**Passos para identificar as linhas:**

* Inicialmente iremos usar a transformada de Hough para detectar as linhas de interesse

1 – Identificar as bordas da imagem

bordas = cv2.Canny(img, 50, 200)- importer imagem e as dimensões da mesma

plt.imshow(bordas, cmap="gray")

2- Aplicando transformada

linhas = cv2.HoughLines(bordas, 1, np.pi / 180, 150, **None**, 0, 0)

|  |
| --- |
| import math |
|  | import numpy as np, cv2 |
|  |  |
|  | width = 300 |
|  | height = 400 |
|  |  |
|  | M2 = 0 |
|  |  |
|  | #quatro pontos iniciais dos corners do grid |
|  | referencePoints = np.float32([[width/4,height/4],[3\*width/4,height/4],[3\*width/4,3\*height/4],[width/4,3\*height/4]]) |
|  | #referencePoints = np.float32([[150,150],[250,150],[250,250],[150,250]]) |
|  |  |
|  | currentPoint = -1 #indica qual dos 4 pontos atuais estah selecionado |
|  | calibrating = True #indica se o modo de calibracao estah ativado |
|  | fullScreen = False #indica se o modo de tela cheia estah ativado |
|  |  |
|  | # inputimage1 = cv2.imread("grid.png") # le a imagem do grid |
|  | inputimage1 = cv2.imread("fluxometro.jpg") # le a imagem do grid |
|  | inputimage1 = cv2.resize(inputimage1, (width,height)) |
|  | scale = cv2.imread("scale2.png") # le a imagem da escala de cor |
|  | rows1, cols1 = inputimage1.shape[:2] # le as dimensoes do grid |
|  | pts1 = np.float32([[0,0],[cols1,0],[cols1,rows1],[0,rows1]]) #cria os pontos de controle para a imagem do grid |
|  | image = np.zeros((height, width, 3), np.uint8) #cria uma imagem colorida para a tela |
|  |  |
|  |  |
|  | #funcao que retorna a cor de um corner especifico para a calibracao |
|  | def pointColor(n): |
|  | if n == 0: |
|  | return (0,0,255) |
|  | elif n == 1: |
|  | return (0,255,255) |
|  | elif n == 2: |
|  | return (255,255,0) |
|  | else: |
|  | return (0,255,0) |
|  |  |
|  | #esta funcao eh chamada sempre que um evento de mouse acontece (clicar, arrastar, soltar, etc) |
|  | def mouse(event, x, y, flags, param): |
|  | if(not calibrating): |
|  | return |
|  | global currentPoint |
|  |  |
|  | if event == cv2.EVENT\_LBUTTONDOWN: |
|  | cp = 0 |
|  | #descobre em qual dos 4 pontos o usuario clicou (precisa estar a uma distancia maxima de 4 pixels para ser considerado o clique) |
|  | for point in referencePoints: |
|  | dist = math.sqrt((x-point[0])\*(x-point[0])+(y-point[1])\*(y-point[1])) |
|  | if dist < 4: |
|  | currentPoint = cp |
|  | break |
|  | else: |
|  | cp = cp + 1 |
|  |  |
|  | if event == cv2.EVENT\_LBUTTONUP: #quando o clique é solto, diz o que o ponto selecionado eh -1 |
|  | currentPoint = -1 |
|  |  |
|  | if currentPoint != -1: #move as coordenadas do ponto selecionado para a posicao lida do mouse |
|  | referencePoints[currentPoint] = [x,y] |
|  |  |
|  | #cria a janela principal da aplicacao |
|  | cv2.namedWindow("test", cv2.WINDOW\_NORMAL) |
|  | #associa uma funcao de eventos do mouse a janela principal criada |
|  | cv2.setMouseCallback("test", mouse) |
|  |  |
|  | #diz que a cor selecionada inicialmente eh branco |
|  |  |
|  | #loop principal |
|  | while True: |
|  | image[:] = inputimage1 #limpa a imagem (pinta toda ela de preto) |
|  |  |
|  | if calibrating: #se estiver com o modo de calibracao ativado, pinta os pontos coloridos em cada corner do grid |
|  | color = 0 |
|  | for point in referencePoints: |
|  | cv2.circle(image, (int(point[0]), int(point[1])),5,pointColor(color), -1) |
|  | color = color + 1 |
|  |  |
|  | M = cv2.getPerspectiveTransform(pts1,referencePoints) # calcula a projecao com base nas coordenadas dos 4 corners |
|  | M2 = cv2.getPerspectiveTransform(referencePoints, pts1) # calcula a projecao com base nas coordenadas dos 4 corners   * Inicialmente iremos usar a transformada de Hough para detectar as linhas de interesse (aula 14)   1 – Identificar as bordas da imagem  bordas = cv2.Canny(img, 50, 200)- importar imagem e as dimensões da mesma  plt.imshow(bordas, cmap="gray")  2- Aplicando transformada  linhas = cv2.HoughLines(bordas, 1, np.pi / 180, 150, **None**, 0, 0)  3**- destacando as linhas que iremos manipular**  import math  bordas\_copia = cv2.cvtColor(bordas, cv2.COLOR\_GRAY2BGR)  **for** i **in** range(0, len(linhas)):  rho = linhas[i][0][0]  theta = linhas[i][0][1]  a = math.cos(theta)  b = math.sin(theta)  x0 = a \* rho  y0 = b \* rho  pt1 = (int(x0 + 1000\*(-b)), int(y0 + 1000\*(a)))  pt2 = (int(x0 - 1000\*(-b)), int(y0 - 1000\*(a)))  cv2.line(bordas\_copia, pt1, pt2, (0,0,255), 3, cv2.LINE\_AA) |
|  | **4 – fazendo varredura afim de identificar as coordenadas das linhas destacadas (escala do fluxometro)**  **USAR O CÓDIGO FEITO NO PRJETO 0, ONDE VARRIAMOS A IMAGEM NO EIXO X E NO Y AFIM DE DIRECIONAR O ROBÔ AO LONGO DA PISTA, DESTE MODO IREMOS IDENTIFICAR A LINHA QNDO FIZERMOS A LEITURA NO EIXO Y(VERTICAL) DA IMAGEM. Ao identificar a quantidade de linhas, iremos dividir pelas dimensão da imagem afim de saber os valores em coordenadas de cada linha, tal como o espaçamento entre elas**  **5 – Agora iremos importar a imagem afim de detectar o nível que se encontra a “bolinha” do fluxometro. Para isso usaremos TRANSFORMADA DE HOUGH para detecção de círculos (AULA15)**   * **Carregando a imagem**   img = cv2.imread('240\_pepsi\_12oz.png')  img = cv2.cvtColor(img, cv2.COLOR\_BGR2RGB)  img\_cinza = cv2.cvtColor(img, cv2.COLOR\_RGB2GRAY)   * **Criando a variável “circulo” que irá armazenar valores de suas cordenadas e do raio**   circulos = cv2.HoughCircles(img\_cinza, cv2.HOUGH\_GRADIENT, dp=1.2, minDist=100)   * **Verificando/identificando os circulos**   **if** circulos **is** **not** **None**:  circulos = np.round(circulos[0, :]).astype("int")  **for** circulo **in** circulos:  print('(**%d**, **%d**) com raio **%d**' % (circulo[0], circulo[1], circulo[2]))   * Após identificar o círculo iremos calcular o seu centro de massa, com o valor do centro de massa iremos identificar em que valor o mesmo se encontra na escala do fluxometro de acordo com o espaçamento entre cada linha( valor este que já saberemos de acordo com o item 4) o link abaixo exemplifica o código a ser utilizado afim de encontrar o centro de massa, em posse do valor de Y do centro de massa, poderemos prosseguir, como foi descrito anteriormente, afim de identificar em que nível(na escala do fluxometro) encontra-se a bolinha   <https://alexandremaciel.wordpress.com/2016/03/07/centro-de-massa-a-partir-de-uma-imagem/>   * Desenhar os círculos que forem detectados   **for** (x, y, r) **in** circulos:  cv2.circle(img, (x, y), r, (255, 0, 0), 4)  plt.imshow(img) |
|  | cv2.imshow("test", image) #exibe a imagem na tela |
|  | key = cv2.waitKey(1) & 0xFF #espera 1ms e verifica se alguma tecla foi pressionada |
|  |  |
|  | if key == ord("c"): #caso a tecla 'c' tenha sido pressionada, habilita ou desabilita o modo de calibracao |
|  | calibrating = not calibrating |
|  |  |
|  | if key == ord("f"): #caso a tecla 'f' tenha sido pressionada, habilita ou desabilita o modo de tela cheia |
|  | if fullScreen == False: |
|  | cv2.setWindowProperty("test", cv2.WND\_PROP\_FULLSCREEN, cv2.WINDOW\_FULLSCREEN) |
|  | else: |
|  | cv2.setWindowProperty("test", cv2.WND\_PROP\_FULLSCREEN, cv2.WINDOW\_NORMAL) |
|  | fullScreen = not fullScreen |
|  |  |
|  | if key == ord("q"): #caso a tecla 'q' tenha sido pressionada, fecha a aplicacao |
|  | break |
|  |  |
|  | cv2.destroyAllWindows() |