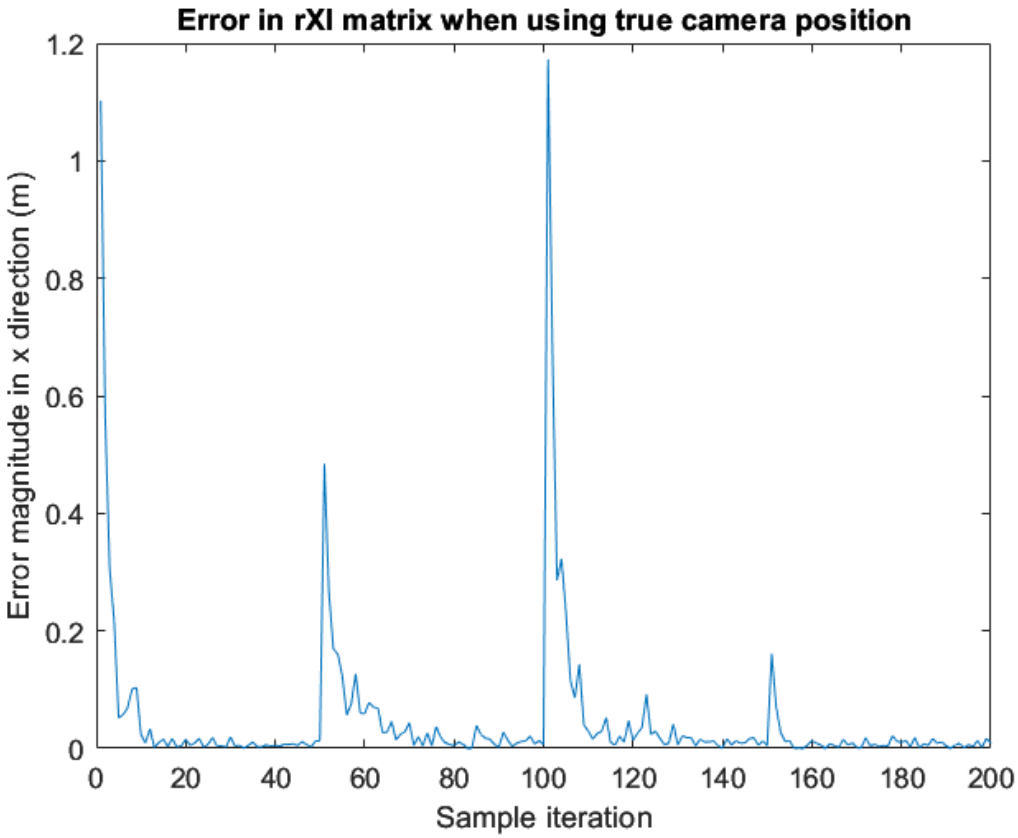

Figure 2 Error in x direction when using true camera position

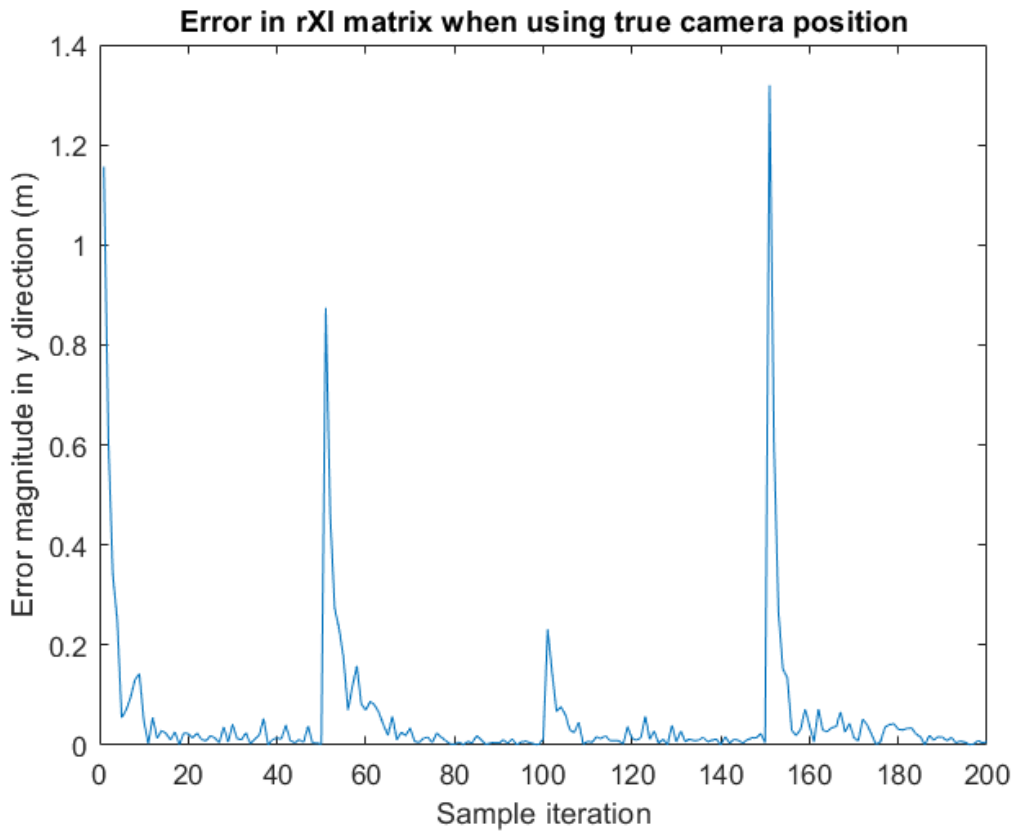

Figure 3 Error in y direction when using true camera position

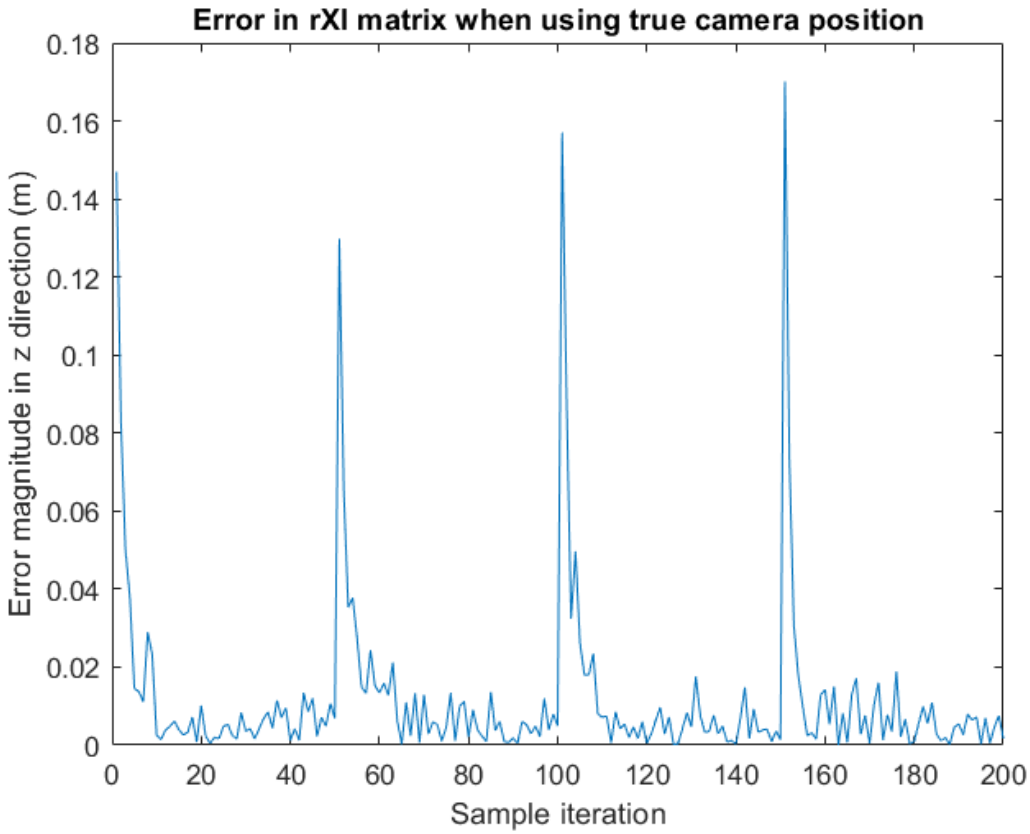

Figure 4 Error in z direction when using true camera position

Figures 2, 3, and 4 all show 4 distinct spikes in the error, which all occur when the spacing between images is 1. The error then greatly decreases as the spacing becomes larger than 10. The "stage" of the circular path does not seem to greatly influence this pattern. This leads to the conclusion that it is more accurate to use images taken further apart from one another to estimate a feature's location, in order to minimise the potential error.

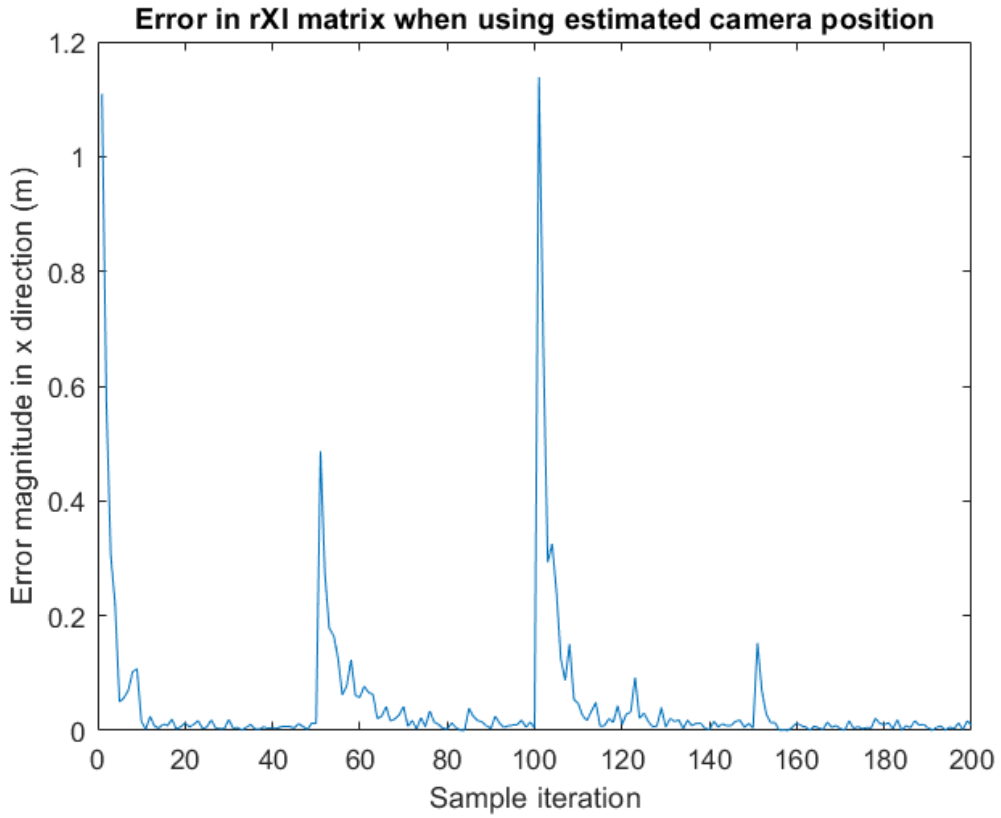

Figure 5 Error in x direction when using estimated camera position

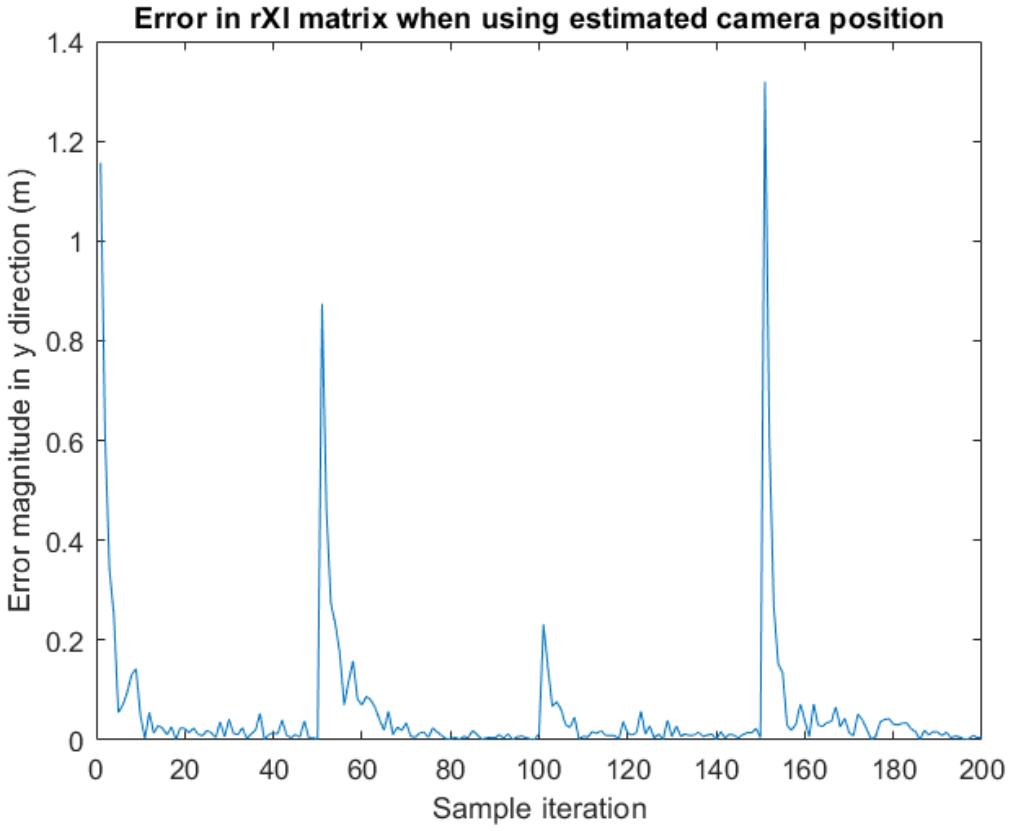

Figure 6 Error in y direction when using estimated camera position

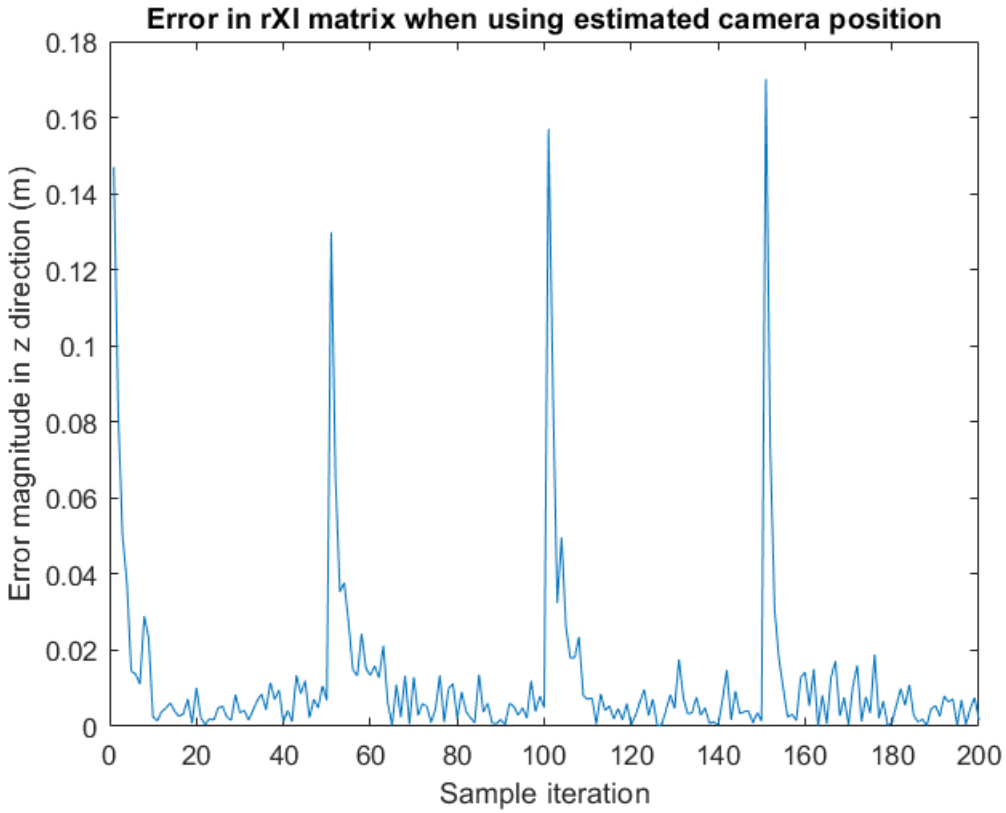

Figure 7 Error in z direction when using estimated camera position

The plots in figures 5, 6, and 7 are generated in a similar fashion to the previous three plots, however, instead of using the true camera position to estimate the location of the feature, the estimated location of the camera from the "noisy" measurements (the code for which was developed during Lab 3) was used to do so. Interestingly, the change from true to estimated position of the camera did not change the overall conclusions that can be drawn from this error analysis.

3D Location estimation: C++ implementation

The same code that was implemented in MATLAB was re-written in C++ as part of a bigger goal of recognizing and finding red and blue balloons in a set of given images.

Finding balloons within image plane

A function to locate red and blue balloons in a given image was developed such that by taking an image as an input, it would check the image for any areas that are within some range of red or blue. The image would then be transformed into a black and white image: the white zone would be the area that is either red or blue, depending on the function input, and the rest of the image would be turned black. In order to remove any noise in the image, it would first be eroded, following by it being dilated by the same amount. This would ensure that any small inconsistencies would be removed.

Each white region in the image would then be surrounded by a contour. Following this, a rectangle would be constructed to enclose the contour, in order to calculate the region's aspect

ratio. The contour would also be encircled by the smallest possible circle, to find an approximate center and radius of the region.

By checking every contour for a minimum and maximum aspect ratio and for a minimum radius, it could be determined what is likely to be a balloon. This was then tuned in two ways to make the selection of objects considered to be balloons better and to find the center of the balloons more accurately.

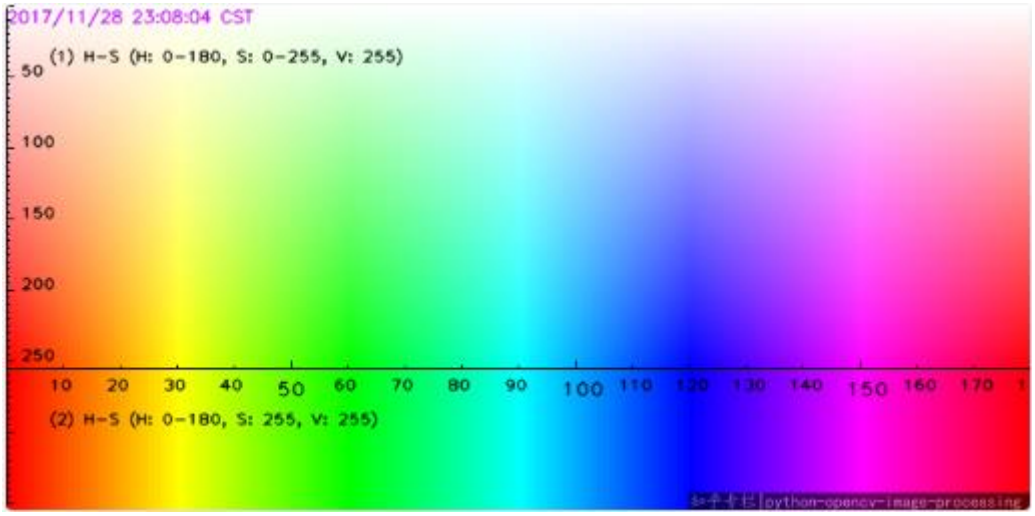

Figure 8 HSV color map used as reference [1]

To better find the balloon's center, the acceptable red and blue color range was tuned, after being initialized by approximating values from the color map in figure 8. This was done by visually checking a number of the images, and making note of what regions of a balloon the function didn't recognize. Then, the RGB color code of that region was extracted by using Microsoft Paint's color picker tool on a screenshot of the unrecognised region. The RGB color code would then be converted to HSV using [2], as this is the color space that the function works in. Finally, the color range would be extended to include this new color. The final ranges for the colors are: blue = [(90,70,100) (110,255,255)], red = [[(0,70,100) (10,255,255)] U [(170,70,100) (180,255,255)]].

To make a better selection of what objects are recognised as balloons, the minimum and maximum aspect ratio and minimum radius were tuned through trail and error. The final values used are shown in figure 9.

Min Aspect Ratio	1.0
Max Aspect Ratio	1.35
Min Radius	100

Figure 9 Tuned values for balloon recognition

Finally, the errors in the estimates for the blue and red balloons are smaller than 5cm, as can be seen in the screenshot in figure 10, alongside the error covariance matrix square root diagonal for each color.

```
Blue balloon estimated 3D location error:
 -0.0118644
 -0.0287593
-0.00772522
Red balloon estimated 3D location error:
-0.0469564
 0.0376353
-0.0031116
Blue error covariance matrix sqrt diagonal:
0.00547868
0.00350961
0.00243408
Red error covariance matrix sqrt diagonal:
0.00792216
0.00856952
0.00577688
```

3. Conclusion

In conclusion, a MATLAB implementation of a linear least squares problem was set up and solved in order to estimate the position of a feature in 3D, knowing its position on the camera plane. This implementation was then coded up in C++, alongside a function that recognises and finds the center of blue and red balloons in a given image set. The final implementation was then shown to have an error of less than 5 cm in any given direction with respect to the known real location of the balloons.

4. References

[1] "Color Filtering/Segmentation/Detection – HSV." *Computer Vision*, 10/04/2023, https://cvexplained.wordpress.com/2020/04/28/color-detection-hsv/

[2] "RGB to HSV color conversion." *RGB to HSV conversion | color conversion*, 10/04/2023,

https://www.rapidtables.com/convert/color/rgb-to-hsv.html