MAE 6760 Final Project:

End Effector Localization Using an Onboard Camera and Kalman Filtering

18 December 2020

Prepared by: Daniel DiAngelis

Professor: Dr. Mark Campbell

Additional technical assistance provided by: Matt Law

1 | Abstract

This project develops a method for the workspace-frame localization of a mini-palletizer robotic arm using an onboard camera. A simplified model was defined to describe the robot's wrist position after each control command is executed, a small set of data was collected to characterize the arm's process noise, and a simple simulation of the arm moving was run. Data was collected of the arm moving randomly with an onboard camera fastened to the robot's wrist as well as an overhead camera to determine the true position. This data was used to train a convolutional neural network (CNN) to make inferences of the position based on the onboard camera image. A Kalman filter was developed to combine the CNN-based sensor model with the platform model. This filter was then tuned and was found to provide more accurate estimates than the CNN alone, but still included significant error most likely resulting from an insufficient amount of training data for the CNN, and inaccuracies in the models defined. While the filter did not perform as well as would be required for this system, it offered promising evidence that the methods presented here could be adjusted and further tuned to produce more accurate estimates.

2 | Project Goals

The goal of this project is to develop a method for the end effector localization of an inexpensive, miniature palletizer robotic arm. The arm used is a "uArm Metal" from uFactory Technology Co. The arm can be seen in *Figure 1*.

Figure 1: uArm Metal [1]

This arm is to be used as the base for a human-robot collaborative design project. The arm system used PWM signals sent to 3 servo motors to set position. This positioning method only accounts for the arm's position relative to its own base, and not the workspace. Therefore, a separate system is required to close the loop between the arm and its workspace. The entire control system that is to be implemented on the platform is shown in *Figure 2* which highlights the area that this project will be focused on.

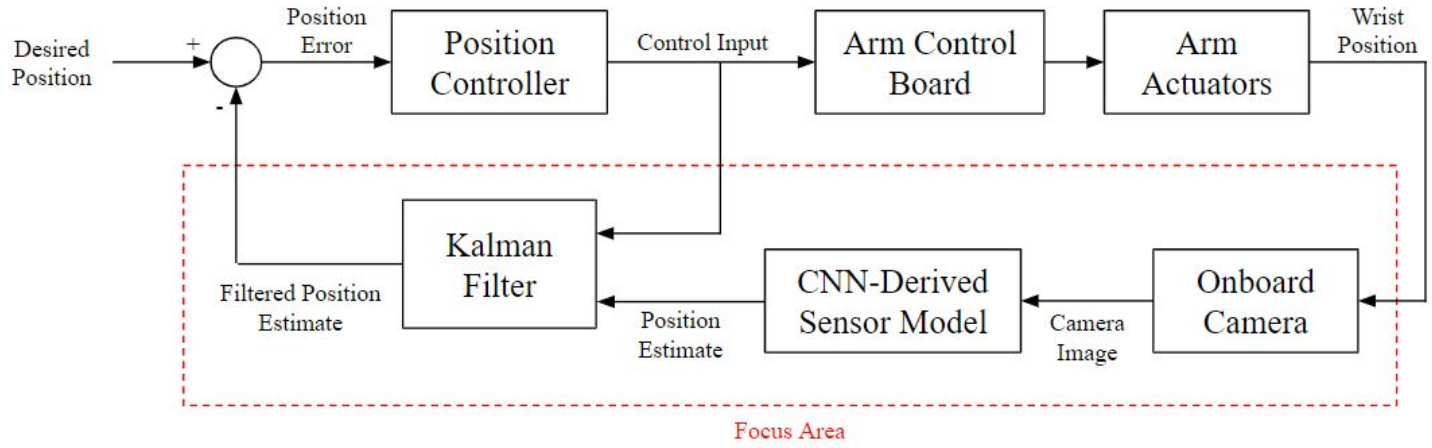

Figure 2: Control system architecture with focus area boxed in red

This project focuses on state estimation done through the use of a Kalman filter on camera readings fed through a convolutional neural network (CNN) trained to make inferences about the position. Workspace-frame localization of the end effector will allow for greater accuracy in the pick and place and projection tasks that will be required of the robot.

3 | Technical Approach

3.1 | Platform Model Development

Due to the level of abstraction at which commands are sent to the robot, the defined model is very simple. The robot state is defined in *Equation 1*.

$$\underline{x} = \begin{bmatrix} x \\ y \\ z \end{bmatrix}$$

$$x, y, z = wrist\ position\ (mm)$$

Equation 1: State Definition

The state is measured from an origin on the workspace floor below the center of the base of the robot to the wrist as shown in *Figure 3*.

Figure 3: Origin and wrist location definitions

This project will focus primarily on estimating the x and y wrist location. The discrete-time dynamic system is defined in *Equation 2* where each timestep is defined after each positional control input is executed.

$$\underline{x}_{k+1} = \begin{bmatrix} x_k + u_{x,k} + w_{x,k} \\ y_k + u_{y,k} + w_{y,k} \\ z_k + u_{z,k} + w_{z,k} \end{bmatrix} = I\underline{x}_k + I\underline{u}_k + I\underline{w}_k$$

$$u = control\ input \quad w = process\ noise$$

Equation 2: Platform model

To get an idea of the accuracy of the robot's servo-based position controls, the arm was started at different points and commanded to move a specified distance. The distance from the start point to the actual end point was then measured. The errors associated with these commands are shown in the histogram in *Figure 4*.

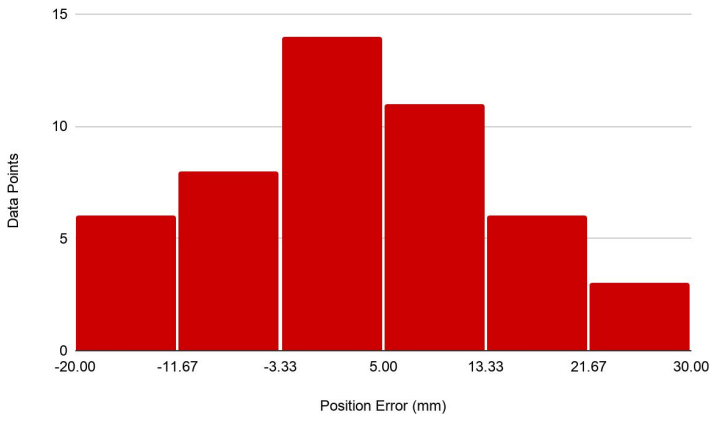

Figure 4: Process noise histogram

The data collected was approximately zero-mean, and had a standard deviation of 11.5. This information along with *Equation 2* was used to simulate the system's response to a set of random control inputs with a realistic amount of process noise. A plot obtained from this simulation can be seen in *Figure 5* which shows a comparison of the wrist position with and without process noise.

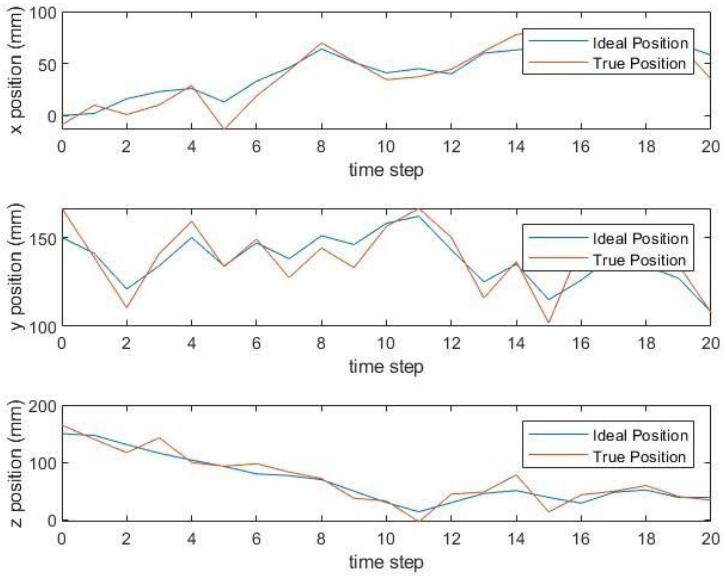

Figure 5: Simulation results of platform

3.2 | Sensor Model Development

A CNN was trained on a set of data containing images taken by a camera onboard the arm and the true position of the end effector found by an overhead camera that would not be a part of the final design. To collect this data, the arm was augmented to hold a Raspberry Pi with an attached camera, as well as an AprilTag [2] fiducial. Images of these augmentations can be seen in *Figure 6*.

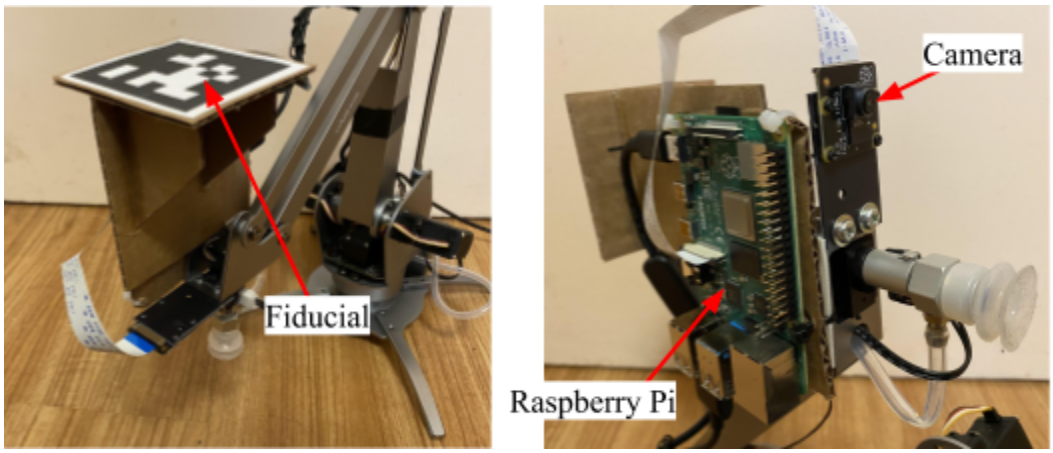

Figure 6: Robot augmentations for sensor model data collection

The Raspberry Pi computer was going to be used to control the arm and both cameras for data collection, but it was found that it was not able to control both the overhead camera and the arm through usb. Because of this, a web-server was set up so the Raspberry Pi could send signals to a separate computer to synchronously take pictures with the overhead camera.

The data collection setup involved a workspace mat that the onboard camera was to view and an overhead camera that would be used to find the true position of the arm by localizing the fiducial. This setup is shown in *Figure 7*.

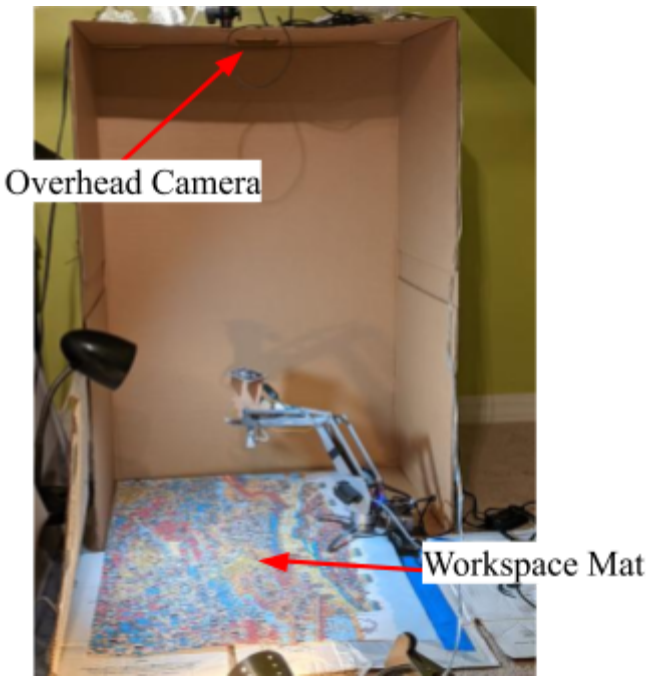

Figure 7: Data collection setup

The first workspace mat used was a "Where's Waldo?" image [3] that was assumed to show enough diversity for the CNN to learn. This image was found to result in low training loss, but high validation loss, so an image with more global structure was generated in the form of a discrete 2D gradient which was used in the final implementation of the system. These two workspace mat images are shown in *Figure 8*.

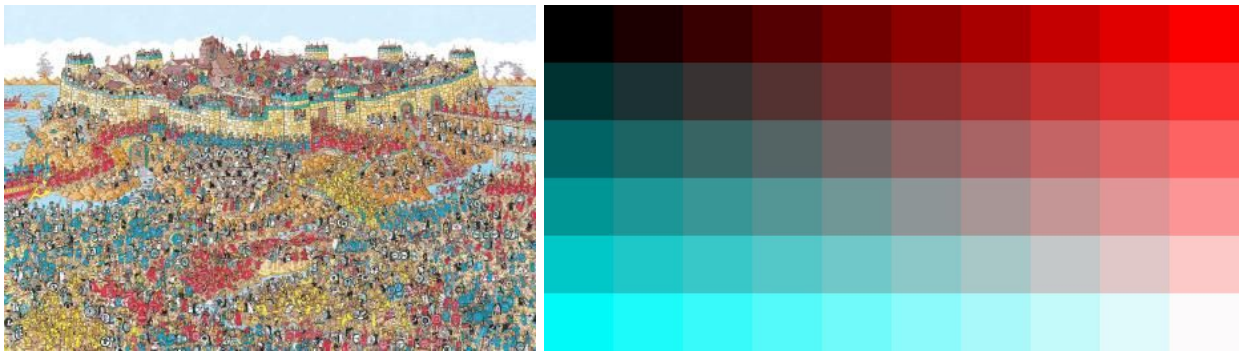

Figure 8: "Where's Waldo" (left) and discrete gradient (right) workspace mat images

The data collection process involved generating random x and y positional control inputs for the robot, and then taking overhead and onboard pictures with the robot in that position.

The collected data was broken into a training set used to train the CNN, and a small testing set not used in training that could be used to simulate typical use. With the CNN able to produce a belief of location within the overhead camera's frame, methods were developed to convert this pixel location to a physical location in the frame defined in *Figure 3*. Histograms showing the error between the CNN's prediction of location and the location detected by the overhead camera for the test data set and training data set are shown in *Figure 9* and *Figure 10* respectively.

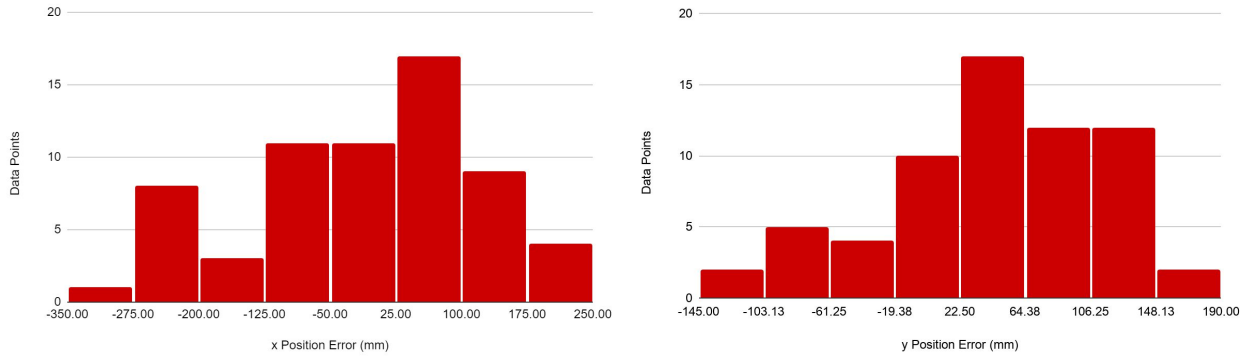

Figure 9: Histograms for CNN estimate error using testing data

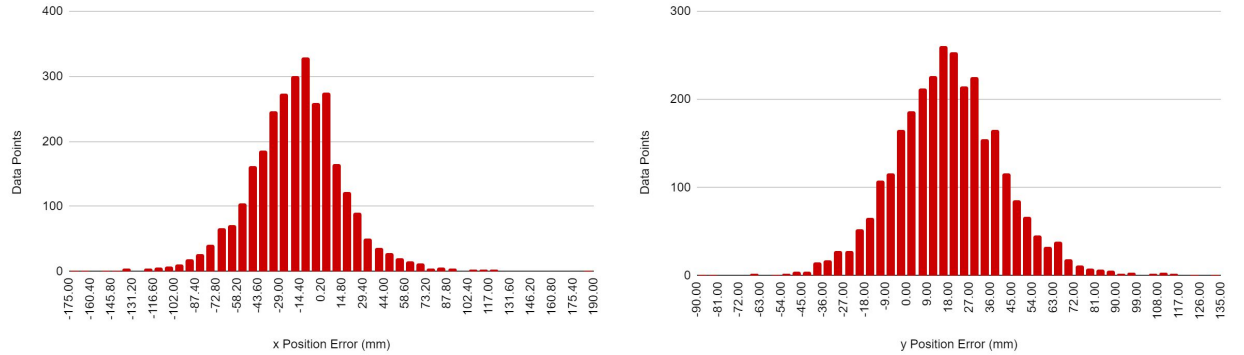

Figure 10: Histograms for CNN estimate error using training data

As expected, the CNN was able to predict locations from the training onboard camera images much more accurately than the testing data. This gap could be lessened by collecting a larger dataset for training. The means and standard deviations associated with the above histograms are shown in *Table 1*. The standard deviations found here were used to inform the filter design.

Table 1: CNN Sensor model statistics

Data set	x position error		y position error	
	mean	standard deviation	mean	standard deviation
Test	-8.154	134.0	43.63	73.99
Training	-16.43	32.58	18.23	23.36

3.3 | Filter Development

Before the data collection and CNN training were completed, a simulation of the filter working on the arm model moving to random positions was created to validate the basic design. This simulation assumed the CNN would return sensor measurements in the form shown in *Equation 3* in which the inaccuracies of the camera are assumed to be zero-mean, additive gaussian noise.

$$\underline{z}_k = \begin{bmatrix} x_k + v_{x,k} \\ y_k + v_{y,k} \end{bmatrix} = I\underline{x}_k + \underline{v}_k$$

$$\underline{z} = sensor\ measurement\ (mm) \quad \underline{v} = sensor\ noise$$

Equation 3: Assumed sensor model

The designed filter is a Kalman filter using *Equation 2*, *Equation 3*, and a 90% measurement rejection gate. The standard deviation of the process noise was set as 12 mm as informed by *Figure 4*, and the sensor noise standard deviation was assumed to be 10 mm. Reasonable values were also chosen for the initial covariance on the state estimate. The results of this initial filter simulation are shown in *Figure 11*.

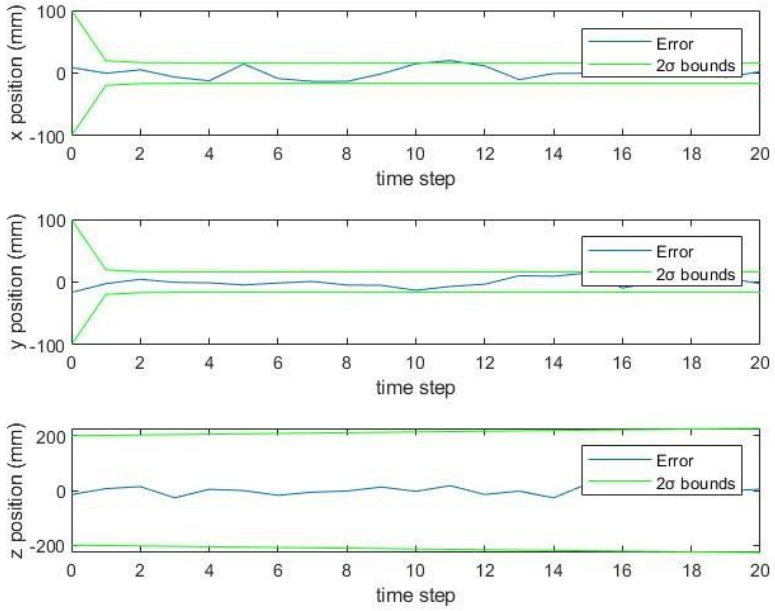

Figure 11: Initial filter simulation results

Figure 11, shows that for the assumed conditions, the filter was able to estimate the wrist position reasonably well and remain consistent for a majority of the measurements. It can also be noted that the uncertainty associated with the estimate of z does not decrease as a result of that state being unobservable by the system. This is acceptable as the z position accuracy is less essential than the x and y accuracies for this application.

To assess the performance of, and tune the filter working on the real sensor model, the data used to train and test the CNN was used in combination with the collected control inputs from the data collection process. The locations found by the overhead camera were used as the ground truth to assess the filter performance.

The filter was tuned first using the test data to assess the performance of the system as it currently exists, and then tuned again using the training data to assess the potential performance of the system given an infinitely large training dataset. The final tuning parameter values are tabulated in *Table 2*.

Table 2: Filter tuning parameters

Tuning Parameter	Filter		
	Initial Simulation	Test Data	Training Data
Process noise covariance (mm)	12^2	30^2	12^2
Sensor noise covariance diagonal (mm)	10^2, 10^2	140^2, 80^2	100^2, 70^2
Measurement rejection gate (mm)	0.90	0.80	0.80

4 | Results

The filter tuned for the training data gave the results shown in *Figure 12*.

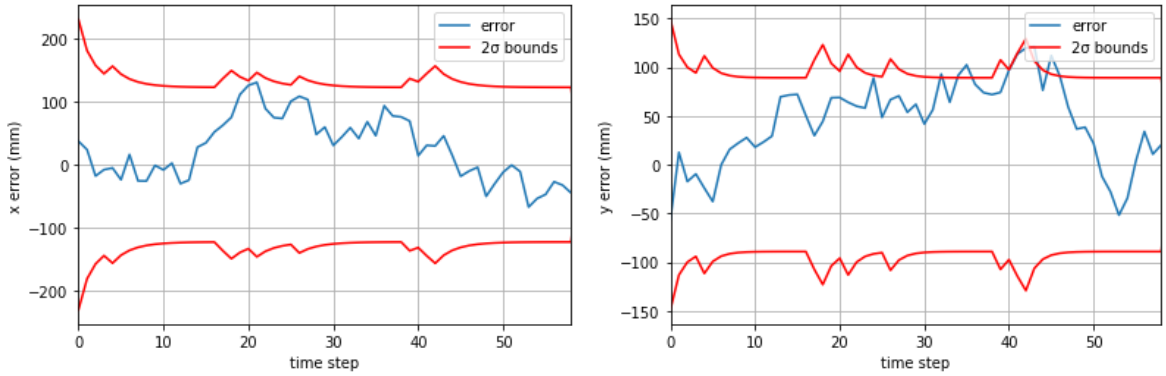

Figure 12: Filter performance, testing data

When looking at the testing data, it can be seen that the filter keeps the x estimates entirely inside the 2σ bounds, and the y estimate is only outside of the bounds for a few estimates. This is promising, but the estimates can be seen to be pretty inaccurate with some of them more than 100 mm away from the true position.

The filter was tuned with the training data gave the results plotted in *Figure 13*.

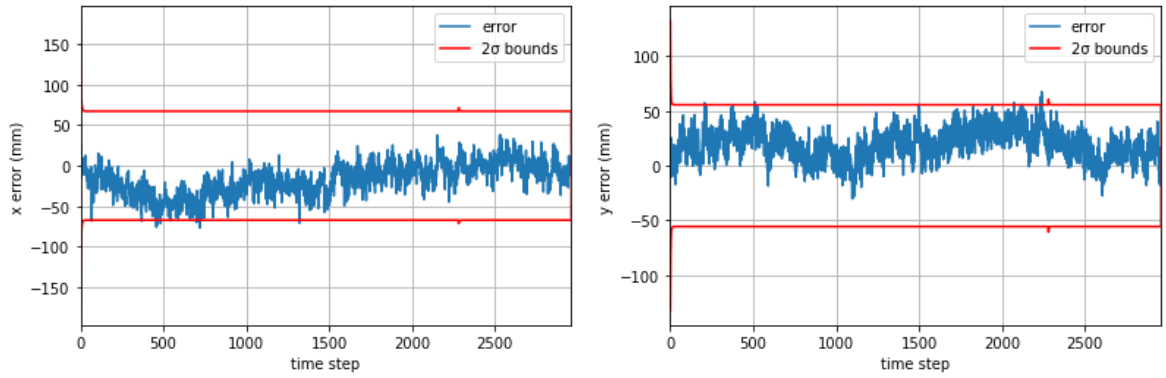

Figure 13: Filter performance, training data

The estimates produced by the filter acting on the training data proved to be significantly more accurate. This is promising for the potential of further developing the CNN to get better estimates.

The overall performance of both filters is summarized in *Table 3*.

Table 3: Filter performance metrics

Data set	x position estimate error		y position estimate error	
	mean	standard deviation	mean	standard deviation
Test	25.30	49.50	43.93	43.18
Training	-16.93	19.16	19.03	14.74

Figure 12, *Figure 13*, and *Table 3* all indicate some bias in the estimates especially in the y position. This is most likely a result of the non-zero mean of the CNN prediction error as described in *Table 1*, but may also be impacted by inaccuracies the overhead-camera-generated locations which were assumed to be the truth. The filter is able to provide more confidence than the CNN alone, but does not indicate more confidence than the

robot's servo controls give. However the filter here is able to provide a location in the workspace rather than just a position relative to the robot's base which is subject shift and move within the workspace.

5 | Lessons Learned

This project showcased many challenges of working with a vision-based CNN. The mat used in the workspace proved to be very influential in the network's performance, and the bias seen in the sensor model described in *Table 1* may be a result of the asymmetric nature of the mat used. The challenge here is to ensure the onboard camera will take distinct images at any location it travels to, but the images must also have enough global structure in order for inferences to be made. It also became evident that in order to train the CNN, a very large set of data is required. The data collected for this project included over 4000 points, but the gap between the training data and testing data performance hints that even more data could dramatically improve the position estimates.

The results of this project also suggested that more complicated platform and sensor models would most likely improve the accuracy of the filter. By ignoring dynamics of the robot, the nonlinear relationships between the position values are lost, and the filter is forced to treat the results of these relationships as process noise resulting in a decrease in confidence. Modeling the process noise as gaussian about each positional dimension is also, most likely, an oversimplification that could be improved. The model chosen for the sensor also fails to account for the bias that is present in the CNN's predictions, and a more accurate sensor model could have great benefits for improving the filter performance. Improving the models for the platform and sensor would most likely require much more data collection on the system to better characterize it as well as a higher order filter.

Additionally, the completion of this project involved many lessons in the use of tools such as AprilTags, virtual environments, several Python libraries, and the required embedded software tools for use of the robot.

To conclude, the method presented here does not provide estimates as accurate as desired, but does offer a way to roughly localize the end effector in the workspace. Further work is to be done to improve this method, and online testing of the filter is to be done on the arm to develop a controller that will allow for positioning within the workspace.

6 | References

1. Sparkfun.com. (n.d.). UArm Metal - Desktop Robotic Arm [Digital image]. Retrieved November 22, 2020, from https://www.sparkfun.com/products/retired/13663

2. APRIL Robotics Laboratory, University of Michigan. (2010). AprilTags Visual Fiducial System. Retrieved November 22, 2020, from https://april.eecs.umich.edu/software/apriltag

3. D. (2016, August 22). Where's Waldo [Digital image]. Retrieved November 20, 2020, from https://medium.com/@realbrickroad1/wheres-waldo-40e77172fa2f