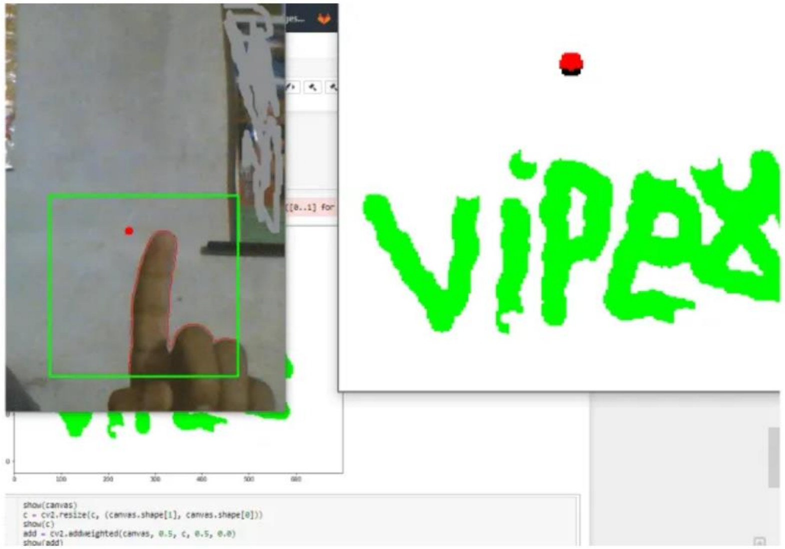
Robotics Security Project:

Using Gretchen to Draw

Idea

- Gretchen can detect shapes, colors and objects
- Can be used in order to save and track the object
- Could this be used to draw with your hand / pen?



```
show(canvas)
c = cv2.resize(c, (canvas.shape[1], canvas.shape[0]))
show(c)
add = cv2.addWeighted(canvas, 0.5, c, 0.5, 0.0)
show(add)
```

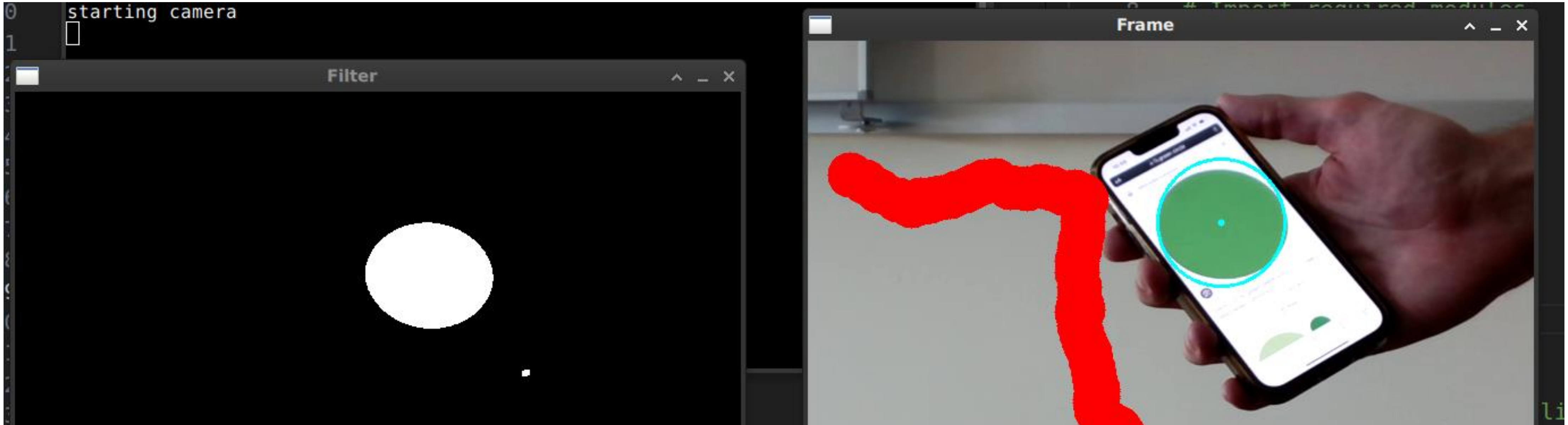
Basic Implementation

Using the example_ball_detctor.py from the lecture as a proof of concept in order to draw

+ It works

- Gets increasingly slower and only draws when a ball is detected

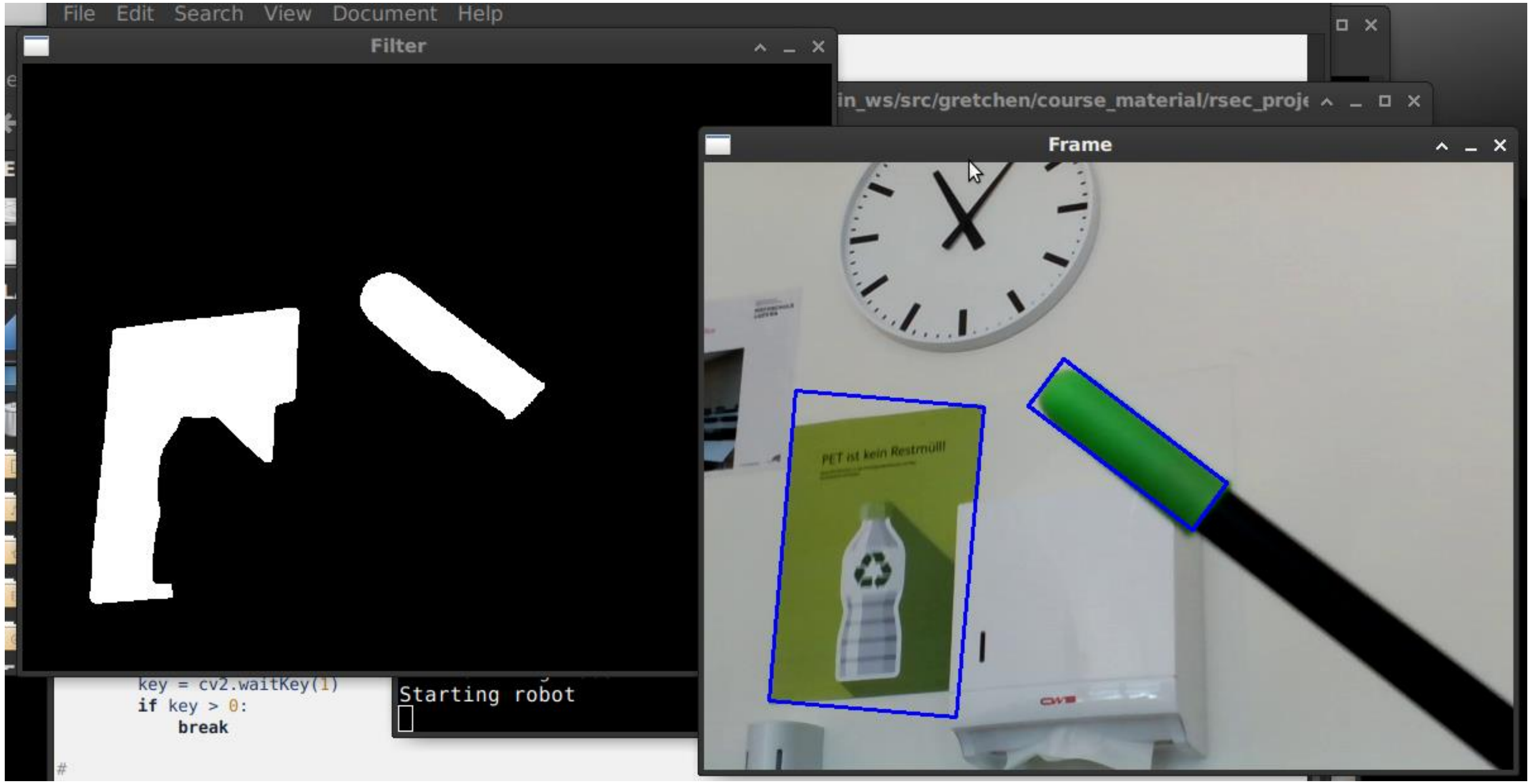
-> The problem was that our *for* loop kept redrawing all the points stored in the array



```
starting camera
```

From Green Ball to Green Pen

- Further changes to example_ball_detector.py to detect different shapes and potentially any shape in a bright green color (green is most distinguishable)

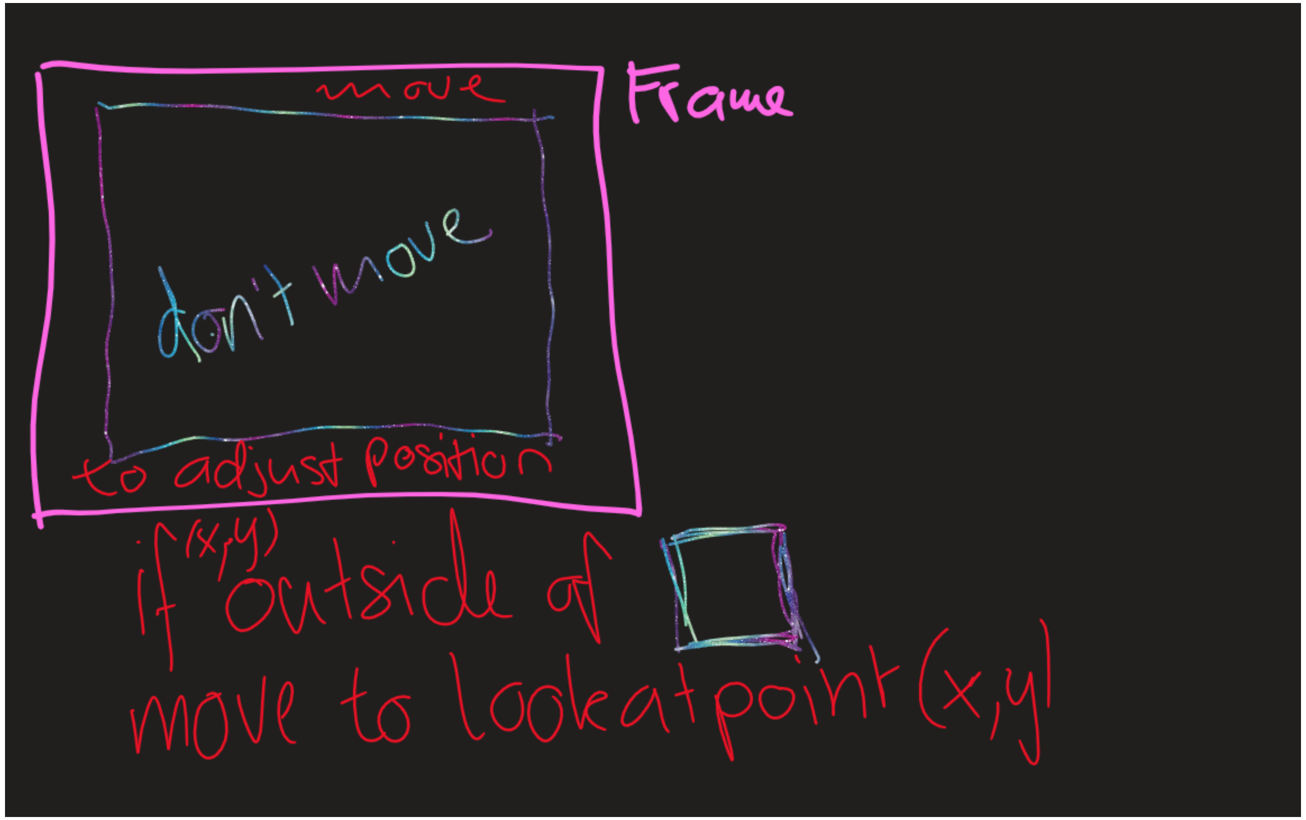


```
key = cv2.waitKey(1)
if key > 0:
    break
```

```
Starting robot
```

Utilizing Gretchens Actuators

- Another feature should the ability for Gretchen to move if the pen reaches the outer edges of the frame



Detection Range

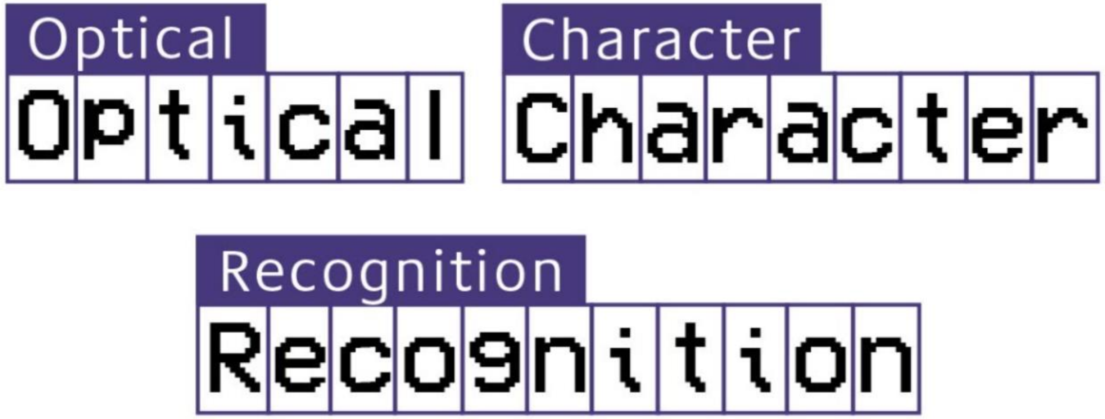
- Approx. 2 meter depending on background
- Can potentially be used to draw on a real whiteboard and digitally record it at the same time
- Using a different color may extend range independent of object size

Text Detection

- Either training our own weights using handwriting (takes time)

OR

- Using the text detector library PyOCR from lesson 06 – Additional Libraries

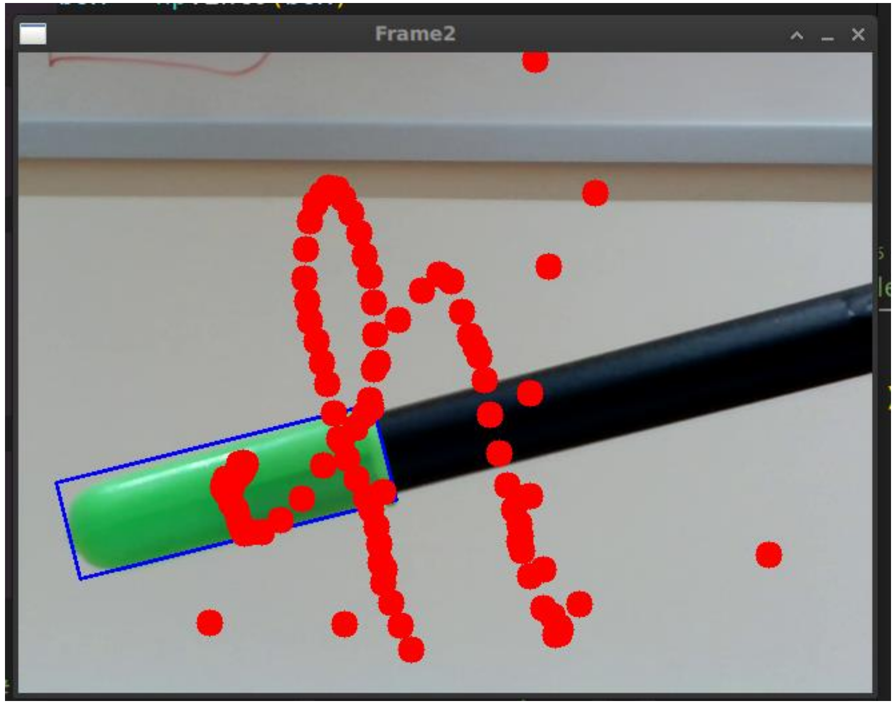


Drawing Optimization with Canvas

To optimize the drawing process the 3 RGB channels are processed into a single channel. The canvas can be seen as a matrix of 1s and 0s.

This is combined with the rectangle detector.

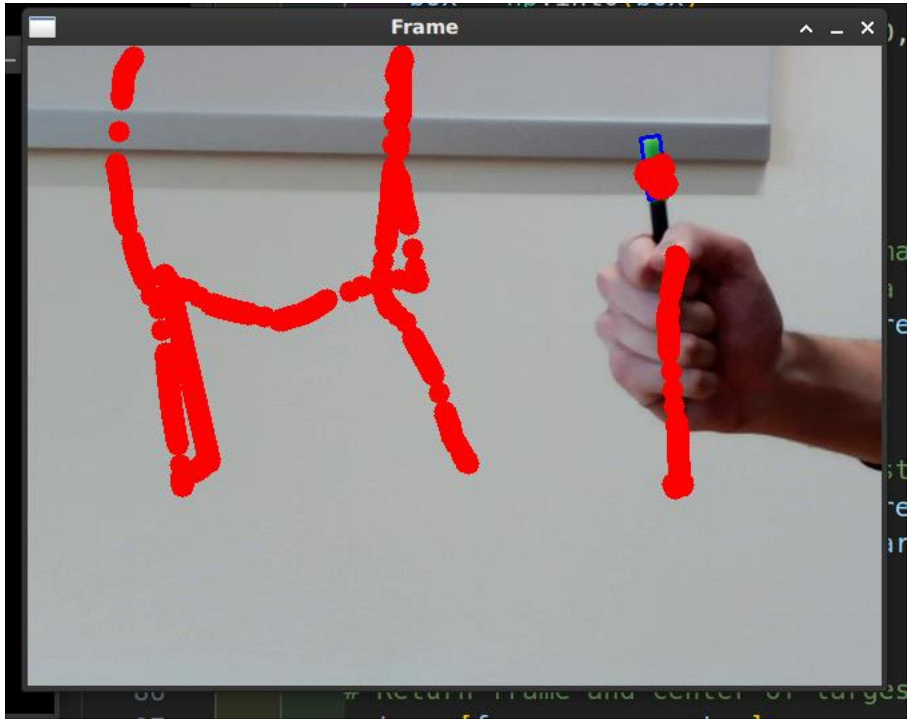
+ Enables „drawing“ with a pen
+ Faster means more responsive with smoother edges



More Polish

Using cv2.line() to draw lines between two points in order to fill the gaps. The points are stored in a queue[] of two. Further optimization by moving parts of the drawing process outside of loops.

+ Consistent enough to write
+ More fluidity and responsiveness

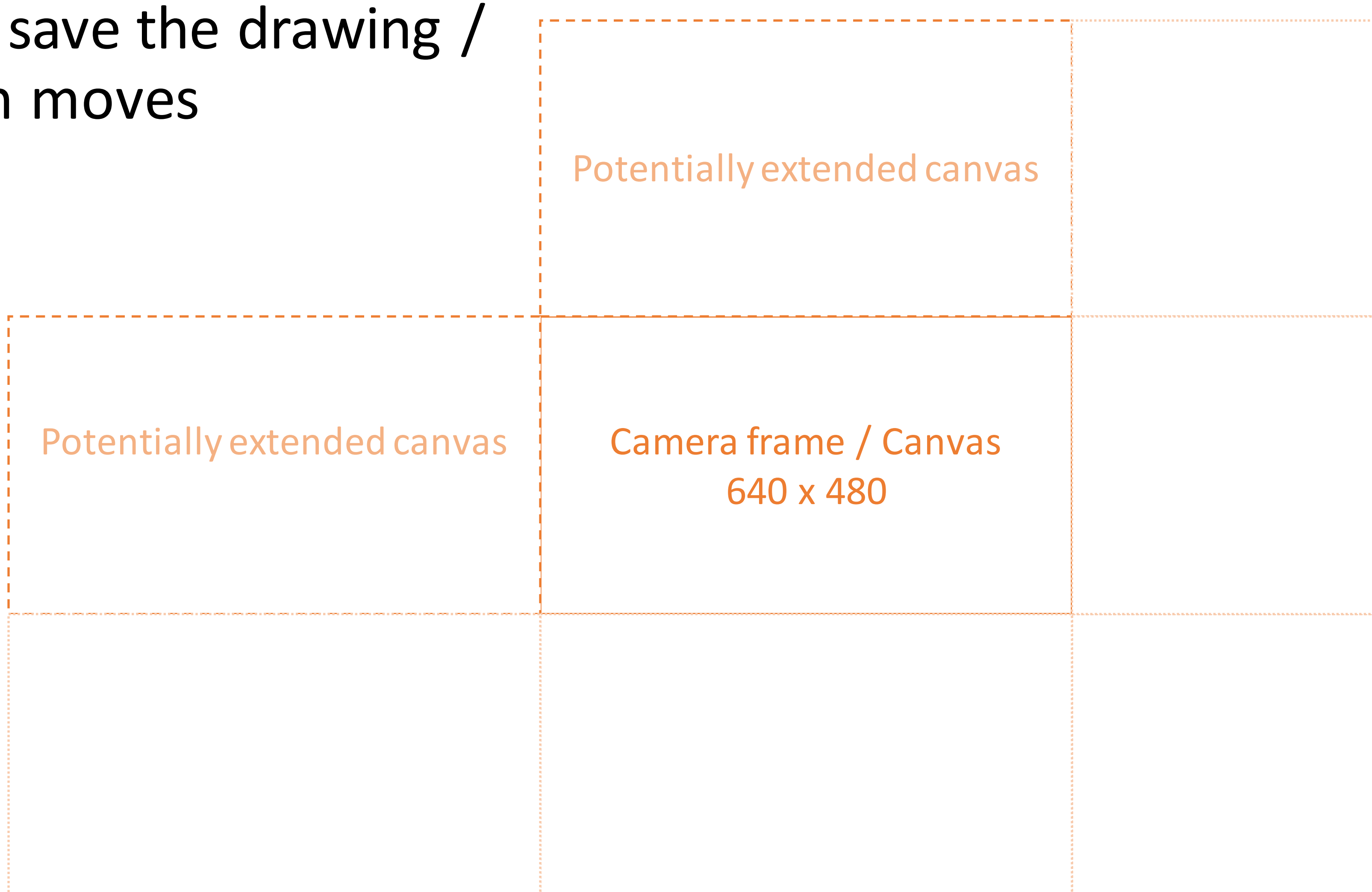


Optimization Problem?

- How to make it smoother?
- How to enable the user to move the hand faster and more freely and still have consistent results?

Expanding and Saving the Drawn Canvas

- It should be possible to save the drawing / writing even if Gretchen moves



Potentially extended canvas

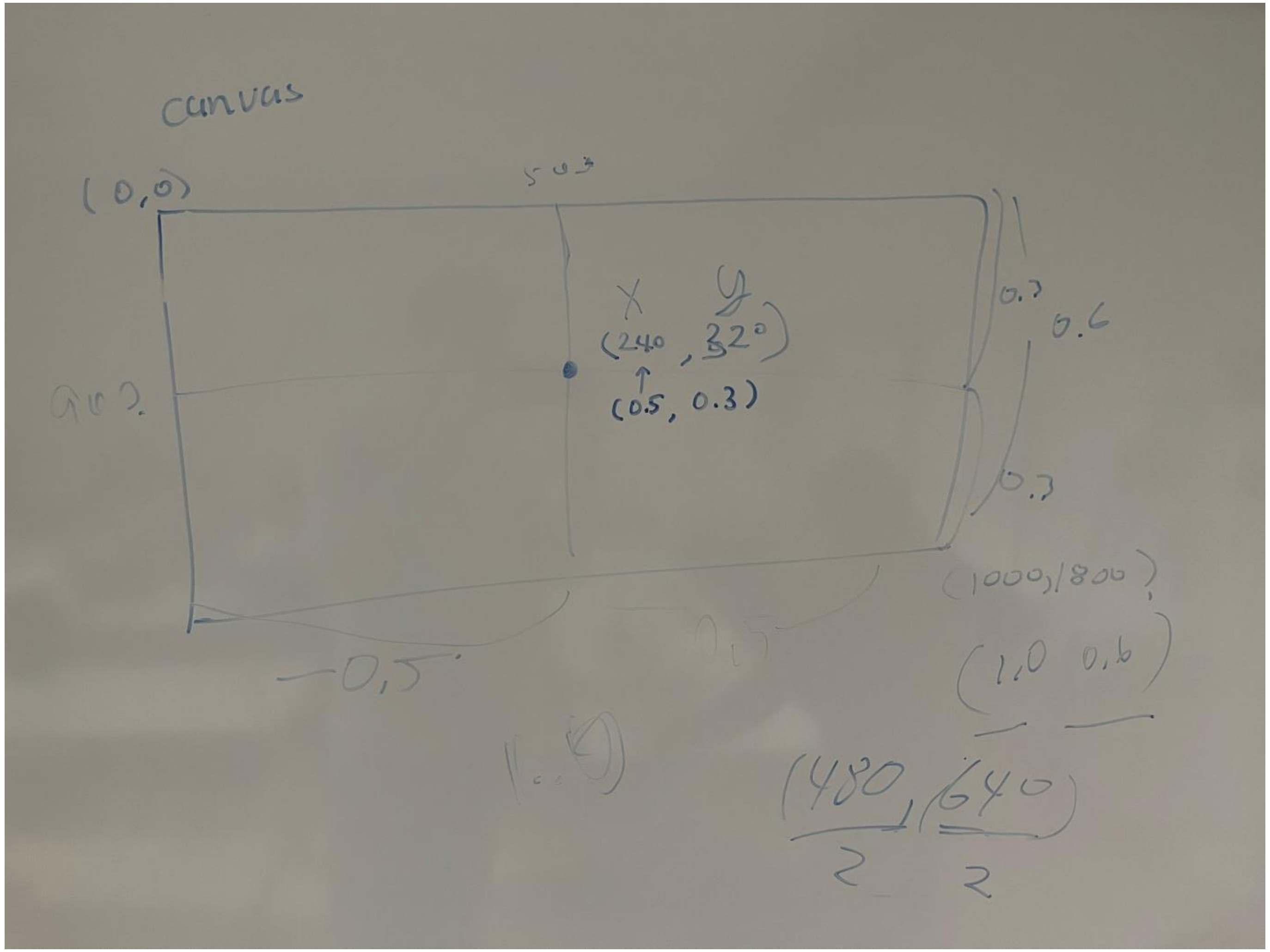
Potentially extended canvas

Camera frame / Canvas
640 x 480

Current State

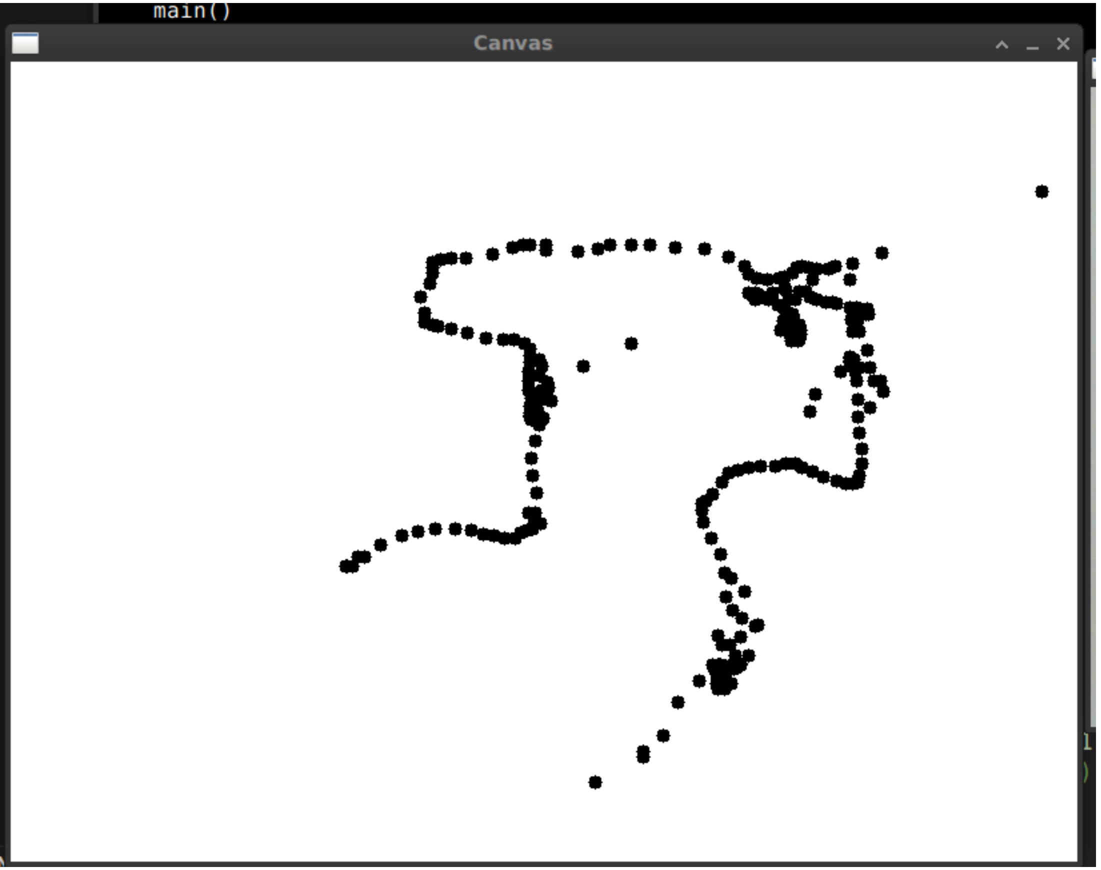
Translating Gretchens coordinates into the canvas coordinate system and centering it in the middle

-> It should draw when the camera is stationary and stop when the camera moves

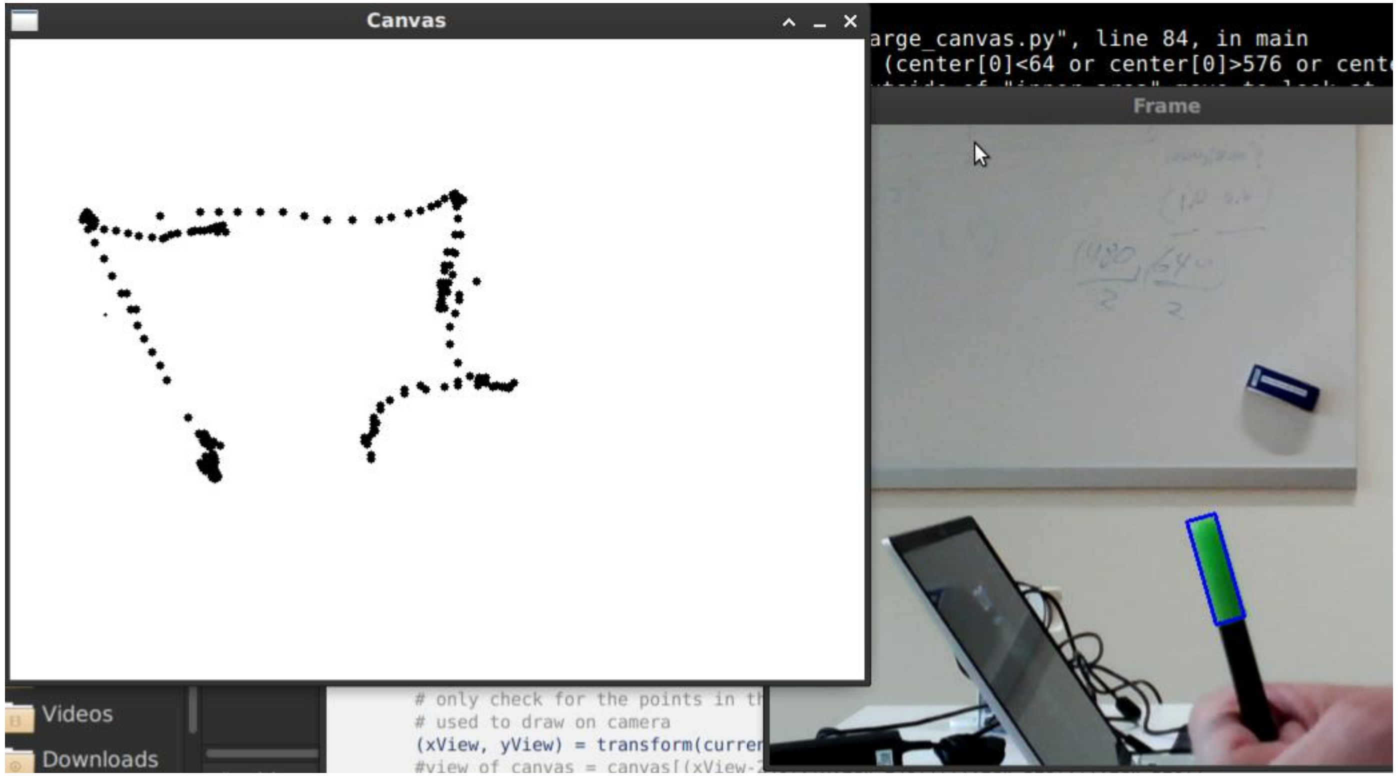


Current State

New feature: A larger canvas, the ability to draw beyond the camera frame and tracking without losing progress, saving of the drawing as png



```
main()
```



```
arge_canvas.py", line 84, in main
 (center[0]<64 or center[0]>576 or cente
```

```
# only check for the points in th
# used to draw on camera
(xView, yView) = transform(curren
#view of canvas = canvas[(xView-2
```