In this lab you will explore directed navigation.

This week you are using the camera on your laptop (or if you are running linux you can use the USB camera). **Be sure to make a new copy of your existing code to work from so that you can revert back to Lab 3/4 code for putting the camera back on top of the robot.**

There's no new starter code this week - keep building from last week's code…

During all lab work
You should keep notes that includes any data you collect or monitor during the lab -- for example, thresholds you decide to use in order to detect whether the robot is facing a white floor or the darker under the table space. In addition, this is a great place to include reminders to yourself -- commands you want to remember to copy-and-paste in the future or errors you don't want to make in the future! This section should be maintained and updated during the lab -- it can supplement any paper-and-pencil recording that you do.

Before you begin, you may wish to read the last section Submit so that you know what you'll be turning in. NEW SUBMIT REQUIREMENTS AT THE END.

Video Capture

Modify the video capture process in the video thread `init` and `run` functions respectively:

```
self.cam = cv2.VideoCapture(0)

retValue, image = self.cam.read()
```

You should now get video from the camera plugged into your laptop to appear. All of your color detection should still work.

Clicks

Your goal is to build a robot system that can successfully navigate to wherever you click in an image (well, anywhere accessible on the floor!)

Identifying and 'locking on' to specific colors will be used to track the position of your robot. In this lab, we will be enabling your robot to move to any position in the camera's view of the floor.

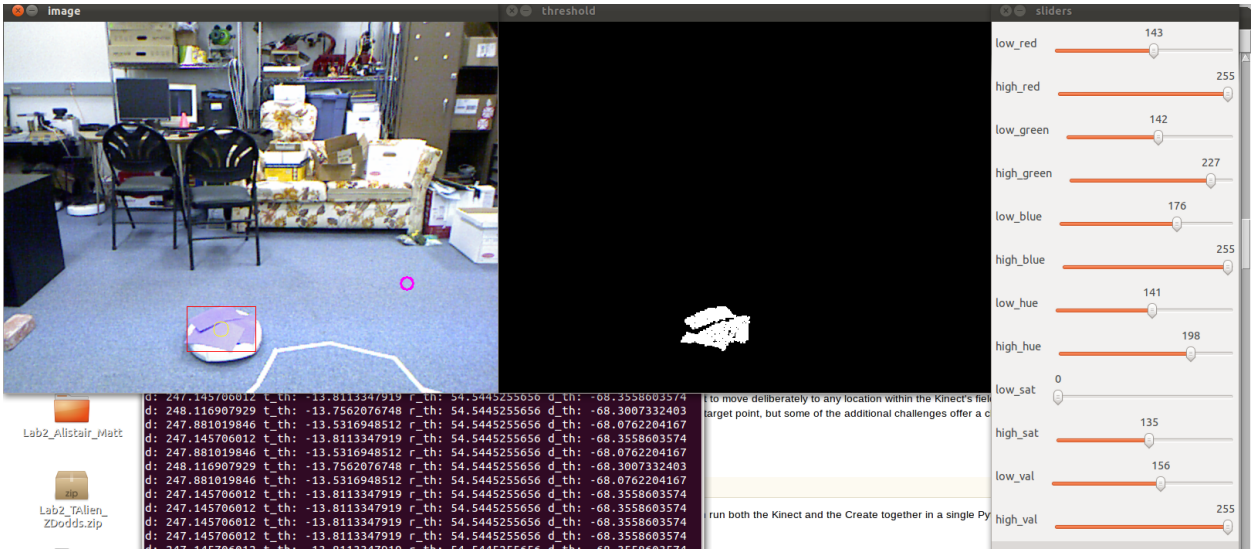
Here is a hint for catching mouse clicks in the main and inside the video thread class respectively:

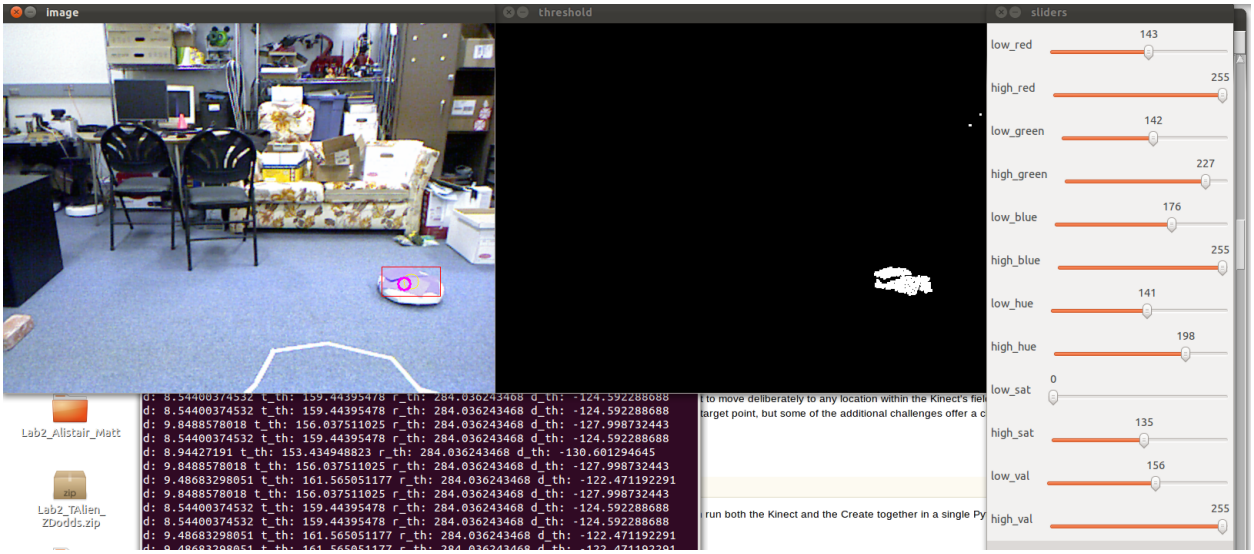
```
cv2.setMouseCallback("Create View", sm.video.click, sm.video)

def click(self, event, x, y, flags, params):
        if event == cv2.EVENT_LBUTTONDOWN:
```

Goal: You should draw a circle where the click was sent.
Optional Goal: You should update the color thresholds to +/- some amount of the right-clicked pixel or add that color as a valid value.

Planning & Visualizing

As an example, in the screenshots below, you see a target location designated by a magenta circle and a robot distinguished by its brightly colored (pick a color of your choice!) costume, with the usual box/circle tracking it. The robot starts at one side of the image and needs to reach the target, within 20 pixels.





This week's lab is more open-ended than the previous three weeks' - you have all of the pieces needed to build a successful point-to-point navigation system, but creating it is up to you. The challenge is to put it together step-by-step, so that you know each facet works before moving on.

Take at least a minute to consider the problems:
- The system will not know which way the robot is heading
- The robot will need to turn to head in the right direction - the system won't know (exactly) how much

Take at least a few minutes to consider the possibilities:
- you could have the robot alternate turning / going straight
- OR you could have the robot go straight but "overlay" a turn on top of that motion to correct its heading ("arcade control")
- you could watch the robot's position over time to determine its current heading
- OR you could have TWO colors on your robot to determine the heading(this would mean having two sets of thresholds)

Regardless, you will likely need to have different states.
- Think about what states you'll need
- Fewer states is better
- It's possible to run without states (but tricky!) But you're 100% welcome to take that approach, too...

To start, it'd nice to be able to print on your image...
- the distance (in pixels) to the goal
- the current heading-angle of the robot (in degrees)
- the desired heading-angle of the robot (in degrees)

Here is some sample code. You can also draw a colored box behind the text to make it easier to read.

```
font = cv2.FONT_HERSHEY_SIMPLEX
cv2.putText(img, 'OpenCV', (10,10), font, 2, (255,255,255), 1,
cv2.LINE_AA)
```

Important math hints on computing the heading
- Regardless of how you get the data, you will compute the heading from two points
- Use `math.atan2` to do this! Do not use plain atan
- Here's an example: `heading = math.atan2( y2-y1, x2-x1 )`
- It's not meaningful unless `math.hypot( y2-y1, x2-x1 ) > threshold` (some # of pixels)
    - That `math.hypot` is the built-in distance formula
    - If it's too small, simply don't update the heading!
- Remember that `math.degrees( rad )` converts rad to degrees. `math.radians( deg )` goes the other way.

Visualize your desired/actual heading with a line and circles, too...
- **Be sure your heading is being shown by a line segment!**
- if you'd like, add circles of different colors to show the start and finish endpoints of your line segment -- this makes it easier to interpret!
- It works best when you do this for BOTH the desired heading and the current heading!

Driving

These are optional steps for building up to all of the navigation possibilities. You can disregard and do it any way you like if this does not appeal.

Get to the center moving left to right
For the first few runs, start your robot to the left of the center, but not directly facing the center. You might build FSM states in order to

- move straight forward for a small period of time (1-2 seconds, maybe)
- remembering the location of the robot before that small motion
- tracking the location of the robot at the end of that small motion
- computing the heading between them!

From there, you'll want to turn in order to adjust the robot's heading more towards the goal point.

- There are many approaches! Consider comparing the heading value, above, with the desired heading value
- The desired heading can be computed in exactly the same way, but it uses different (visible/known) points...
- You might want to determine how fast the robot rotates in degrees per second when drive_direct(-50,50) is called
- But, all you need is to head roughly in the direction of the target, and then adjust - slightly - in order to ensure that the robot makes it there.

Get to the center from any direction
Next, generalize your finite-state machine so that your robot can make it to the center from any initial location around it.

Why is it harder going right-to-left? It's that atan2 has a discontinuity where the angle "wraps around"

- You might not worry about this - just let the robot do lots of turning
- You might have two cases and have the robot turn the smaller distance
- You might have the robot go backwards, if its heading is more than 90 degrees away from the desired heading

Get to any point you choose
This is your ultimate goal. For some good videos, mark an x on the floor with tape and then click, so the user knows where you clicked.

Submit

You will earn an A if you have completed this lab (including a quality writeup and appropriate video evidence).

A writeup (each lab will have a writeup like this):

- In the **Progress** section, you should have a brief introduction that restates the tasks of the week's lab, along with a record of any notes, data, brainstorming ideas, and progress (positive and negative!) you made. For example, here you should explain the algorithm -- or algorithms -- you used for line-following. Briefly, what was your reasoning behind them? If you changed approaches, explain what happened to motivate that change. The Progress section does not have to be huge: usually 3-4 paragraphs is a good size to aim for, for each lab
- In the **Results and Media** section, include at least one paragraphs on your results, including (1) how your robot did (2) any surprises or changes that emerged as you worked on it and (3) things your robot can't do that you hoped it would, and (4) any problems that arose -- and whether/how they were overcome. Be sure to include:
    - Optional: Still pictures of your robot -- maybe a selfie with your team

- At least two videos of your robot (10-20 seconds is best) - ideally, one with it running well and one where it's not, in order to contrast the two
- For each video, you should include a caption: 3-4 sentences summarizing each of the videos: what it will show, how well the robot does, and (briefly), why.
- You should include captions for each image, as well.
- Video should be submitted as links.
- **Include, too, at least one movie of your robot -- using a view of the monitor -- succeeding in a point-to-point navigation task.**
  - If it's using the center of the image, that's ok - start with left-to-right
  - If it can navigate from right-to-left, all the better -- show that off, too
  - If it can get to more than one point in a row, e.g., by clicking again, wonderful.
- **More generally, be sure your video(s) show off what your system can do!**
- **If possible, also include a movie of your robot using the view of the platform-and-floor, instead of the view of the monitor.**
  - For this one, be sure to place a piece of tape or other marker on the floor to show the spot where the user clicked.
  - That way, the viewer knows where the robot is heading!
- **Do include a diagram of your finite-state machine, if you used one - or an alternative picture of your organization, if you didn't!**

- Finally, you should have a short **Reflection** section on how the lab went overall, including parts that were particularly frustrating (if any), any realizations/insights (positive or negative), and how you would try to extend your robot's capabilities if you had additional time. Also welcome but not required here are suggestions for other tasks that fit the lab's theme or other variations that would be interesting. (From these latter thoughts, you might converge on an idea you're excited about for a final project.)
  - Also, in this section, you should include at least a note on anything extra or different your team might have tried -- either something suggested in the lab or something you designed yourself. (I don't want to miss or forget these!)

Upload your code.