**Histogram**

|  |  |
| --- | --- |
| **import cv2** **import matplotlib.pyplot as plt** **def rescale(frame, scale=0.7):**  **width = int(frame.shape[1] \* scale)**  **heigth = int(frame.shape[0] \* scale)**  **return cv2.resize(frame,(width,heigth),interpolation=cv2.INTER\_AREA)**  **img = cv2.imread('bien.jpg')** **img\_scale = rescale(img)** **cv2.imshow("img", img\_scale)** **img\_gray = cv2.cvtColor(img\_scale,cv2.COLOR\_RGB2GRAY)** **# cv2.imshow("img2", img\_gray)** **histogram = cv2.calcHist([img\_gray],[0],None,[256],[0,256])** **plt.figure()** **plt.title('Histogram')** **plt.plot(histogram)** **plt.show()** **cv2.waitKey(0)** | |
|  |  |

|  |  |
| --- | --- |
|  |  |

|  |  |
| --- | --- |
|  |  |

**Cân bằng histogram**

|  |  |
| --- | --- |
| **import cv2** **import matplotlib.pyplot as plt** **def rescale(frame, scale=0.8):**  **width = int(frame.shape[1] \* scale)**  **heigth = int(frame.shape[0] \* scale)**  **return cv2.resize(frame,(width,heigth),interpolation=cv2.INTER\_AREA)**  **img = cv2.imread('morning.jpg')** **img\_scale = rescale(img)** **img\_yuv = cv2.cvtColor(img\_scale,cv2.COLOR\_RGB2YUV)** **img\_yuv[:,:,0] = cv2.equalizeHist(img\_yuv[:,:,0])** **img\_equalize = cv2.cvtColor(img\_yuv,cv2.COLOR\_YUV2BGR)** **cv2.imshow("img", img\_equalize)** **histogram = cv2.calcHist([img\_equalize],[0],None,[256],[0,256])**  **plt.figure()** **plt.title('Histogram')** **plt.plot(histogram)** **plt.show()** **cv2.waitKey(0)** | |
|  |  |
|  |  |
|  |  |

**Laplacian**

|  |
| --- |
| **import cv2** **import matplotlib.pyplot as plt** **def rescale(frame, scale=0.8):**  **width = int(frame.shape[1] \* scale)**  **heigth = int(frame.shape[0] \* scale)**  **return cv2.resize(frame,(width,heigth),interpolation=cv2.INTER\_AREA)**   **img = cv2.imread("dinh-oc-vit-nganh-go-6-300x300.jpg")** **img\_scale = rescale(img)** **img\_gray = cv2.cvtColor(img\_scale,cv2.COLOR\_RGB2GRAY)** **# remove noise** **img = cv2.GaussianBlur(img\_gray,(3,3),0)**  **# convolute with proper kernels** **laplacian = cv2.Laplacian(img,cv2.CV\_64F)** **cv2.imshow("img", img)** **cv2.imshow("img\_laplacian", laplacian)**  **cv2.waitKey(0)** |

|  |  |
| --- | --- |
|  |  |
|  |  |

|  |  |
| --- | --- |
| **import cv2** **import matplotlib.pyplot as plt** **import numpy as np** **def rescale(frame, scale=0.8):**  **width = int(frame.shape[1] \* scale)**  **heigth = int(frame.shape[0] \* scale)**  **return cv2.resize(frame,(width,heigth),interpolation=cv2.INTER\_AREA)**  **kernel = np.ones((2, 2), np.uint8)** **img = cv2.imread("images.jpg")** **img\_scale = rescale(img)** **img\_gray = cv2.cvtColor(img\_scale,cv2.COLOR\_RGB2GRAY)** **# remove noise** **img\_blur = cv2.GaussianBlur(img\_gray,(3,3),0)**  **# convolute with proper kernels** **laplacian = cv2.Laplacian(img\_blur,cv2.CV\_64F)** **img\_erosion = cv2.erode(laplacian, kernel, iterations=1)** **img\_dilation = cv2.dilate(img\_erosion, kernel, iterations=1)** **cv2.imshow("img", img)** **cv2.imshow("img\_laplacian", laplacian)** **cv2.imshow("img\_erosion", img\_erosion)** **cv2.imshow("img\_dilation", img\_dilation)** **cv2.waitKey(0)** | |
|  |  |
|  |  |
|  |  |
|  |  |

**Edge**

|  |
| --- |
| **img = cv2.imread("images.jpg")** **img\_gray = cv2.cvtColor(img\_scale,cv2.COLOR\_RGB2GRAY)** **# remove noise** **img\_blur = cv2.GaussianBlur(img\_gray,(3,3),0)** **img\_edges = cv2.Canny(img\_blur,100,200)**  **cv2.imshow("img\_edge", img\_edges)** |

|  |  |
| --- | --- |
|  |  |
|  |  |
|  |  |

**Hough Transform**

|  |
| --- |
| import numpy as np import cv2  img = cv2.imread('image\kich-thuoc-ban-co-vua.png') img\_gray = cv2.cvtColor(img , cv2.COLOR\_BGR2GRAY) edges = cv2.Canny(img\_gray, 50, 150, apertureSize=3) lines = cv2.HoughLines(edges, 1, np.pi /180, 200)  for line in lines:  ro, theta = line[0]  a = np.cos(theta)  b = np.sin(theta)  x0 = a \* ro  y0 = b \* ro  x1 = int(x0 + 1000 \* (-b))  y1 = int(y0 + 1000 \* a)  x2 = int(x0 - 1000 \* (-b))  y2 = int(y0 - 1000 \* a)  cv2.line(img, (x1, y1), (x2, y2), (0, 0, 255), 2)  cv2.imshow("img", img) cv2.waitKey(0) |

|  |  |
| --- | --- |
|  |  |
|  |  |

**Hàm logarith, hàm mũ**

|  |
| --- |
| import cv2  import imutils  import numpy as np  **# logarithmic, ham mu**  img = cv2.imread('logimg.jpg')  img\_log = (np.log(img+1)/(np.log(1+np.max(img))))\*255  img\_log = np.array(img\_log, dtype=np.uint8)  cv2.imshow('log\_image', img\_log)  cv2.imshow('original\_img', img) |

|  |  |
| --- | --- |
|  |  |

**Mean filter, median filter, weighted filter**

|  |
| --- |
| kernel = np.ones((5, 5), np.float32)/25  kernel1 = np.array([[1, 2, 1],  [2, 4, 2],  [1, 2, 1]])/16  img = cv2.imread('noiseimg.png')  img\_median = cv2.medianBlur(img, 9)  img\_mean = cv2.filter2D(src=img, ddepth=0, kernel=kernel)  img\_weighted = cv2.filter2D(src=img, ddepth=0, kernel=kernel1)  cv2.imshow('orig\_img', img)  cv2.imshow('median\_image', img\_median)  cv2.imshow('mean\_img', img\_mean)  cv2.imshow('weighted\_img', img\_weighted) |

|  |  |
| --- | --- |
| *Ảnh gốc* | *Ảnh lọc trung bình* |
|  |  |
| *Ảnh lọc trung bình có trọng số* | *Ảnh lọc trung vị* |
|  |  |

**Sharpening, segmentation**

|  |
| --- |
| kernel2 = np.array([[0, 1, 0],  [1, -4, 1],  [0, 1, 0]])  img = cv2.imread('dep1.jpg')  img = imutils.resize(img, width=443, height=591)  img = cv2.cvtColor(img, cv2.COLOR\_BGR2GRAY)  cv2.imshow('origi\_img', img)  img = cv2.medianBlur(img, 7)  image\_sharp = cv2.filter2D(src=img, ddepth=0, kernel=kernel2)  s, thresh\_sharp = cv2.threshold(image\_sharp, 20, 255, cv2.THRESH\_BINARY)  cv2.imshow('sharp\_img', image\_sharp)  cv2.imshow('segmentation\_img', thresh\_sharp) |

|  |  |  |
| --- | --- | --- |
| *Ảnh gốc* | *Ảnh sau khi sharpening* | *Ảnh sau khi thersholding từ ảnh sharpening* |
|  |  |  |