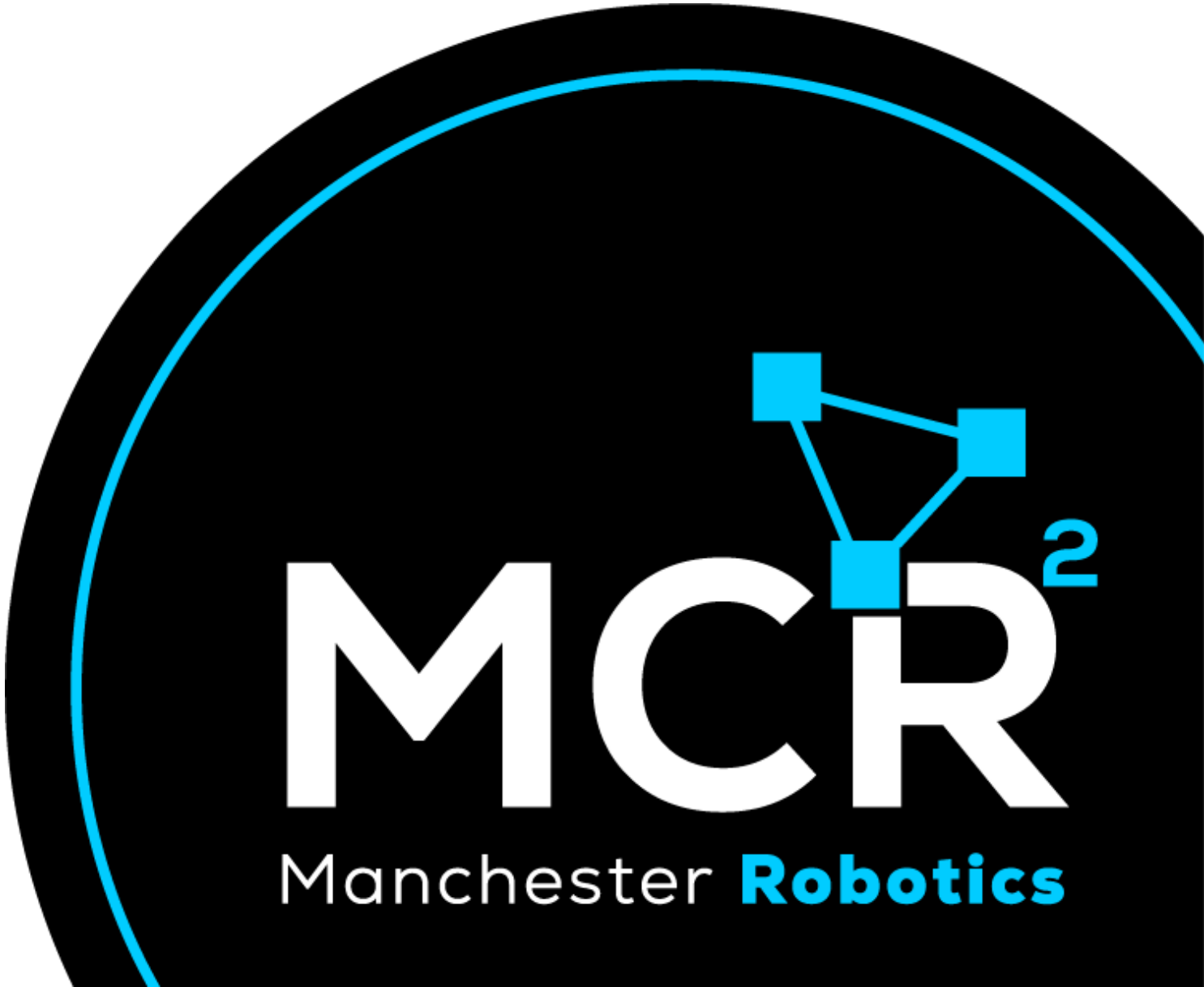
{Learn, Create, Innovate};

Line Following

Line detection and following using OpenCV and ROS



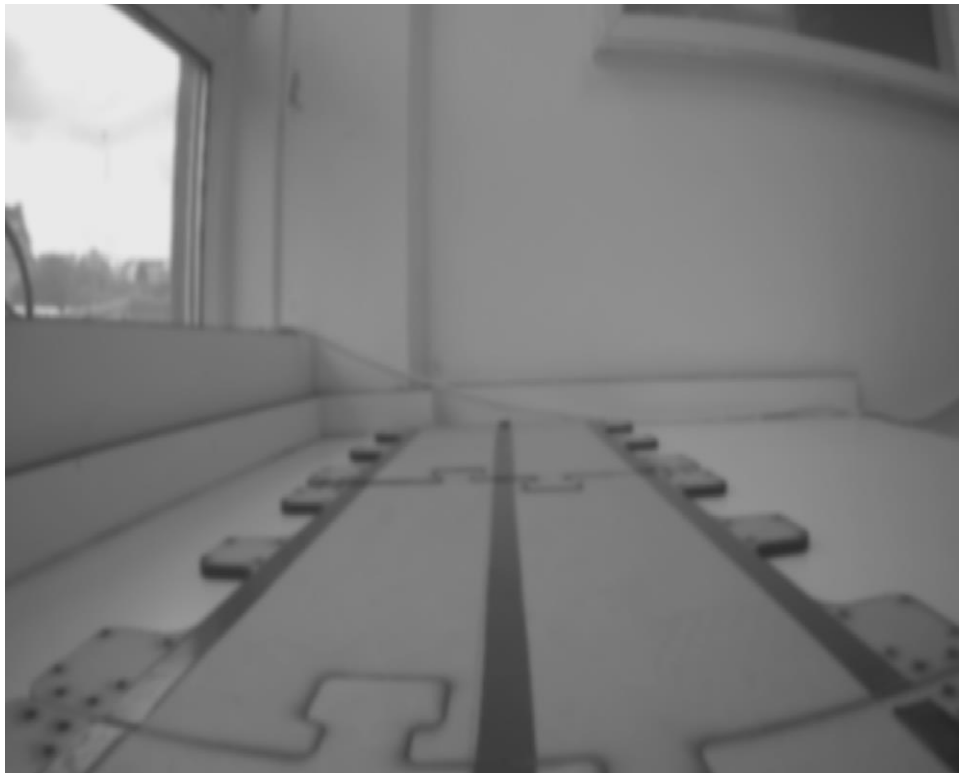
The Task

Line Detection

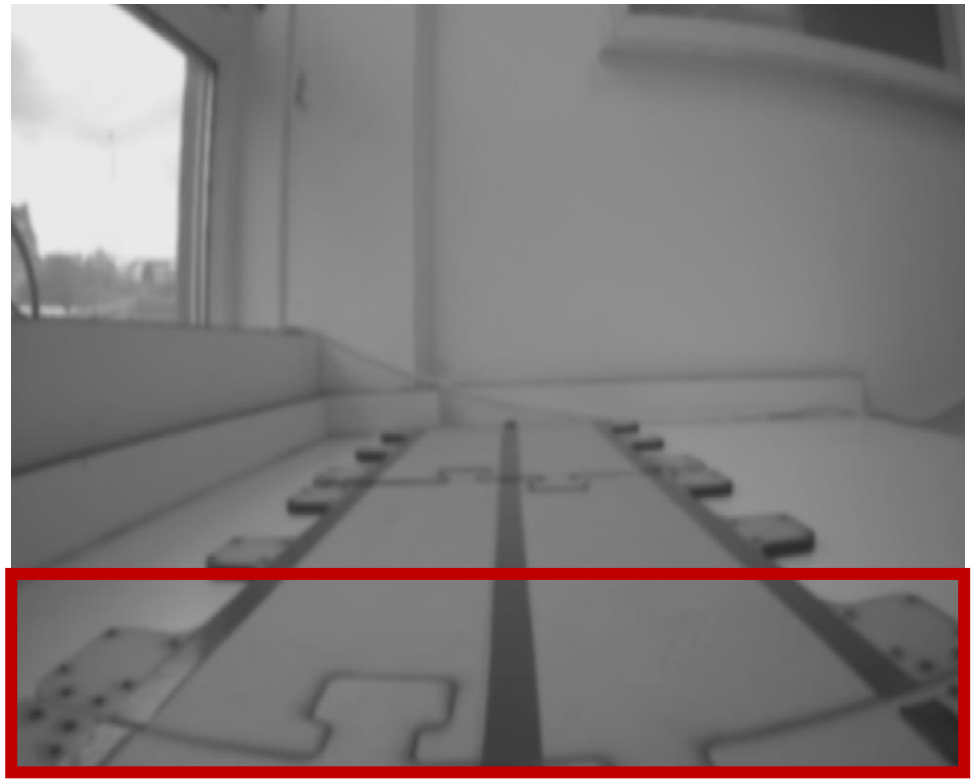
→ ???

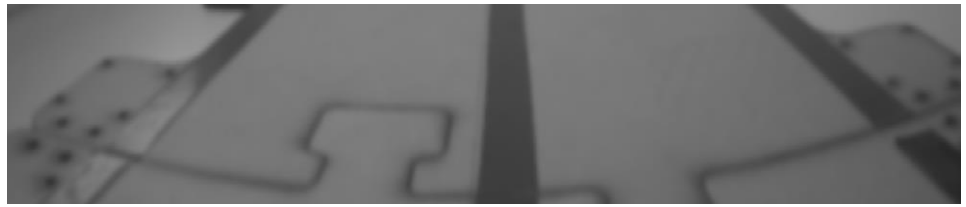
Pre-processing

- Rotate image if necessary
- Reduce image size
- Change image encoding if desired
- Remove noise
  - Gaussian Blur
  - Dilate and Erode



Region of Interest





Convert to graph

- The picture is a matrix of values between 0 and 255
- Each row will have lower values around the line

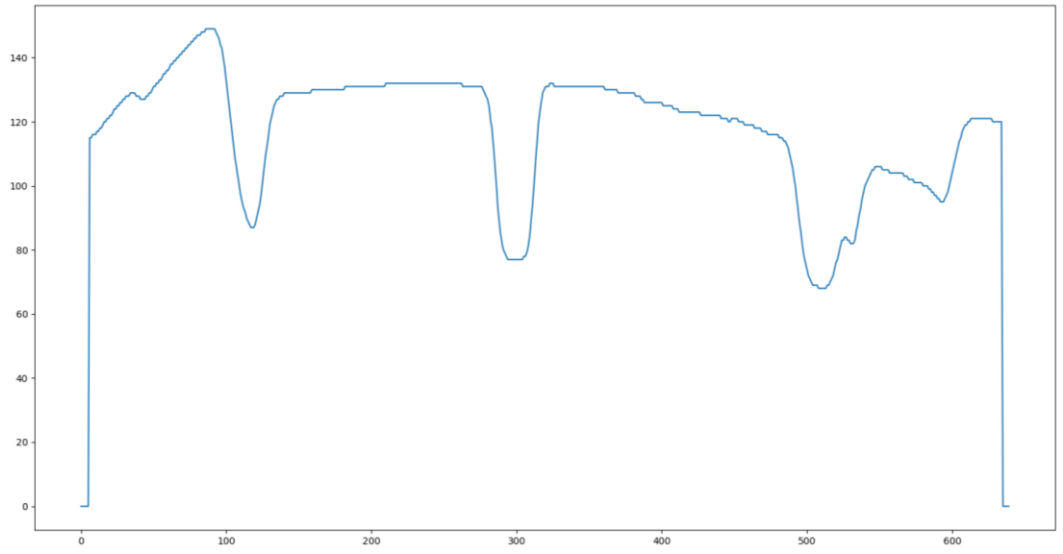
3	4	151	⋯	112	2	3
5	1	175	⋯	158	3	6
1	2	147	⋯	145	31	2
⋮	⋮	⋮	⋱	⋮	⋮	⋮
3	5	163	⋯	122	41	10
18	12	189	⋯	168	15	11
2	14	74	⋯	185	19	5

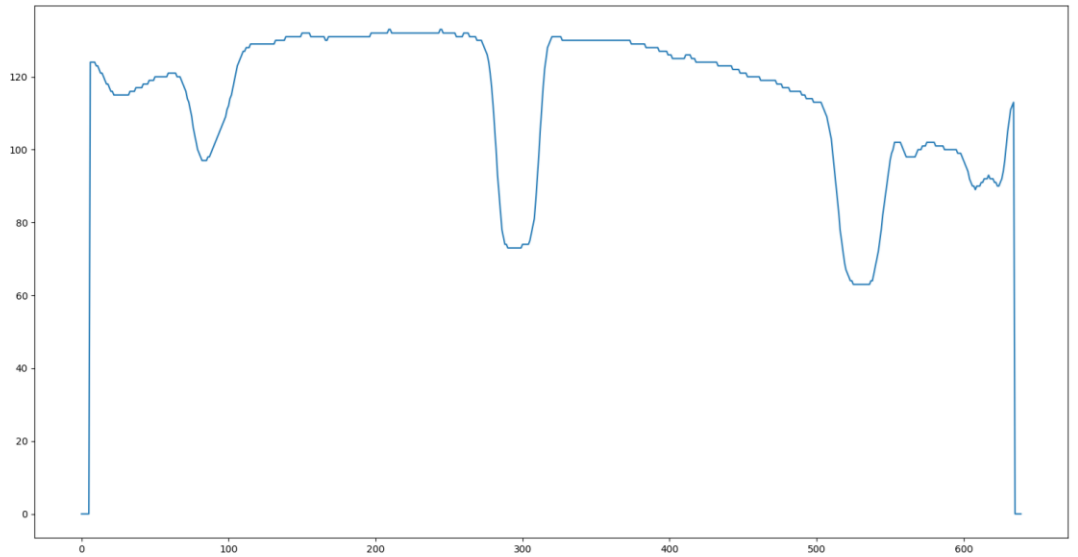
Image height

Image Width

Convert to graph

- The picture is a matrix of values between 0 and 255
- Each row will have lower values around the line

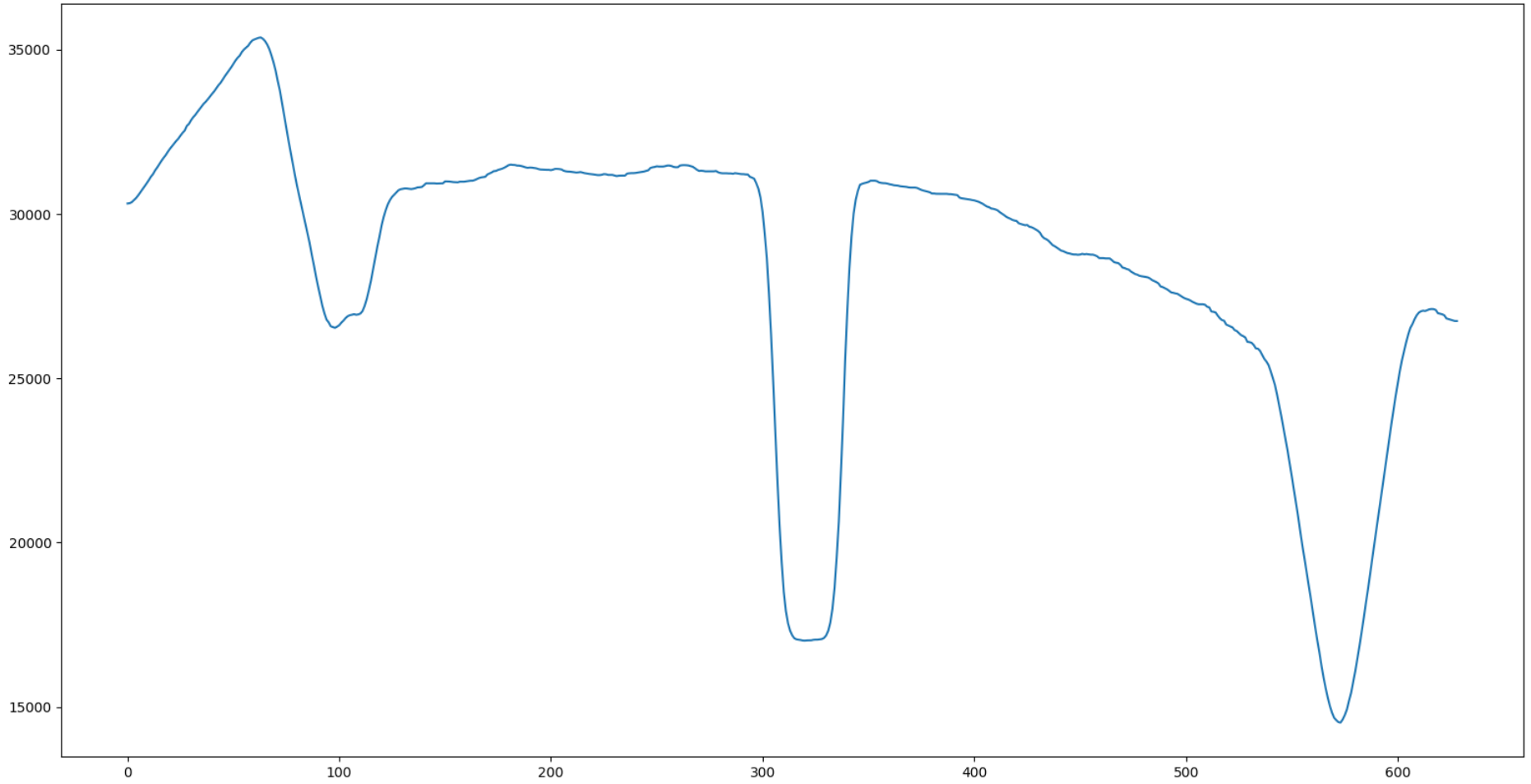




Convert to graph

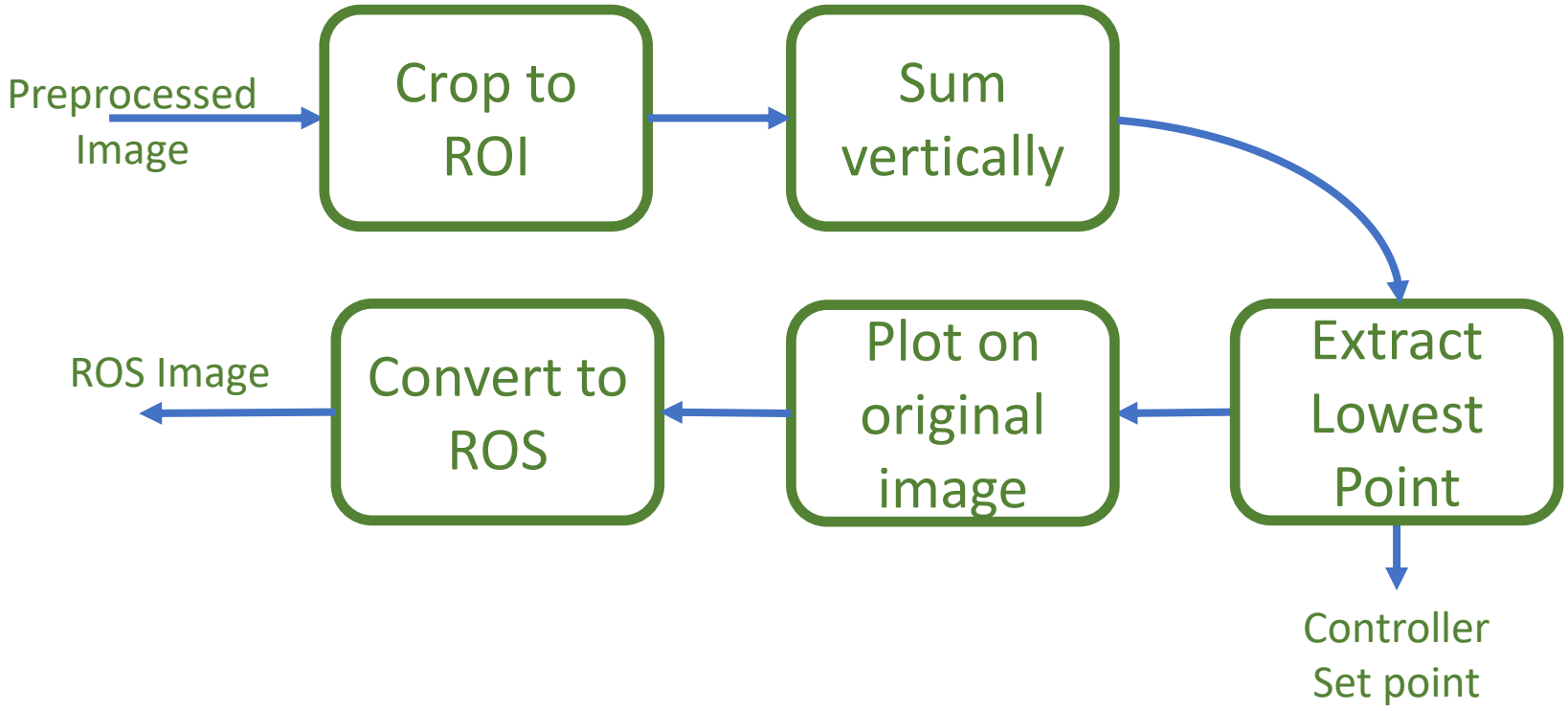
- The picture is a matrix of values between 0 and 255
- Each row will have lower values around the line
- Exaggerate this by summing vertically
- This is easily done using numpy

Convert to graph



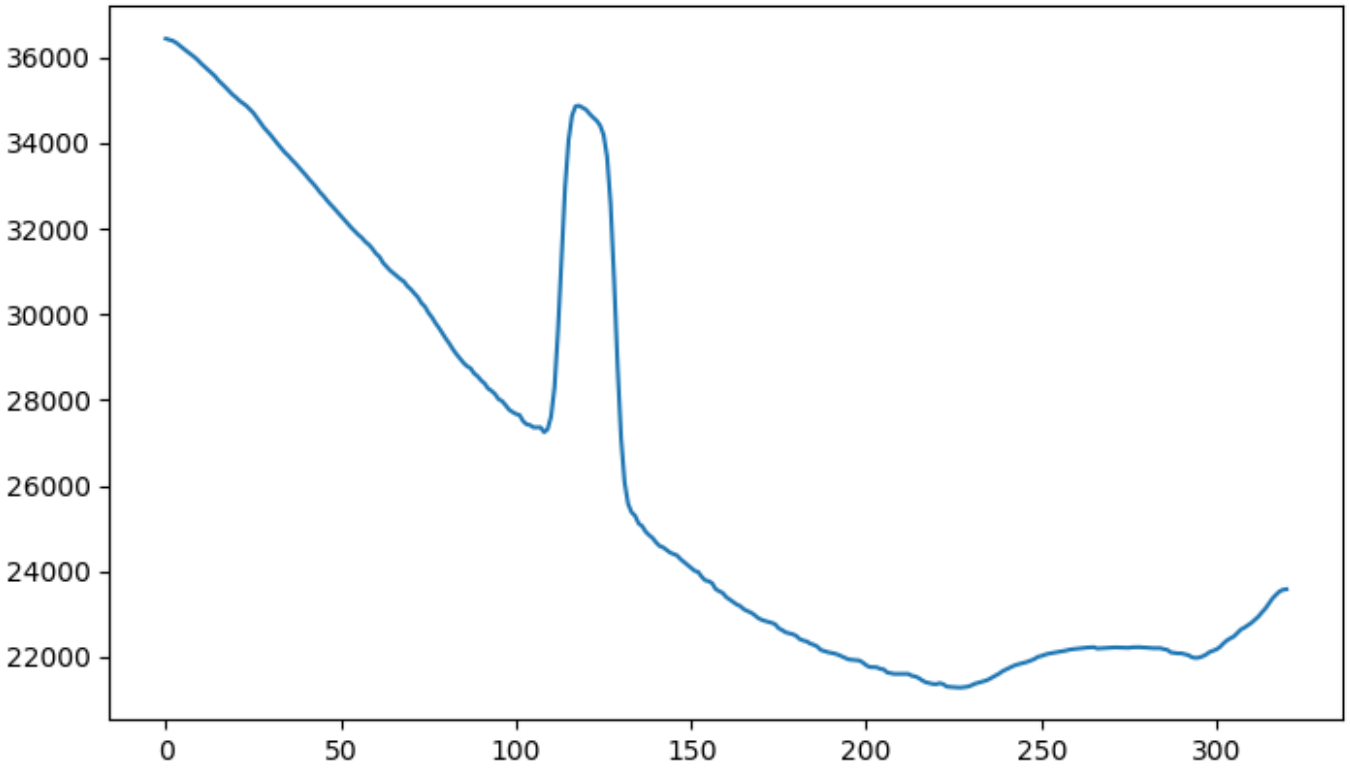
Activity: Minimum Value

- The simplest solution is to just use the trough



- Problems with this solution?

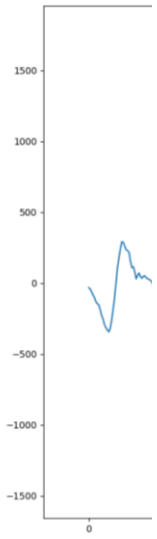
Activity: Minimum Value

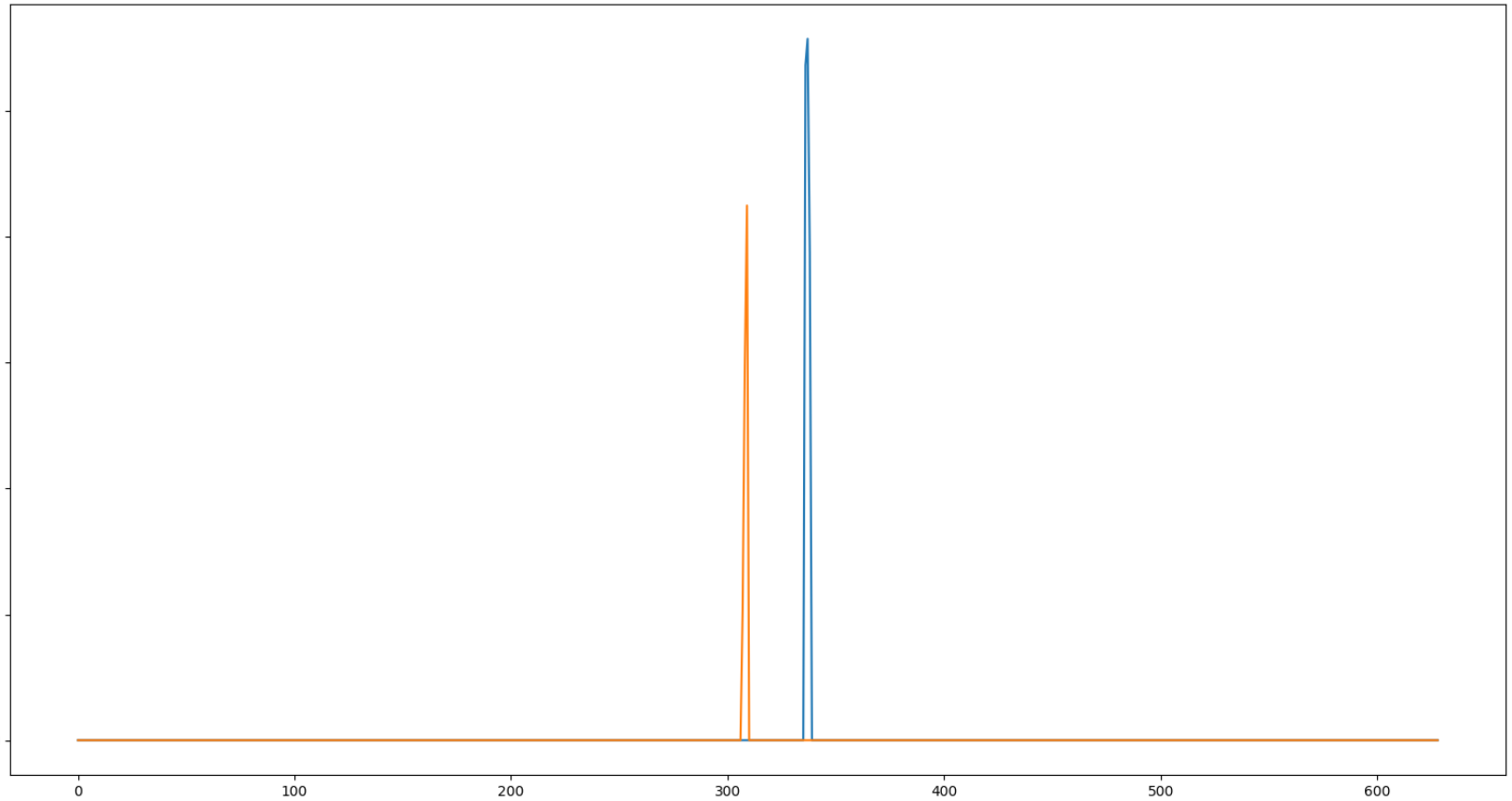


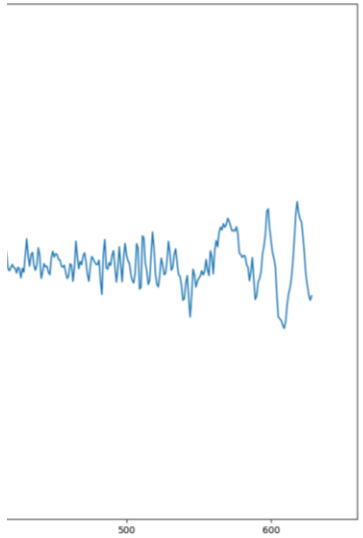
- Problems with this solution?

Increasing reliability

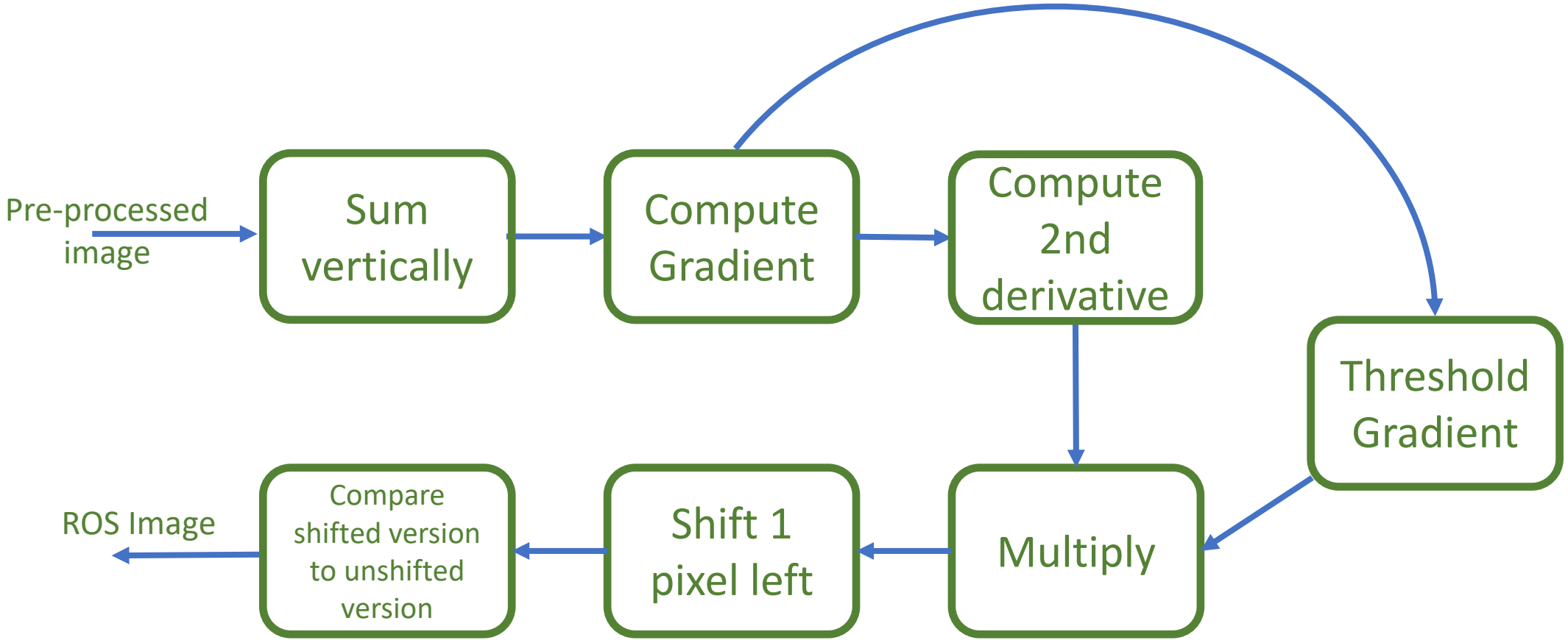
- If c
- I Is







Increasing reliability



Activity: Edge Detection

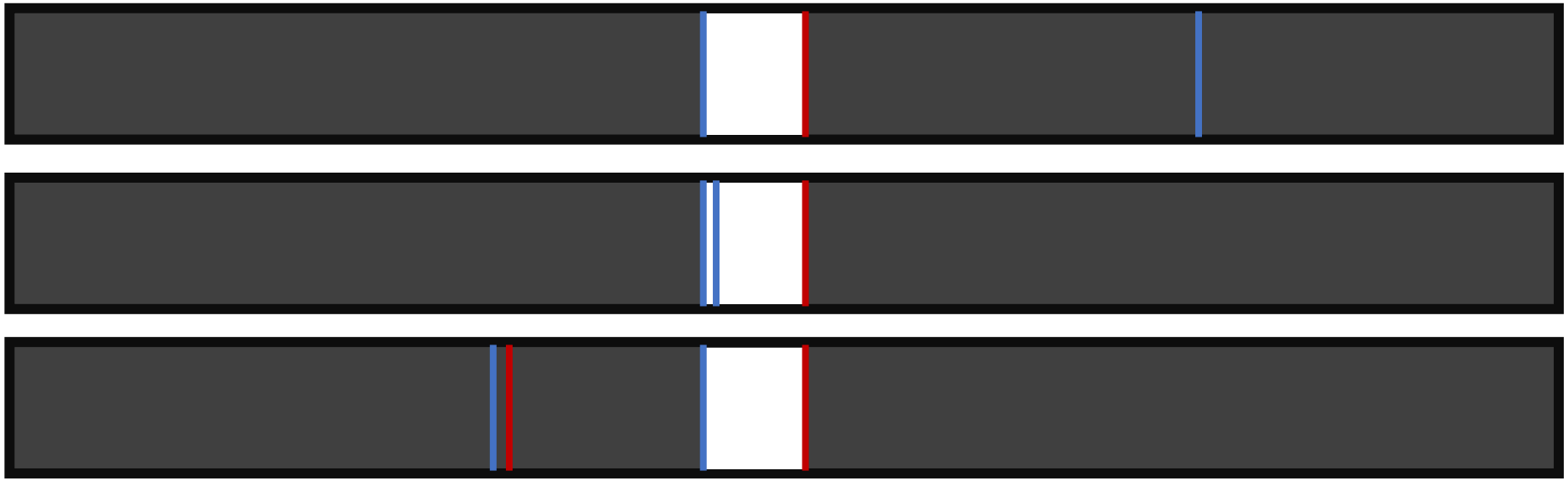
- Using a method of your choice, write a node that can observe the lines on the track and output an array of left and right edges

```
left_edges = np.array([4,311, 623])
right_edges = np.array([10,324,634])
```

- You will likely find that many methods are still imperfect, as noise can add phantom edges. Try to achieve the best performance by fine-tuning your thresholds and other parameters.

Further processing

- Every left edge must have a right edge associated with it
- Consecutive edges should be of alternating types
- The line has a maximum and minimum width
- Exception handling is essential



Exception Handling

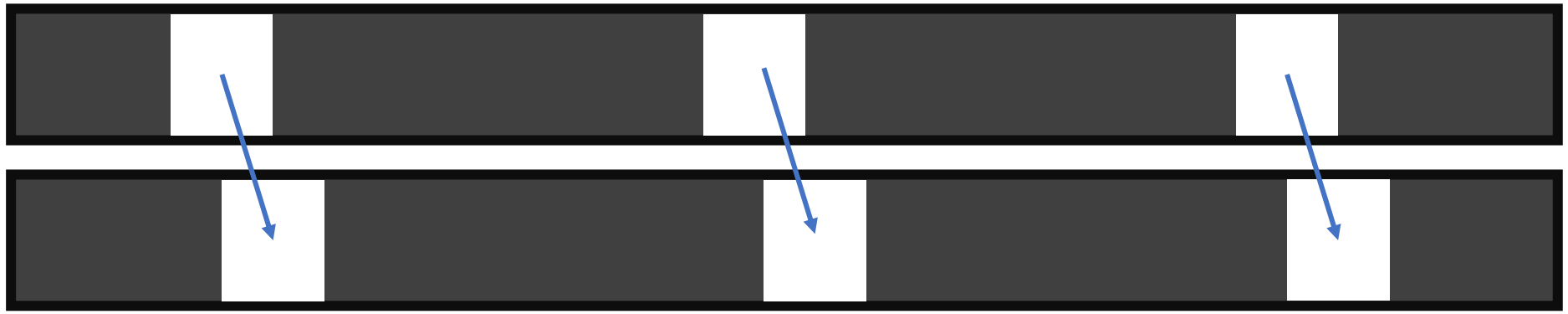
- If we try to subtract our `left_edges` vector from our `right_edges` vector to determine the centre point of the line, numpy may throw an exception
  - `[x,y,z] - [a,b]`
  - `[x,y] - [a]`
- You should build exception handling into your code, such that an error value is return if a value for the centre of the line cannot be found
  - NaN – `np.nan()`

Returning a value

- We want a single value to be input into our controller: how far the robot is from the centre of the track.
- We can determine the centre-point of the line and the width of the line easily
  - `lineWidth = left_edges - right_edges`
  - `linePos = ((right_edges + left_edges) / 2.0) - (frameWidth / 2.0)`
- Line width can be used to reject noise
- Line position can be used as the input to our controller

Multiple Lines

- The track has 3 lines
- The easiest method of keeping track is to associate each line in the frame with a line in memory



- Associate some properties with each line:
  - Which line (centre, left, right)?
  - Position
  - Distance between neighbouring lines

Activity: Better Line Detection

- Implement a selection of processing methods to determine where the centre of the track is.
- Your code should output a single value, in pixels, where 0 indicates the camera is centralised with respect to the track
- It should also output an image with rectangles drawn over where lines are detected. Use the width of the lines and their position to locate the rectangles

The Controller

- A PID controller should be sufficient
  - Input is our line position as determined earlier
  - Our control variable is ω
  - v can just be fixed for now. It will be controlled by external factors such as traffic lights and signs.
- Error handling
  - We also need the controller to be able to cope when our line detection fails, and outputs NaN
  - Suggested method, allow a fixed amount of time where the controller continues at its current speed and turn rate. After that time has elapsed, stop.

Activity: Line following

- Implement a controller such that your robot can follow the track
- The forward speed can be fixed for now.
- If your robot cannot find a line, it should stop.