**Kinematic Data Analysis: Rough Draft**

This Process is a walkthrough of the functions of the code and maybe so explanations in parts. I would appreciate feedback or questions about clarity and thoroughness so we can all understand how this works before we add it into the report.

The csv files are stored in a folder called Data in the project space.

The names of the files are used to categorize the final data

The data is pre-processed to organize relevant information from the data as well as make analyzing it easier later in the pipeline

Notation: ‘names’ in quotes as values in the dataset. *Names* in italics are variables in the program. **Names** in bold are libraries used in the program

**Pre-Processing:**

The **pandas** library is utilized for its ease of use with large data set

The **numpy** library is used for its power math engine

‘timestamp’, ‘rx-green’, and ‘ry-green’ are pulled from the dataset and collected into a new dataset called *puck\_data*

We calculate a column of delta values based on ‘timestamp’ called ‘delta’. This is done because the time delta is frequently used throughout the rest of the program

‘delta’ is used with the deltas of ‘rx-green’ and ‘ry-green’ to find the velocities ‘vx-green’ and ‘vy-green’ which are then used to find acceleration ‘ax-green’ and ‘ay-green’. This is done because I’m not certain about the reliability of the captured velocities and accelerations.

Now the interesting part. We need to find a way to split up the test as they were recorded in one take. This can be done by leveraging the fact that the puck is at rest for a moment before the test, and then luckily falls out of frame after the test.

Leveraging this we need only find the last zero velocity before the position stops being captured to pull out the test data into pieces.

These pieces are then collected and passed to the next stage of the program

**Filtering:**

The Kalman Filter is an extensive system that would require more than 3 pages to fully explain I will briefly touch on what I believe the be the most vital components, but if you needed more information, I would recommend researching it at:

<https://www.kalmanfilter.net/default.aspx>

<https://www.youtube.com/watch?v=CaCcOwJPytQ&list=PLX2gX-ftPVXU3oUFNATxGXY90AULiqnWT>

The Kalman filter is a discreet filter meaning it works on a defined time step (Not Continuous) This time step does not need to be constant, but it does need to be known.

**Step 1 (Initialize):** Take initial measurements to start the system. You will need to measure the current *state* of the system as an N\*1 matrix and find the measurement covariance. This is an N\*N matrix which defines the relationship between the variances of each value of the *state*. Finaly define a matrix called the *Model* which states how to system should change given a certain time step.

**Step 2 (Predict):** Multiply the *Model* based on the time step by the current *State* to get the predicted state called *pState.* Then multiply the *Model* by the covariance then by the transpose of the *Model* to get the predicted covariance called *pCov.*

**Step 3 (Update):**

Now your current understanding of the system probably is inaccurate, but we have new data to add to our current understanding. So how should we integrate this information? Well luckily for us those covariance matrices we have been calculating can tell us the uncertainty of the measurement. This along with the fact we have an equation to calculate the next covariance matrix means we can use some calculus to minimize the uncertainty.

By formatting the result as the optimal mix between our current estimate and the new data we can simply find the difference between the data and our estimate called correction and multiply it by the calculated mix called gain, to adjust the estimate.

As you can see in the code the covariance update equation is a lot more complicated than the estimate update equation. If you wish to get a better understanding of it, please reference source 1. Oversimplified, it is the same optimization equation for state, but instead soled for covariance.

**Step 4 (Repetition):** Steps one and two are repeated over the rest of the data set and the estimate and covariance at each step if added to a column for graphing purposes.

Finally. The X and Y components are uncorrelated and processed separately. Just do everything for X then do it for Y. The data is the merged to get the filtering results

**Post Processing:**

Relevant data is collected and written to a csv using pandas.