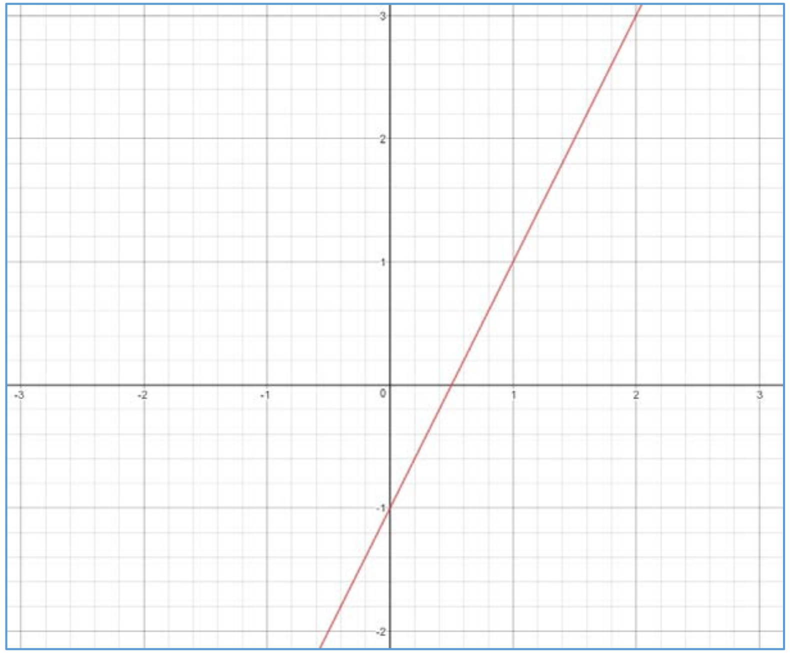
Hough Parameterization

Get Ready to Transform Reality!

Consider the Linear Equation

$$y = 2x - 1$$

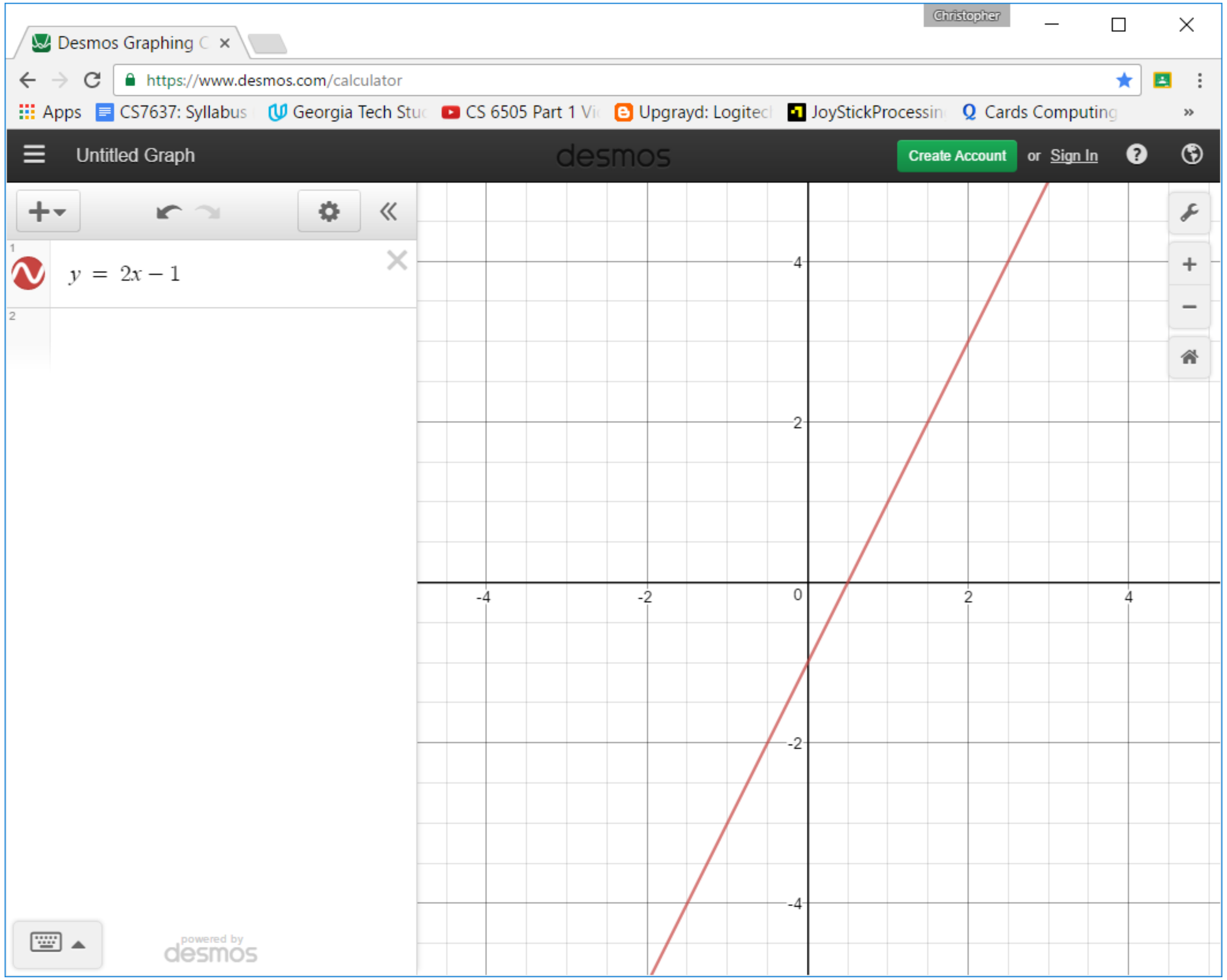
This equation's graph is:



Let us define the general form of this equation to be:

$$y = ax + b$$

Cartesian Space: X and Y Axis



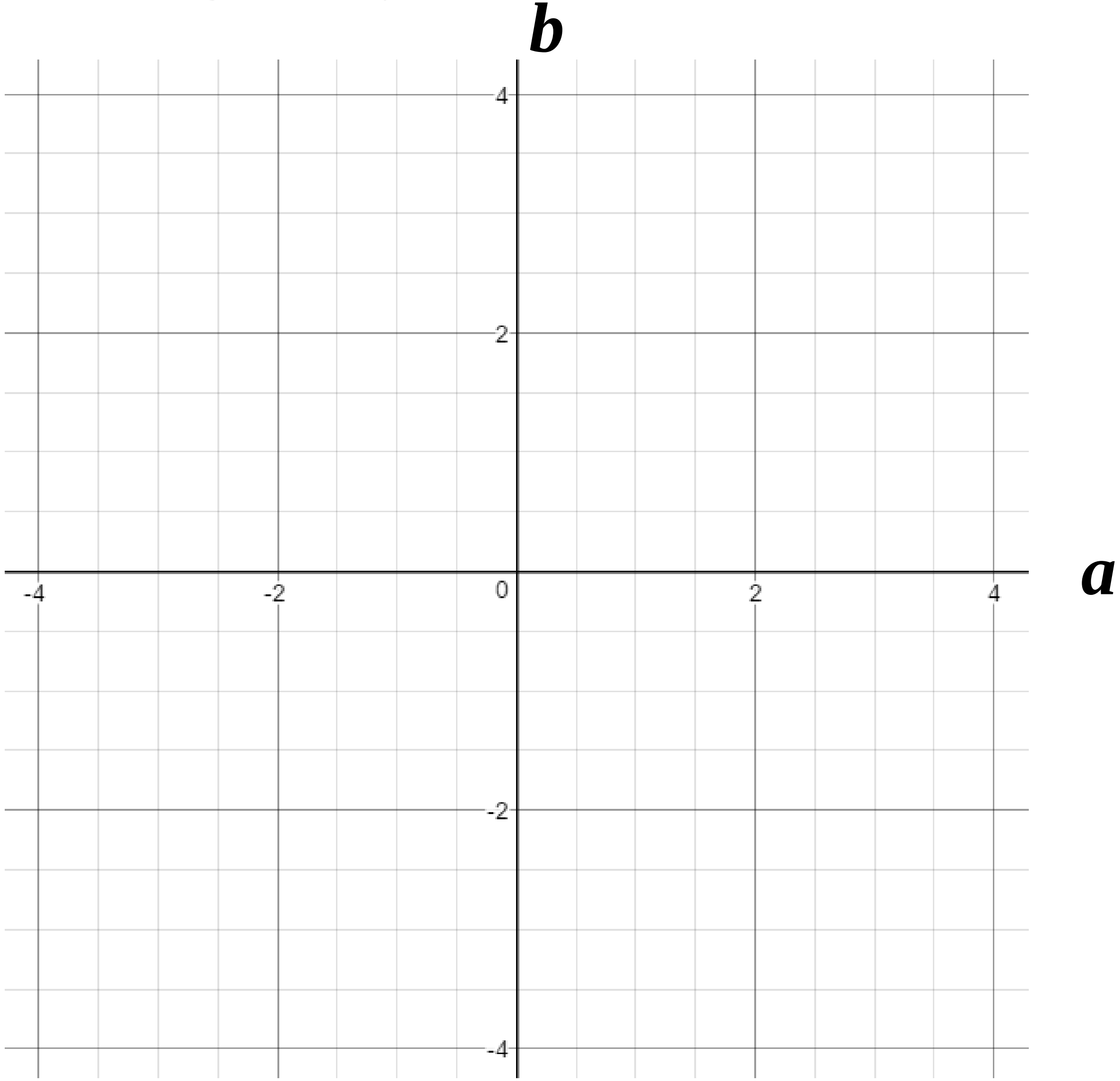
$y = 2x - 1$

$$y = ax + b$$

x and *y* are the variables

a and *b* are the constants

Hough Space: a and b axis



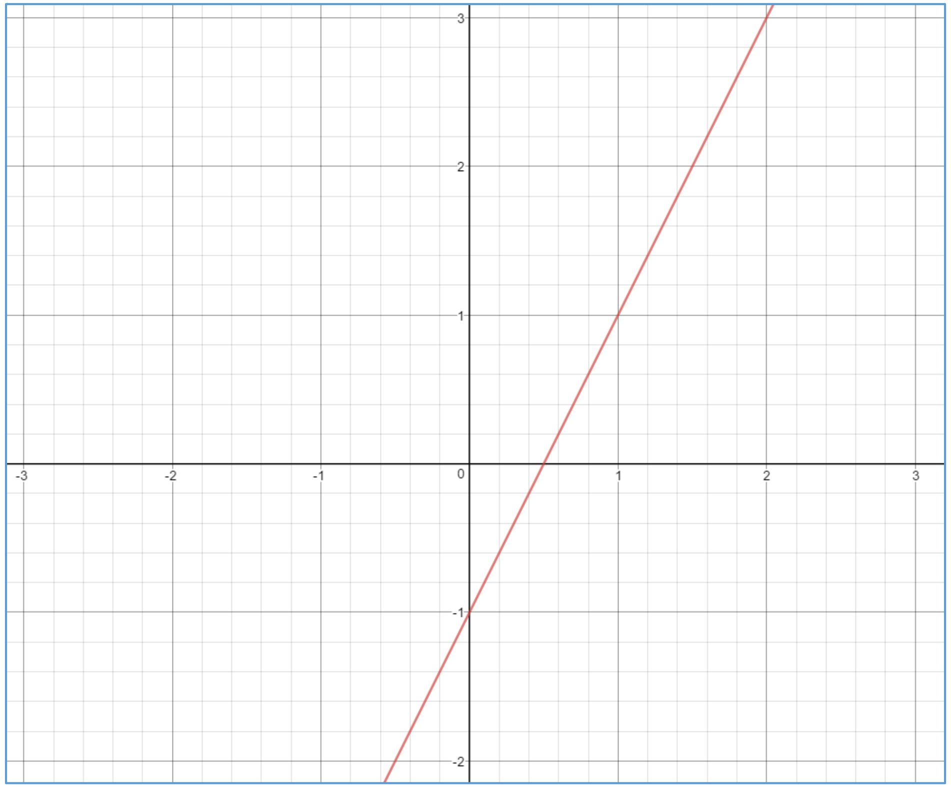
$$b = -xa + y$$

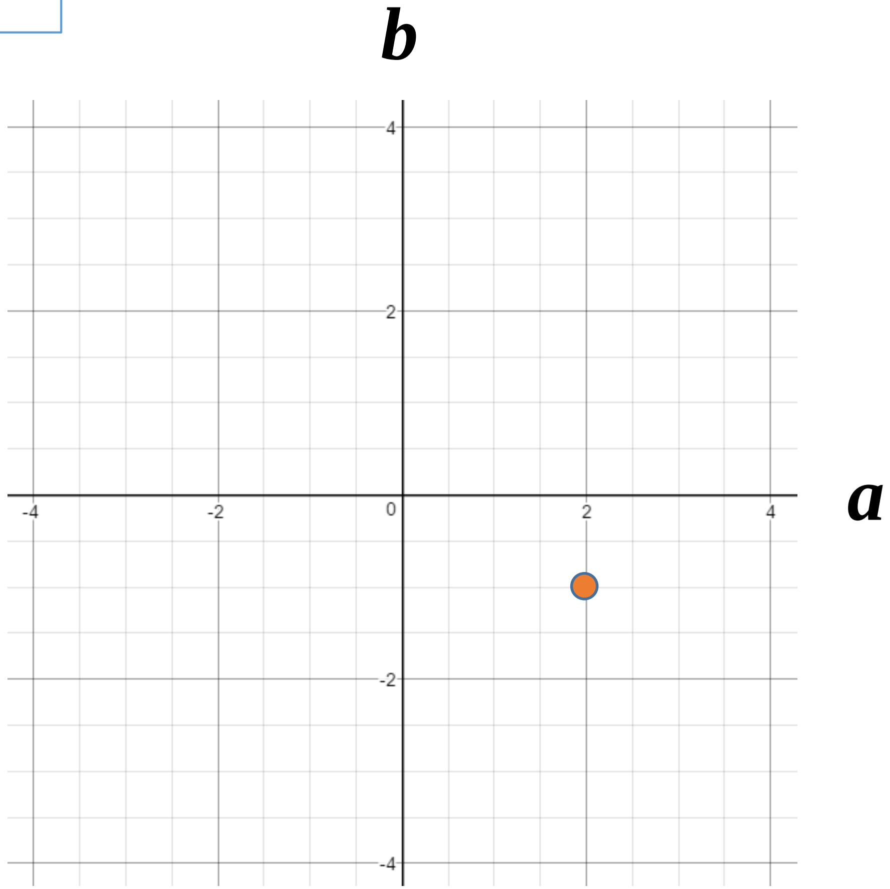
b and *a* are the variables

x and *y* are the constants

A *Line* in Cartesian Space is a *Point* in Hough Space

$$y = 2x - 1$$

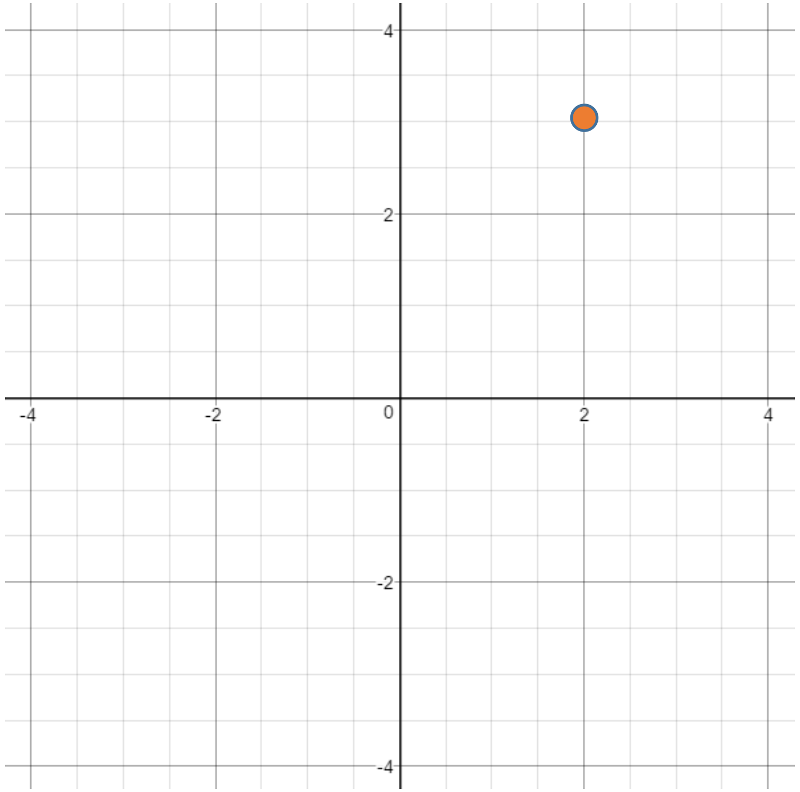


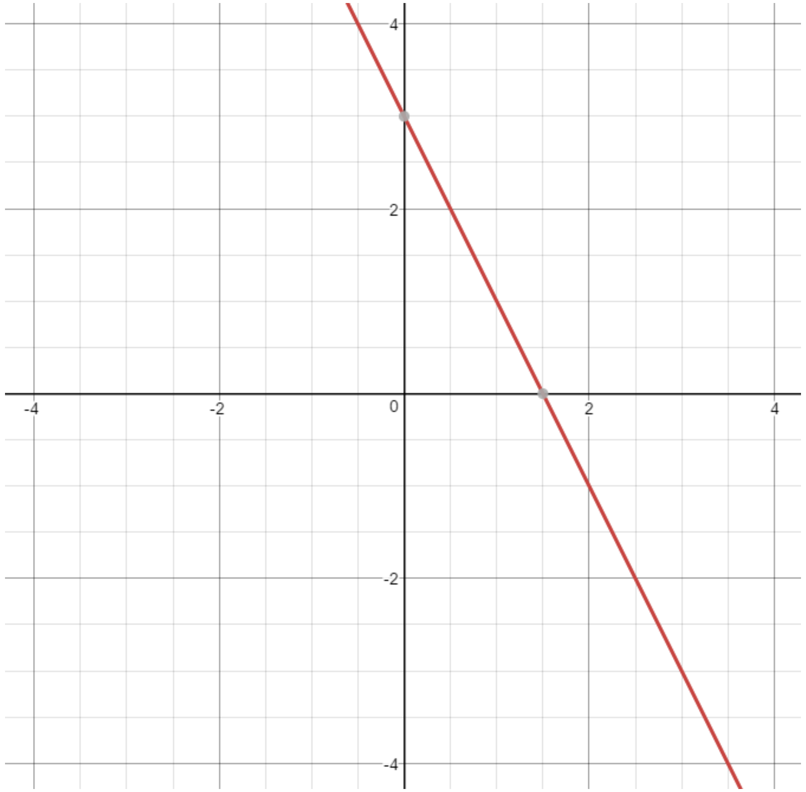


A *Point* in Cartesian space is a *Line* in Hough Space

$(2, 3)$

$b = -2a + 3$





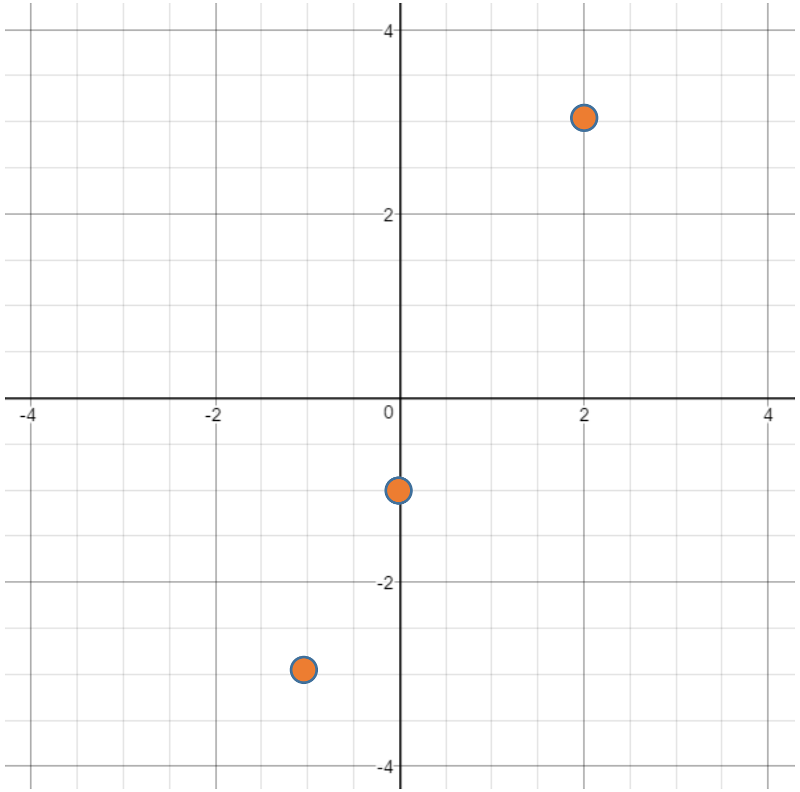
$b = -xa + y$

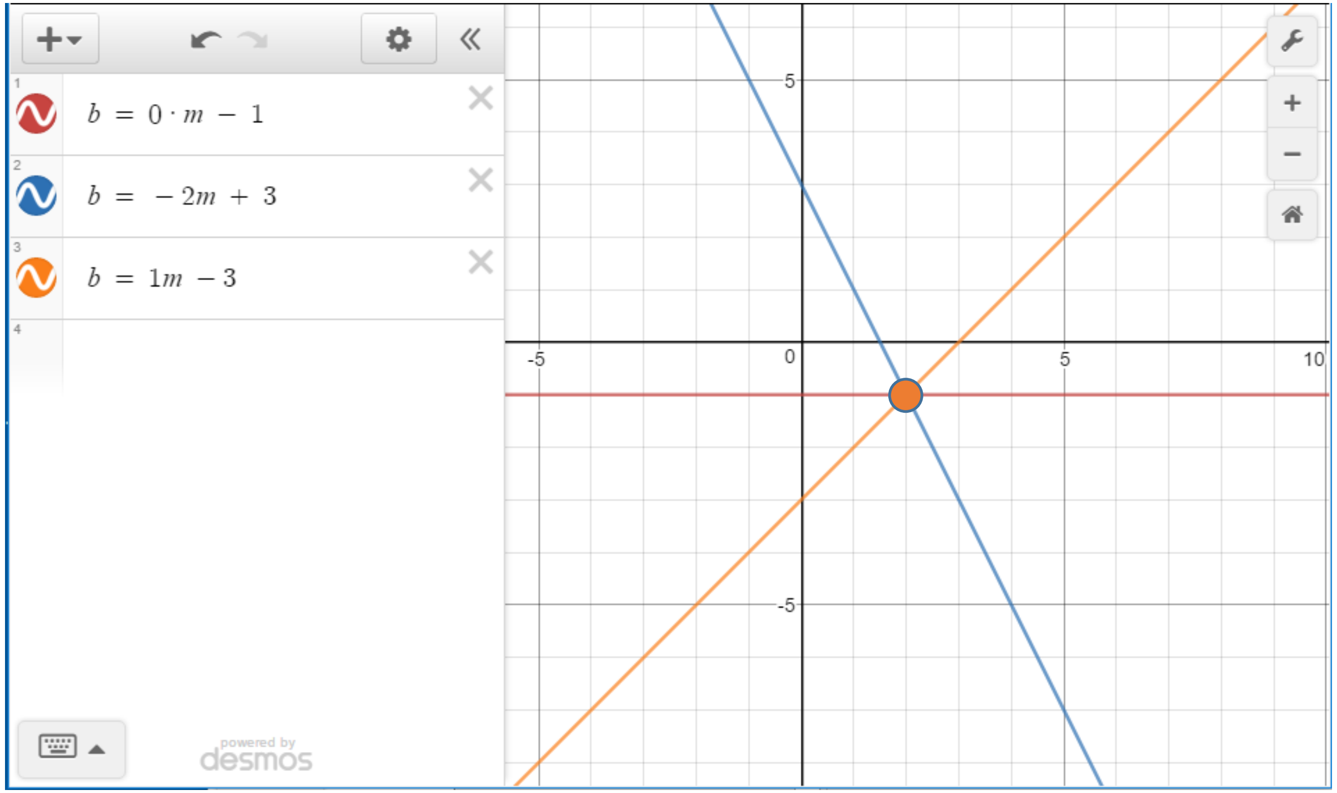
Why???

- We can take a set of points in Cartesian Space and Use Hough Space to “Vote” on the best line for the points in Cartesian Space

Cartesian Points:

(0, -1)
(2, 3)
(-1, -3)




$b = 0 \cdot m - 1$

$b = -2m + 3$

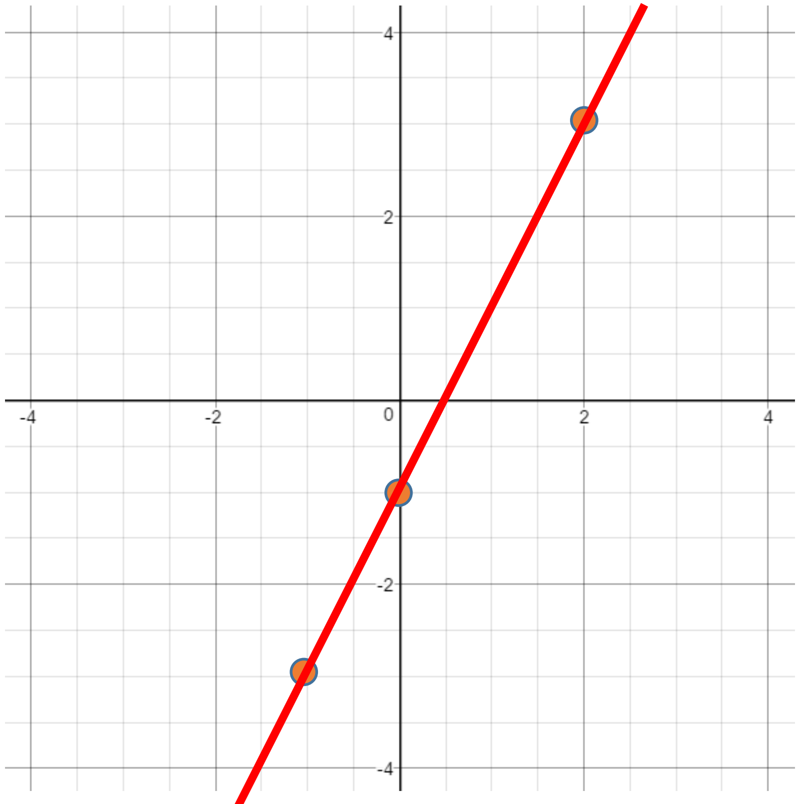
$b = 1m - 3$

Why???

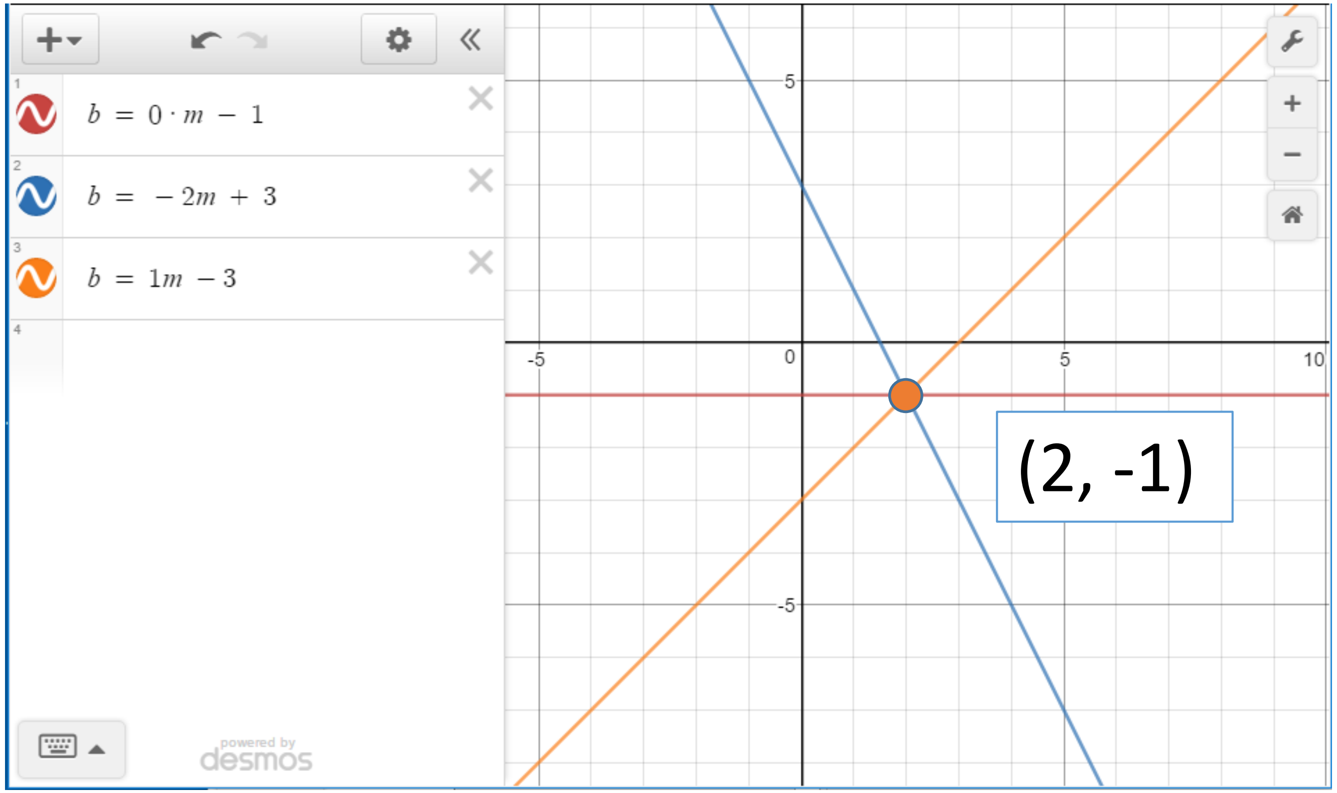
- We can take a set of points in Cartesian Space and Use Hough Space to “Vote” on the best line for the points in Cartesian Space

Cartesian Points:

(0, -1)
(2, 3)
(-1, -3)



$$y = 2x - 1$$


1 $b = 0 \cdot m - 1$

2 $b = -2m + 3$

3 $b = 1m - 3$

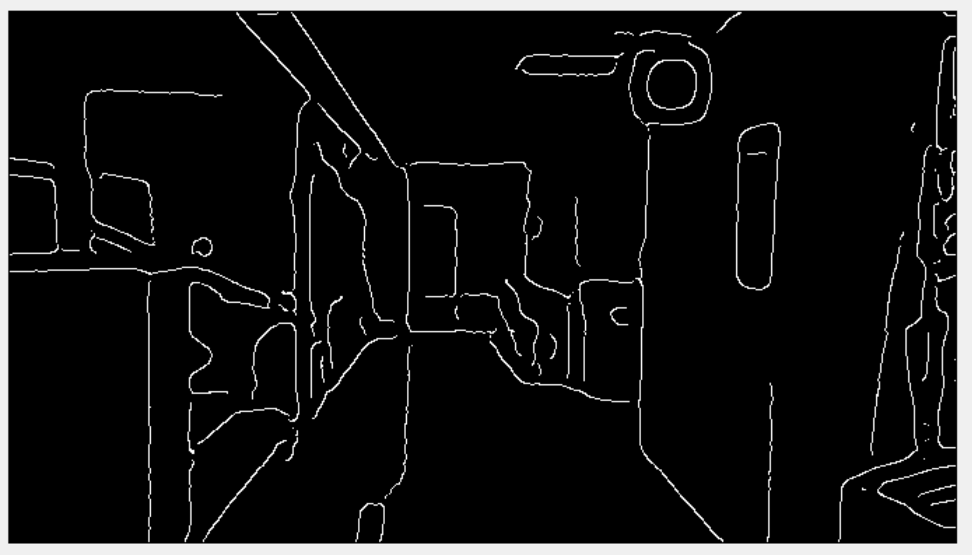
4

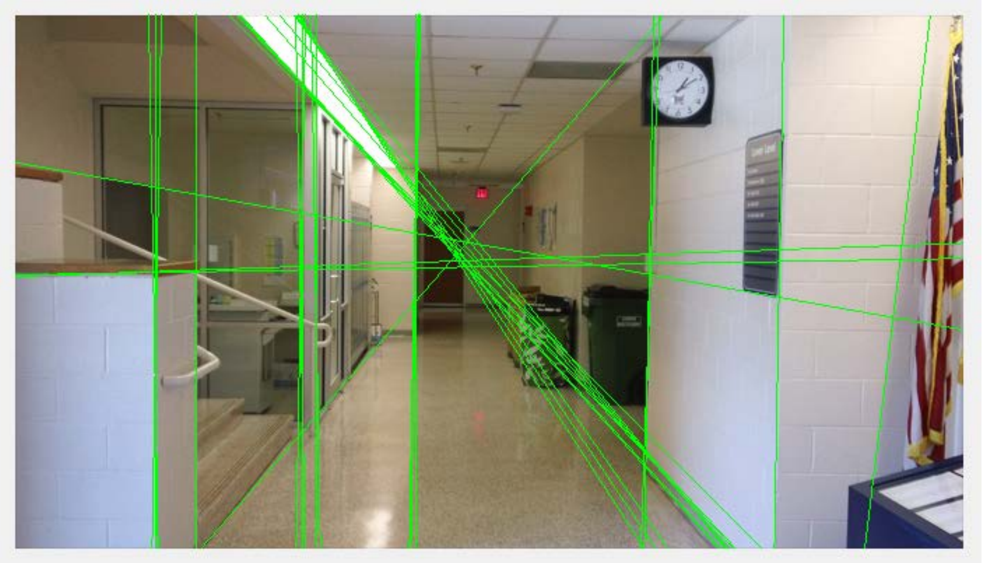
(2, -1)

a = 2, b = -1

Computer Vision: We can see Lines!

- Use the Points in an Edge Image to compute the Lines.
- The Computer then knows lines as objects with properties of angles and locations.
- Many applications to Robotics and Perception of self in Space.

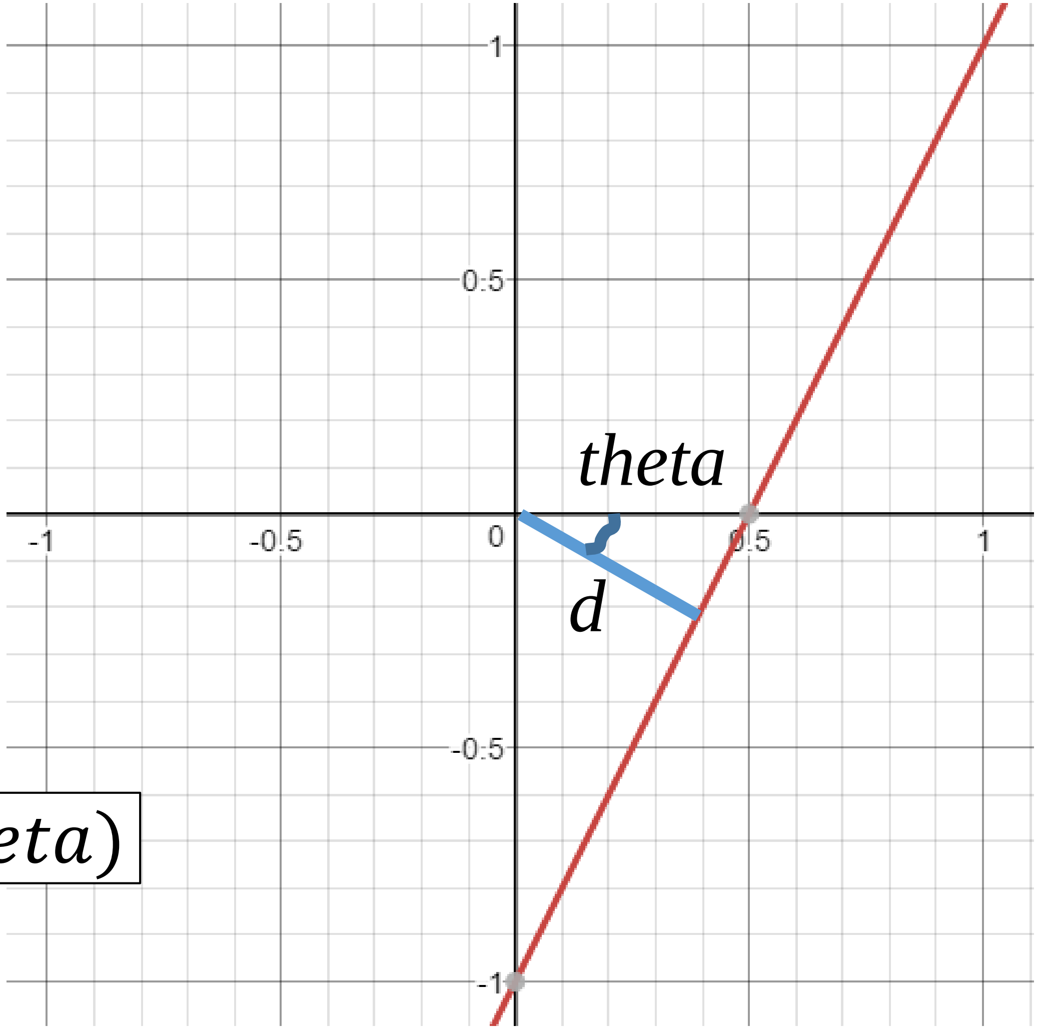




But . . . In Computer Vision we use trigonometric Hough Space

- Define Lines according to their vectors from the origin:
  - *d : Distance from Origin*
  - *theta : Angle from Origin*
- Use a trigonometric expression of x and y to compute the Hough Space.

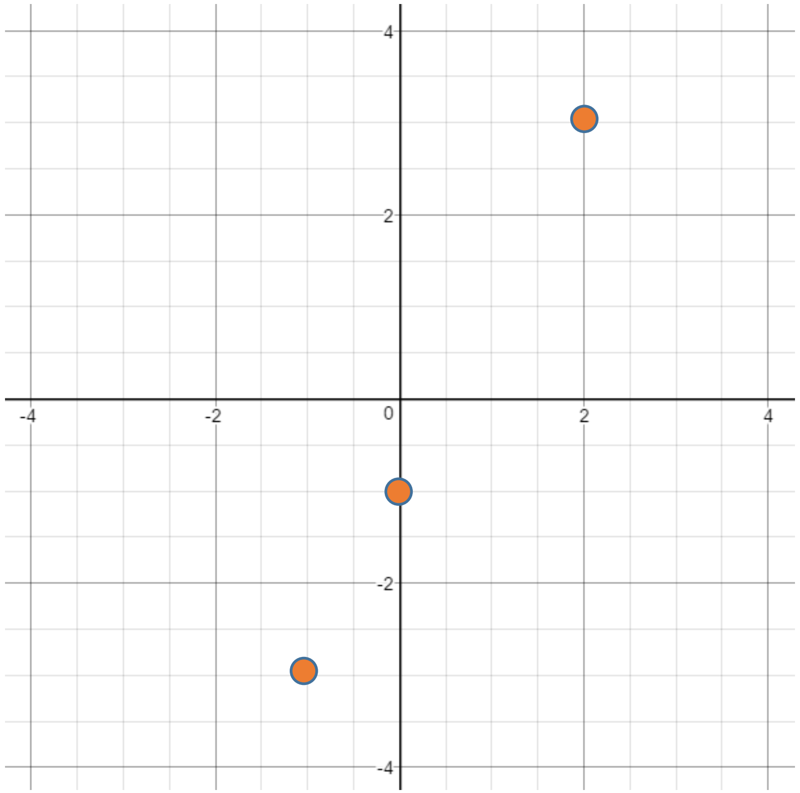
$$\boxed{d = x * \cos(theta) + y * \sin(theta)}$$

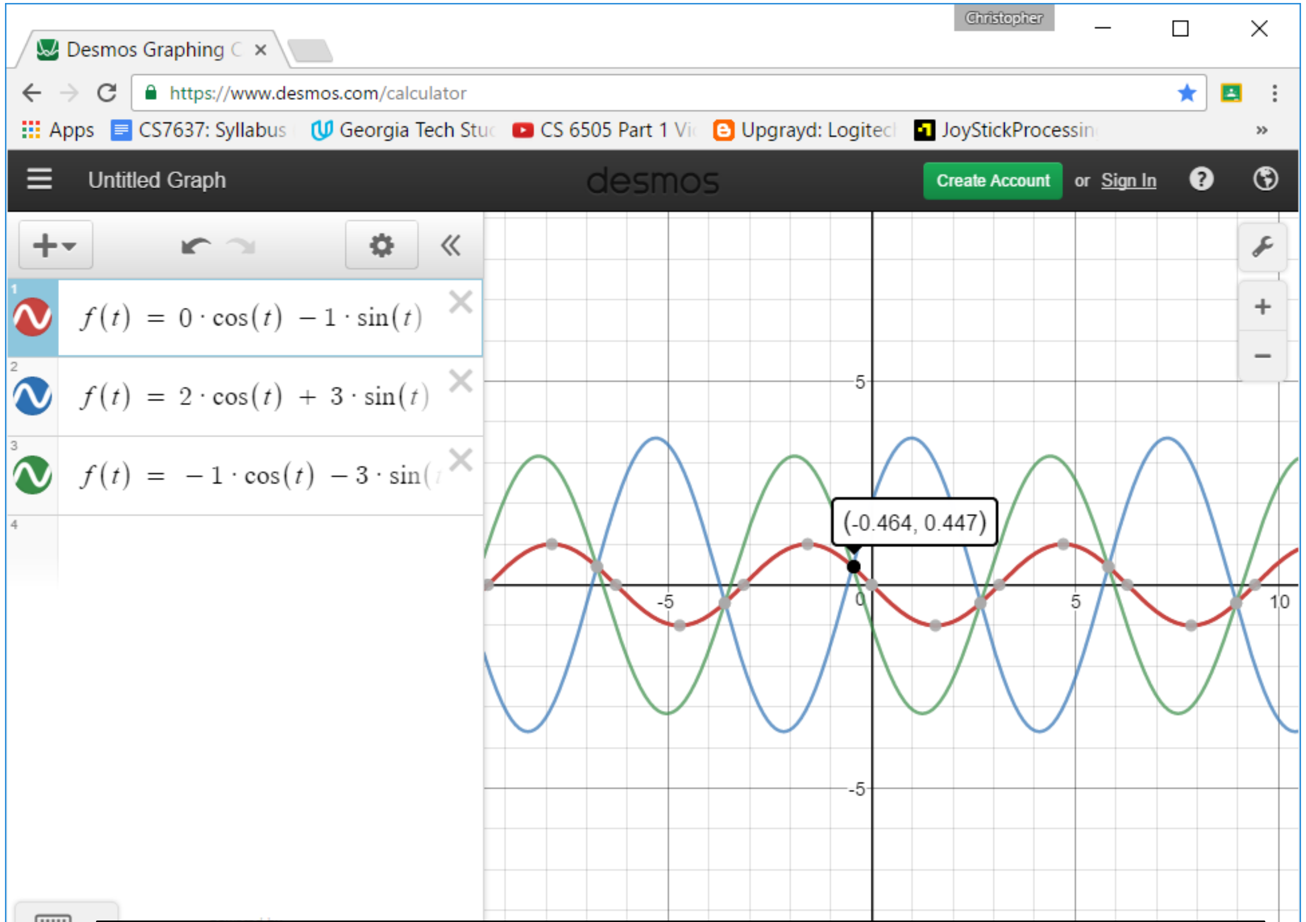


Example: Trigonometric Hough Space

Cartesian Points:

(0, -1)
(2, 3)
(-1, -3)





1 $f(t) = 0 \cdot \cos(t) - 1 \cdot \sin(t)$

2 $f(t) = 2 \cdot \cos(t) + 3 \cdot \sin(t)$

3 $f(t) = -1 \cdot \cos(t) - 3 \cdot \sin($

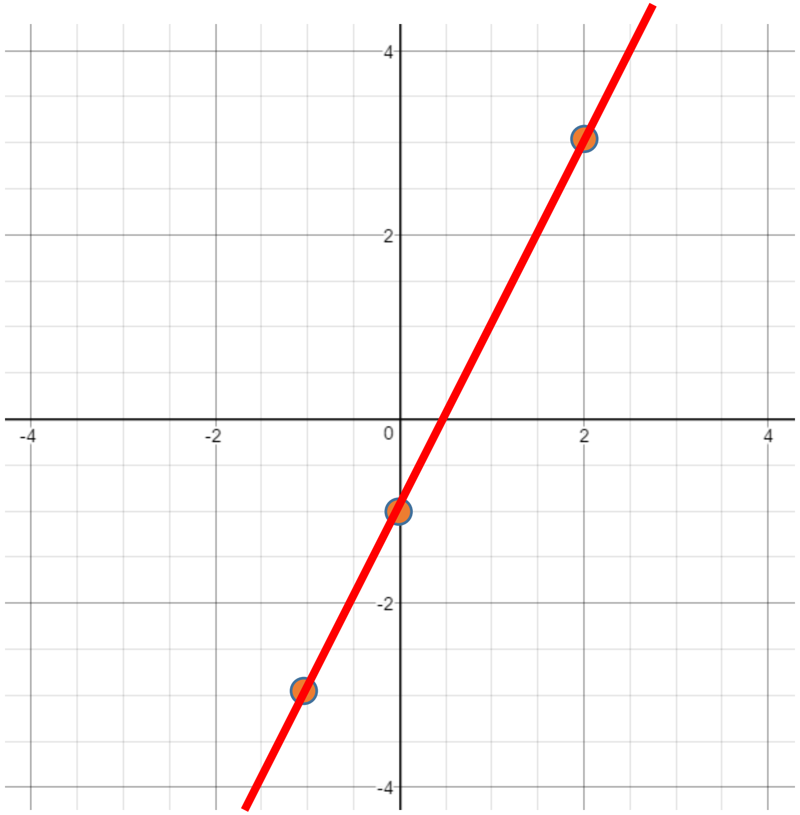
4

$$d = x * \cos(theta) + y * \sin(theta)$$

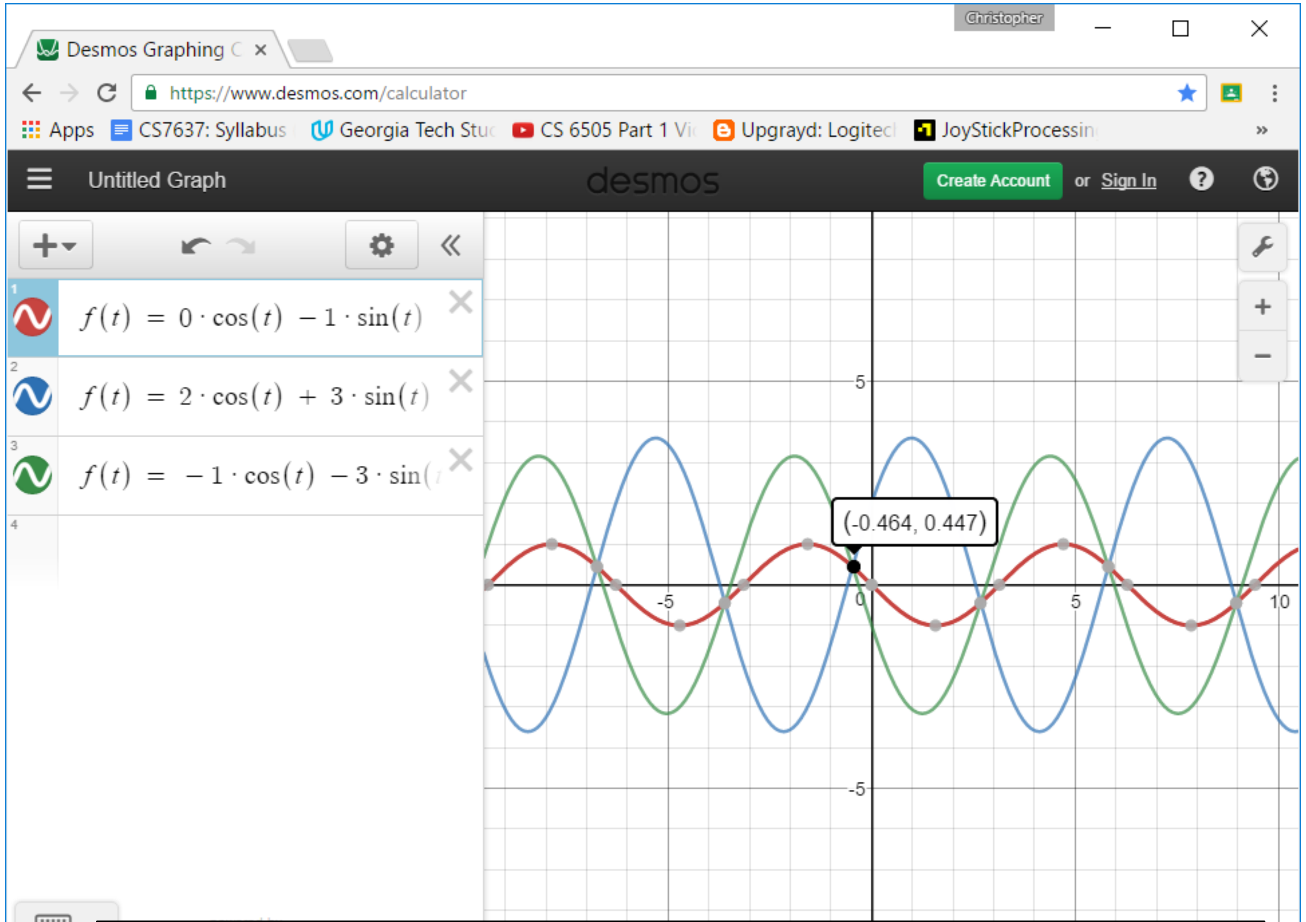
Example: Trigonometric Hough Space

Cartesian Points:

(0, -1)
(2, 3)
(-1, -3)



d = 0.447, theta = -0.464 Radians



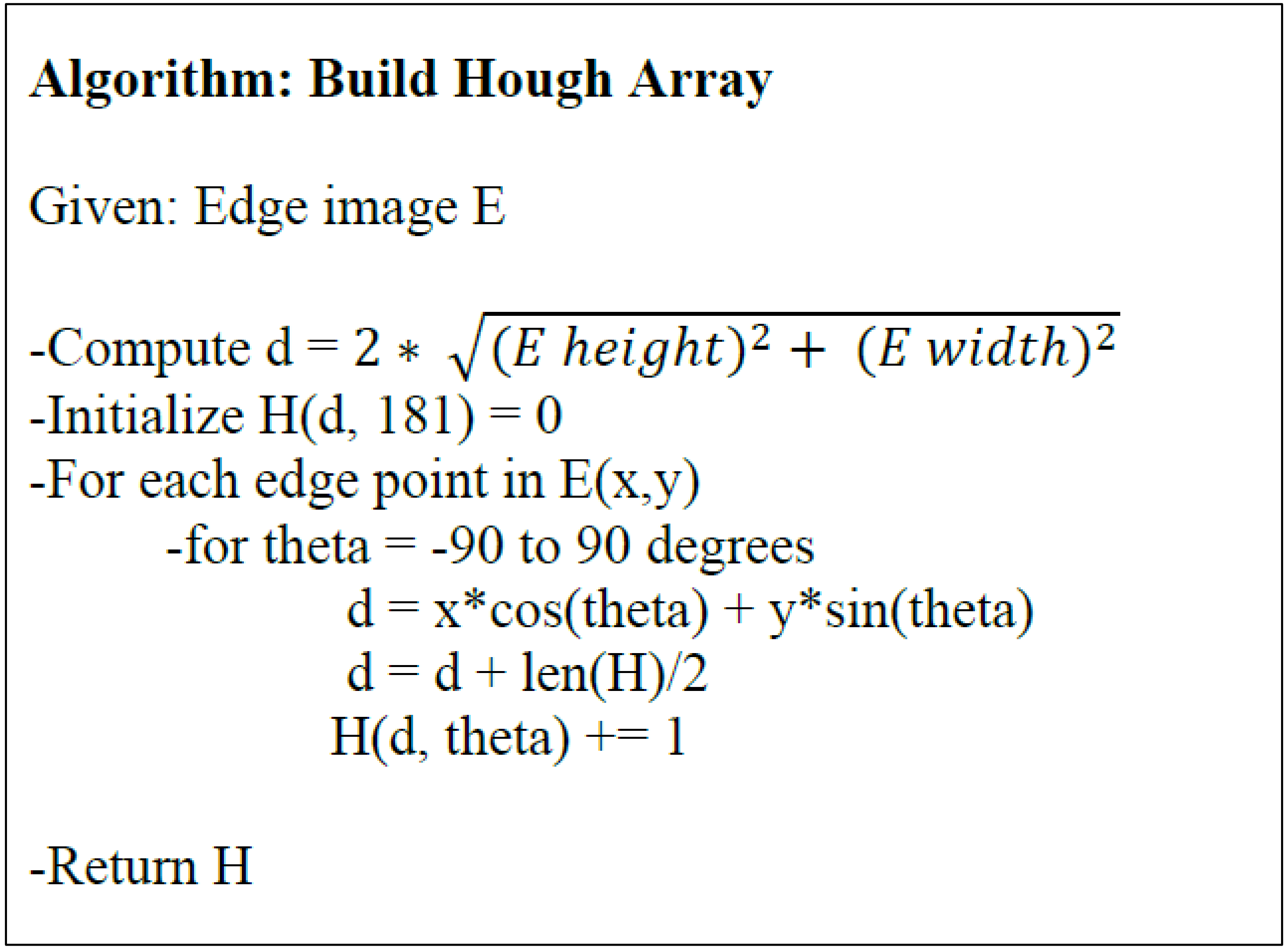
$f(t) = 0 \cdot \cos(t) - 1 \cdot \sin(t)$

$f(t) = 2 \cdot \cos(t) + 3 \cdot \sin(t)$

$f(t) = -1 \cdot \cos(t) - 3 \cdot \sin($

$$d = x * \cos(theta) + y * \sin(theta)$$

Algorithm for Voting in Hough Space (Creating a Hough Accumulator)



Algorithm: Build Hough Array

Given: Edge image E

```
-Compute d = 2 * $\sqrt{(E\ height)^2 + (E\ width)^2}$
-Initialize H(d, 181) = 0
-For each edge point in E(x,y)
          -for theta = -90 to 90 degrees
                    d = x*cos(theta) + y*sin(theta)
                    d = d + len(H)/2
                    H(d, theta) += 1

-Return H
```

Notes:

1. Edge Image E is an image consisting of only 1’s and 0’s.
2. The Array H is a numpy array with dimensions of d rows and 181 columns with data type of np.uint8
3. Calculations will be done as floats, but must be converted back to integers before voting in H array.
4. This algorithm ‘votes’ for the best polar locations of lines in Edge image E.