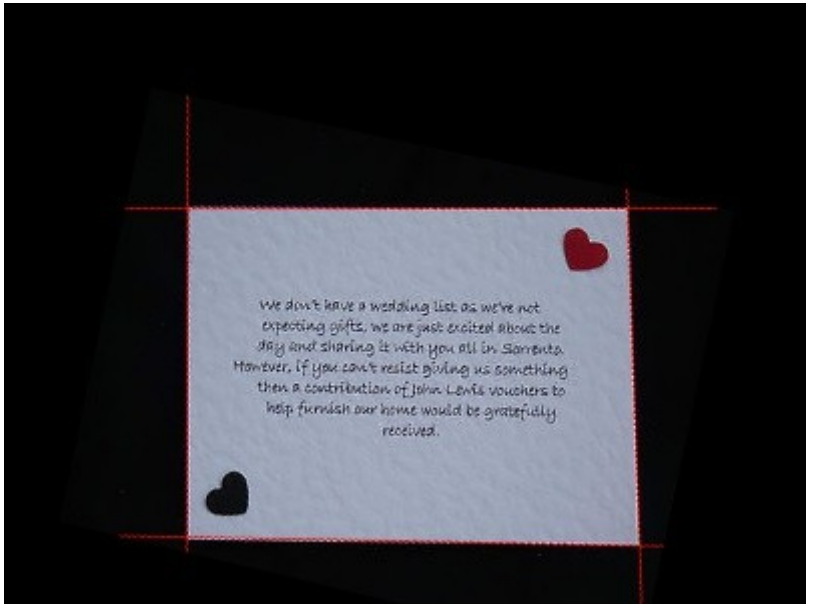
Sistema de Correção de Folha

entrada



saída

Encontre as bordas

- Importe o OpenCV, NumPy e MatPlotlib
- Leia a imagem usando o imread do Opencv
- Guarde o tamanho da imagem: linhas, colunas e canais de cores usando o shape. Informações interessantes para uso futuro.

- Altere para Escala de cinza:
  - gray = cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)
- Use a função Canny para encontrar as bordas
  - edges = cv2.Canny(gray,50,150,apertureSize = 3)
- Grave essa imagem usando o imwrite do OpenCV

Para ver as imagens lado a lado

- plt.subplot(121),plt.imshow(img),plt.title('Input')
- plt.subplot(122),plt.imshow(img2),plt.title('Output')
- plt.show()
-
- img e img2 ficam lado a lado

Use a Transformada de Hough

- A função é a:
- lines = cv2.HoughLines(edges,1,np.pi/180, LIMITE)
- s1, s2, s3 = lines.shape
- Deve ser feito um for para encontrar o valor de LIMITE para que a quantidade de linhas (s1) seja 4.

Conversão de Polar para Cartesiano

- Para cada linha, a variável lines tem o valor de rho e theta, converte-se para coordenada cartesiana e calcula dois pontos:
- for i in range(s1):
- rho = lines[i][0][0]
- theta = lines[i][0][1]
- a = np.cos(theta)
- b = np.sin(theta)
- x0 = a*rho
- y0 = b*rho
- x1 = int(x0 + 1000*(-b))
- y1 = int(y0 + 1000*(a))
- x2 = int(x0 - 1000*(-b))
- y2 = int(y0 - 1000*(a))

Encontrar a equação da reta dado dois pontos

- Use a função que recebe dois pontos:
- def line(p1, p2):
- A = (p1[1] - p2[1])
- B = (p2[0] - p1[0])
- C = (p1[0]*p2[1] - p2[0]*p1[1])
- return A, B, -C
-
- E a chame dentro do for para cada linha:
- L[i] = line([y1,x1], [y2,x2])

Desenho das retas

- Para compreender melhor, é interessante desenhar as retas na imagem, logo use a função line do OpenCV (dentro do for) para desenhar as retas
- cv2.line(img,(x1,y1),(x2,y2),(0,0,255),1)

- É interessante gravar esta imagem intermediária

Interseção das retas

- Função que encontra o ponto de interseção, dado duas retas:
- def intersection(L1, L2):
- D = L1[0] * L2[1] - L1[1] * L2[0]
- Dx = L1[2] * L2[1] - L1[1] * L2[2]
- Dy = L1[0] * L2[2] - L1[2] * L2[0]
- if D != 0:
- x = Dx / D
- y = Dy / D
- return x,y
- else:
- return False

Pontos de Interseção

- Como são 4 retas, que formam um retângulo, deve-se encontrar 4 pontos de interseção.
- 

Aplicando o Warp

- http://www.pyimagesearch.com/2014/08/25/4-point-opencv-getperspective-transform-example/
- M = cv2.getPerspectiveTransform(pts2, pts1)
- warp = cv2.warpPerspective(img, M, (400, 300))
-