

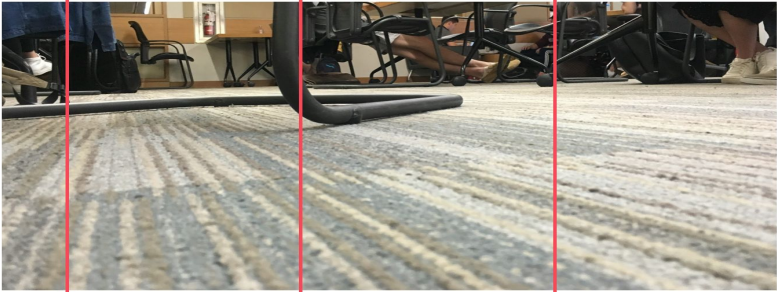


1. Proposal Title and Group Members
  - a. **An iRobot Create that moves around, mapping the chair legs, and avoiding collision as well.**
  - b. *Group 3*
    - i. Yating Zhan (yz1339)
    - ii. Alexandra Serralta (avs327)
    - iii. Ricardo Spinola (rs3826)
2. Background
  - a. In previous homeworks, our group made a line detection method that can detect lines in an image. We want to extend that functionality and apply it by solving problems like detecting chair legs in a classroom setting. This project is based on the line-detection method and dynamic map building problem we learnt from class. We will also incorporate resources such as the iRobot Create and a webcam.
3. Problem Identification
  - a. This project involves the line detection and depth calculation problem in Computer Vision and the SLAM path planning problem in Robotics.
4. Project Aim
  - a. Program the robot to analyze its surroundings (chair legs) in order to avoid collision.
  - b. Gradually build a map of containing the position data of chair legs in a room
5. Research Questions
  - a. How can one calculate depth of objects in an image?
  - b. How can one calculate an object's distance from source using its position in an image?
  - c. How can one extract data from an image and verify this information to be correct/accurate?
6. Significance of Questions
  - a. In order to create a two-dimensional map, we will need to calculate the object depth of the many obstacles in the classroom in order to be able to find their separation relative to each other. Without this information, a reliable map is unobtainable.
  - b. It is necessary to more or less have an idea of the pivot of the camera, and the distance from the robot to the obstacles in the picture, in order to further create a reliable map.
  - c. We need to consider how we can use Computer Vision tools such as convolutions and filters in order to make more readily visible the vertical items in an image and the plane they stand on.
7. Literature Review
  - a. Slam For Dummies
  - b. Tutorials about iRobot Create
8. Theoretical Framework
  - a. Our setup will consist of

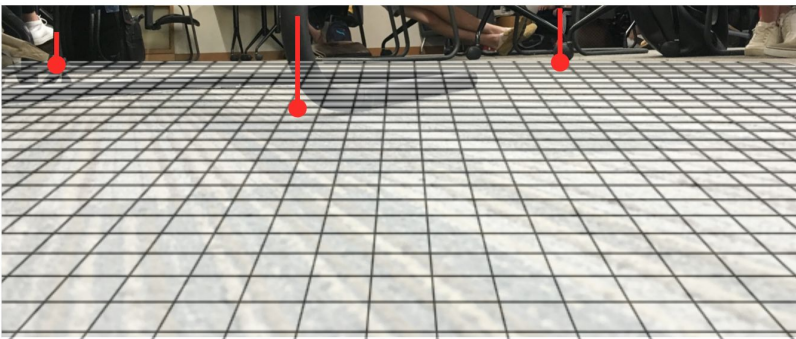
- i. A webcam camera module
  - ii. One iRobot Create
  - iii. One computer tethered to both the camera and the iRobot Create, to do all the processing.
- b. The robot will move around obstacles by analyzing its surroundings using the camera mounted on the robot. In theory, the way the computer will process the images that are taken from the robot will go as follows:
- c.

<p>SAMPLE IMAGE</p>  <p>irrelevant, noise</p> <p>“Expected” horizon line</p>	<p>STEP 1</p> <ul style="list-style-type: none"> <li>- Take a snapshot (this is a sample, not from the class and not using the chairs that will be used in the project). Most of the upper half of the image is not going to be needed or analyzed.</li> </ul>
<p>SAMPLE IMAGE</p>  <p>“Expected” horizon line</p>	<p>STEP 2</p> <ul style="list-style-type: none"> <li>- Since the camera is going to be at a predetermined angle, we can expect certain elements of it to remain constant, in this case the horizon line.</li> </ul>
	<p>STEP 3</p> <ul style="list-style-type: none"> <li>- From this, analyze the image and look for vertical and almost vertical lines.</li> </ul>



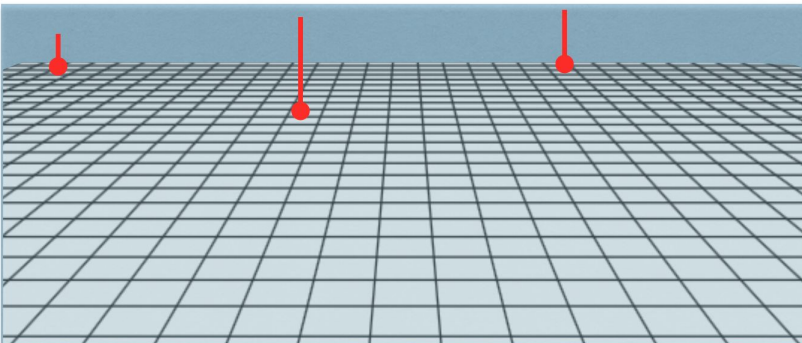
#### STEP 4

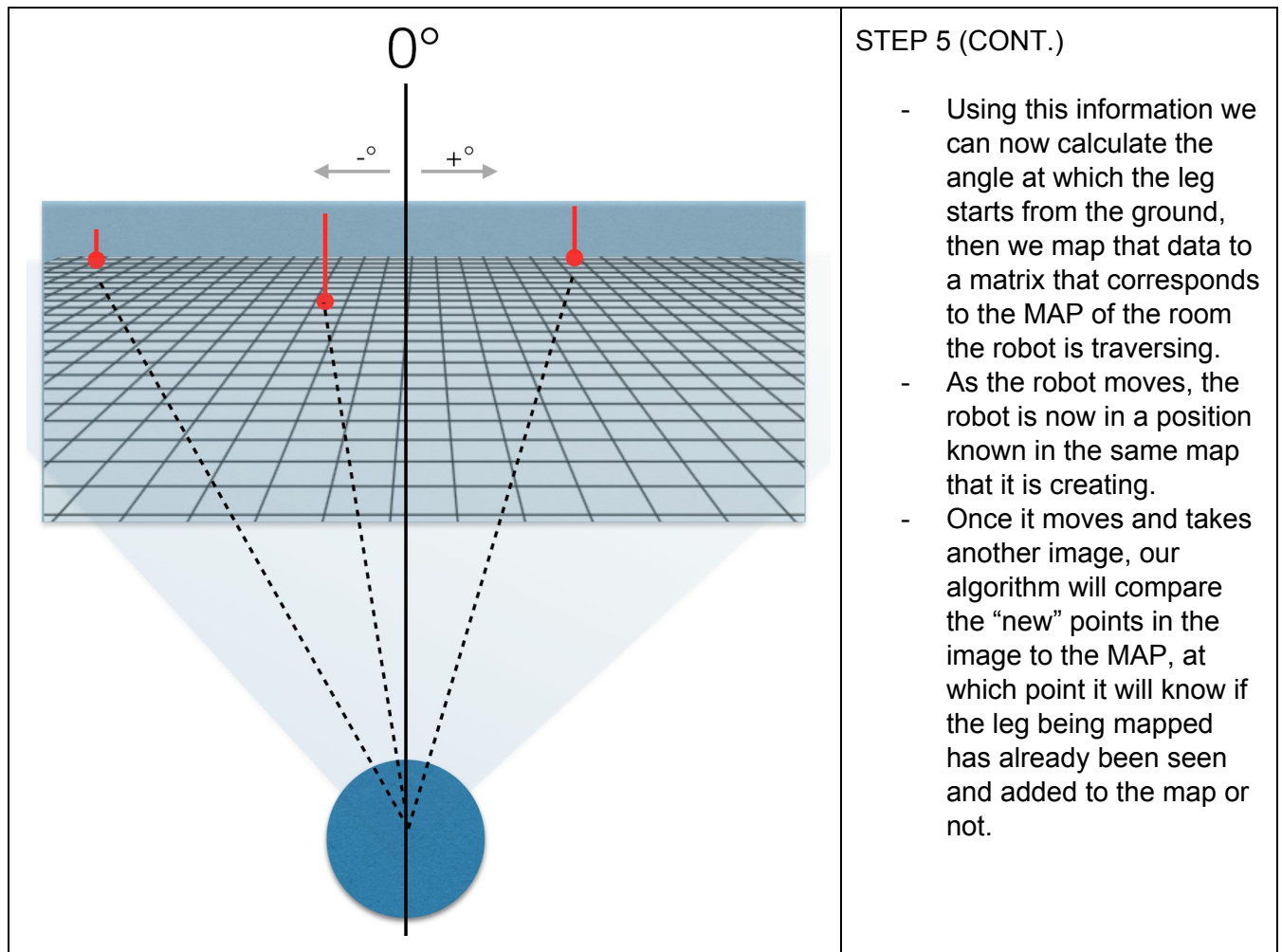
- Determine where these lines start and end (below the expected horizon)
- Gather information about these line starts.



#### STEP 5

- Transfer the information to another matrix, one that maps the images' approximate horizon to a theoretical matrix (created before) in which each location (where a leg meets the floor) contains the distance of that point to where the camera is (or the center of the robot).





## 9. Research Methods

- Build Methodology: we will twist and improve our methods as we experimenting our functions

## 10. Resources

- OpenCV
  - cv2.HoughLinesP: This method is used to detect line segment with a very useful parameter: **minLineLength** – Minimum line length. Line segments shorter than that are rejected. We can manipulate this parameter so that we will only use the nearest pole.
  - cv2.Canny: this method helps us to find the chair legs
- Python built-in functions
- API Tutorials
- Previous Homeworks
- Literature Introduced In Class

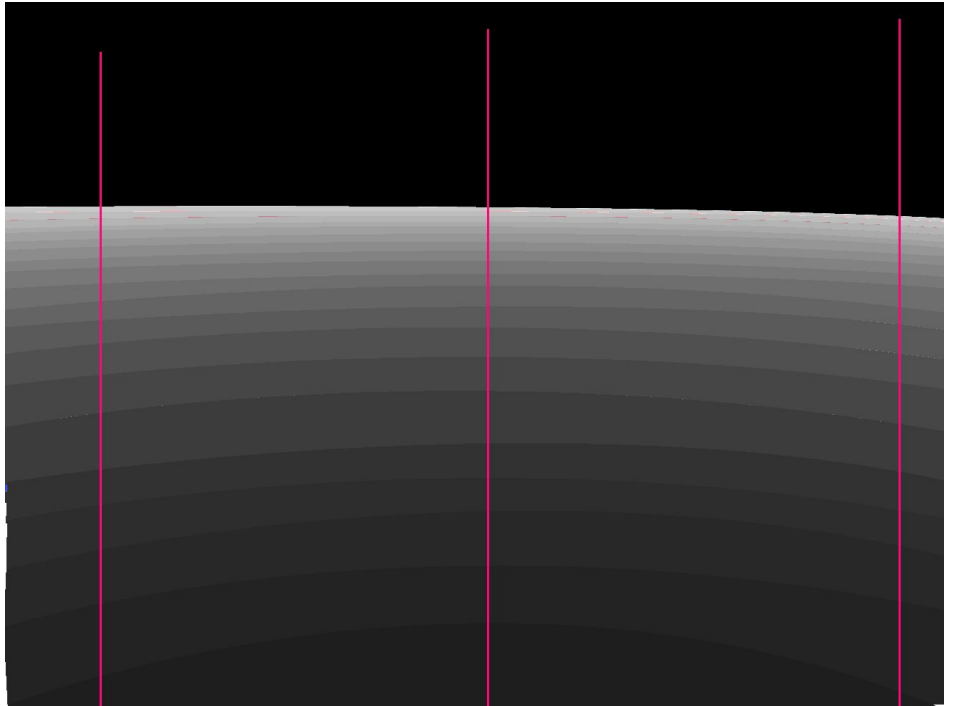
## 11. Timeline

- Week 1 (April 18th - April 25th):

- i. Met with Dr. Cicconet and Researched iRobot's API and projects:
  - 1. Question1: How to control the robot (code from web)
  - 2. Question2: How to get its configuration space (calculated by velocity and time the robot run)
  - 3. Question3: What are the resources we can use (Canny edge detect)
  - 4. Modification from original proposal:
    - a. After seeing Dr. Cicconet, we decide to use a webcam instead of Raspberry Pi for the following reasons:
      - i. Raspberry Pi has a lot of extra functionality that we don't use
      - ii. It's not convenient to program in Raspberry Pi since we only have one and need to set up time to program on it
      - iii. Alternatively, we bought a webcam and connect it to our laptop, sending pictures back to our laptop and use the program in our laptop to control the Robot.
- ii. Created the ground truth depth matrix:
  - 1. We Use a piece of code we found online to control the webcam to take a picture and return an cv object.  
(<http://stackoverflow.com/questions/11094481/capturing-a-single-image-from-my-webcam-in-java-or-python>) We decide that our demension of the matrix will be from 30 cm until 200 cm. Angle range is -20 to 20 degree.
  - 2. We measured the depth from the camera and the angle.



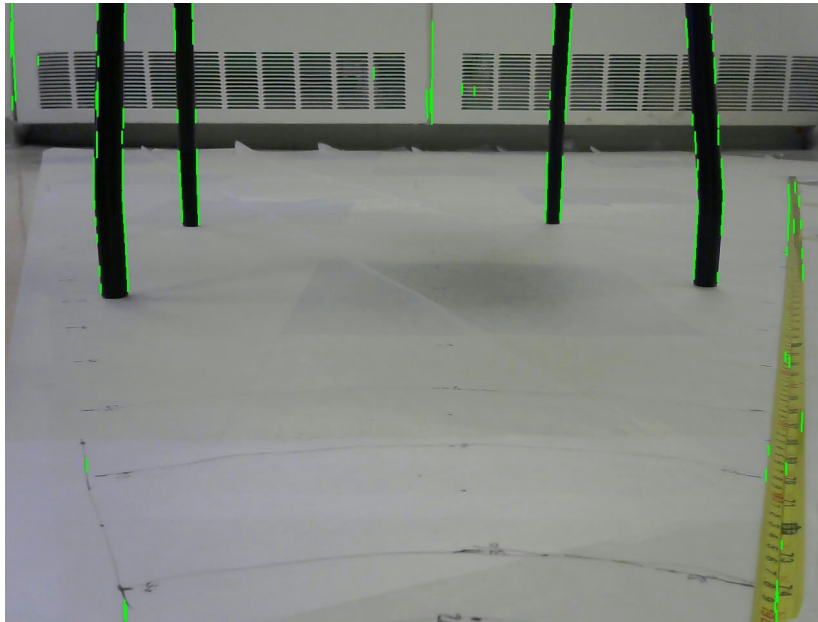
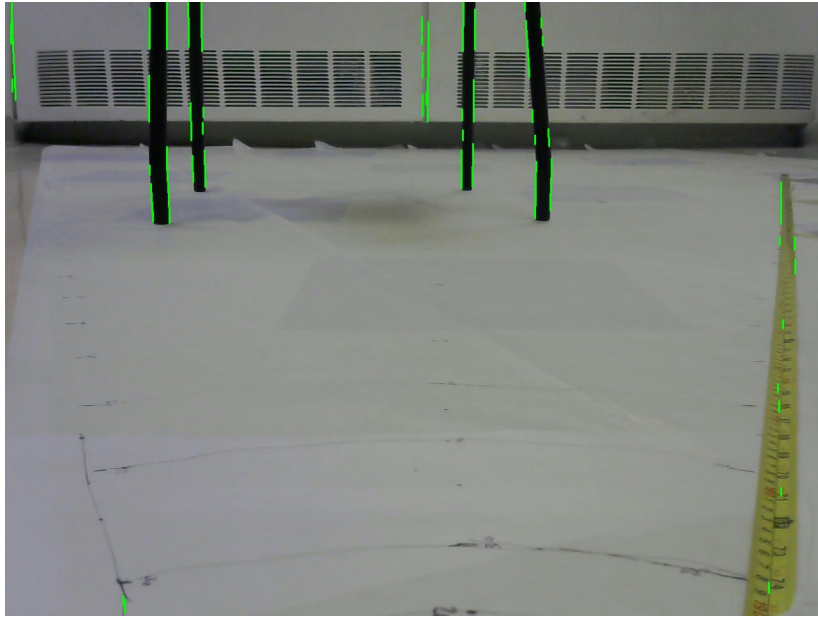
3. Created a depth map:



When we find a chair leg with the pixel pair  $(x, y)$  we will compare it with this map and produce (distance, angle) values. Based on this information, we can calculate the separation from the chair to the robot and the separation between two chairs.

4. Wrote a method to detect chair legs (black poles):
  - a. Name: findLegs(image)
  - b. Input: static image may or may not contain poles
  - c. Output: array of pixel pair of the chair legs in the image
  - d. Problem we are facing: The output from Canny Edge detection is really nice but when we try to find lines to get the ending of the chair legs, it's not giving us nice lines especially on the closer chairs.





5. Wrote a method to convert the pixel pair of the ending of the chair legs to the distance and angle respective to the robot according to the ground-truth depth matrix:
  - a. Name: `convert(x,y)`
  - b. Input: the ending of the chair leg in the image
  - c. Output: (distance, angle)
6. Wrote another method to calculate distance between two chair legs:
  - a. Name: `separation(x1, y1, x2, y2)`
  - b. Input: the ending of the chair legs in the image
  - c. Output: separation between the two poles

- b. Week 2 (April 25th - May 2nd):
  - i. Write the method to calculate the separations between all the poles
    - 1. Name: getSeparations(SepToRobot)
    - 2. Input: An array contains all the pole's relative position to the robot (separation and angle)
    - 3. Output: A two dimensional array contains the separation between each pair of poles
  - ii. Write a method to create a path depends on the environment:
    - 1. Name: createPath(Environment)
    - 2. Input: vector of environment contains separations between poles and robot
    - 3. Output: Short path (  $\leq 2$  feet)
  - iii. Write a method to control the robot's movement:
    - 1. Name: executePath(path)
    - 2. Input: A short path
    - 3. Output: new configuration space
- c. Week 3 (May 2nd - May 9th):
  - i. A main function to start the program:
    - 1. Name: main()
    - 2. Important variables:
      - a. robotConfs[] ---- an array of the space configurations of the robot
      - b. sepToRobot[] ---- an array of the separation between all the poles to the robot's current configuration space
    - 3. Pseudo:
 

```

robotConfs[0] = the configuration space where the robot starts
(hardcoded in for now).
while(true)
    image = takePicture()
    sepToRobot = findLegs(image)
    sepBetPoles = getSeparations(sepToRobot)
    path = createPath(sepToRobot, sepBetPoles)
    newConf = executePath(path)
    robotConfs.add(newConf)
              
```
  - ii. Clean things up, write final report, and prepare for demo and presentation

## 12. References

iRobot Create projects:

<http://www.irobot.com/About-iRobot/STEM/Create-2/Projects.aspx>

Slam For Dummies:

[http://ocw.mit.edu/courses/aeronautics-and-astronautics/16-412j-cognitive-robotics-spring-2005/projects/1aslambblas\\_repo.pdf](http://ocw.mit.edu/courses/aeronautics-and-astronautics/16-412j-cognitive-robotics-spring-2005/projects/1aslambblas_repo.pdf)

A piece of code to take a picture using webcam:



<http://stackoverflow.com/questions/11094481/capturing-a-single-image-from-my-webcam-in-java-or-python>

## Week 1 Diary

1. Discussion with Dr. Marcelo:
  - a.
2. Modification from original proposal:
  - a. After seeing Dr. Marcelo, we decide to use a webcam instead of Raspberry Pi for the following reasons:
    - i. Raspberry Pi has a lot of extra functionality that we don't use
    - ii. It's not convenient to program in Raspberry Pi since we only have one and need to set up time to program on it
    - iii. Alternatively, we bought a webcam and connect it to our laptop, sending pictures back to our laptop and use the program in our laptop to control the Robot.
3. Experiment with the base depth matrix:
  - a. We Use a piece of code we found online to control the webcam to take a picture and return an cv object.  
(<http://stackoverflow.com/questions/11094481/capturing-a-single-image-from-my-webcam-in-java-or-python>) We decide that our demension of the matrix will be from 30 cm until 200 cm. Angle range is -20 to 20 degree.



b.



4. Assignment until next meeting:
  - a. Everyone: put any resources and references here, write what you did.
  - b. Ricardo:
    - i. Build a matrix image
  - c. Alex:
    - i. A method to detect chair leg and where the leg ends. Example image: chairLeg0.jpg (left side closest leg is 80 cm away and right side closest leg is 85 cm away)
  - d. Yating:
    - i. Method of separation
    - ii. Learn code to control robot
    - iii. Process Ricardo's image to get real distance and angle
5. Next meeting time: Monday 2-6 pm
6. We decide to use Python