ชื่อ นางสาวอัสวาณี อารง รหัสนิสิต 6621604700

จาก texture ต้นแบบ จำนวน 3 ภาพ ให้เขียนโปรแกรมเพื่อดำเนินการตามลำดับ ดังต่อไปนี้

|  |  |  |
| --- | --- | --- |
| Texture1 | Texture2 | Texture3 |
| D:\Documents\Desktop\texture\imgTexture\texture1.png | D:\Documents\Desktop\texture\imgTexture\texture2.png | D:\Documents\Desktop\texture\imgTexture\texture3.png |

0. เปิดไฟล์ภาพ texture แปลงเป็น grayscale

1. เขียนฟังก์ชัน calMeanGray(img) รับอินพุตเป็นภาพ grayscale และตอบกลับเป็นค่าเฉลี่ยระดับสี (Mean intensity) ที่ทำการ normalize แล้ว

|  |
| --- |
| def calMeanGray(img):      mean = np.mean(img)      normalize = mean / 255      return normalize  meanTexture = calMeanGray(texture\_gray)  print(meanTexture) |

เรียกใช้ฟังก์ชัน calMeanGray และบันทึกผลลัพธ์

|  |  |  |  |
| --- | --- | --- | --- |
|  | texture 1 | texture 2 | texture 3 |
| Normalize Mean intensity | 0.33 | 0.50 | 0.41 |

2. เขียนฟังก์ชัน calFEdgeness(img, threshold) คำนวณค่า Fedgeness โดยมีขั้นตอนการทำงานตามลำดับดังนี้

2.1 คำนวณค่าขอบภาพ โดยการทำ Sobel Edge Detection กับภาพอินพุต

2.2 ปรับภาพที่ได้เป็นค่าบวกทั้งหมด โดยใช้ absolute

2.3 นับจำนวน pixel ที่มีค่าขอบ มากกว่าค่า threshold ที่กำหนด

2.4 Fedgeness = จำนวน pixel ที่เป็นขอบภาพ / จำนวน pixel ทั้งหมด

|  |
| --- |
| def calFEdgeness(img, threshold) :      sobel = cv2.Sobel(img, cv2.CV\_64F, 1, 1, ksize=5)      sobelABS = np.abs(sobel)      totalPixels = sobel.size      totalEdgePixels = np.sum(sobelABS > threshold)      fedgeness =  (float) (totalEdgePixels / totalPixels)      return fedgeness |

เรียกใช้ฟังก์ชัน และบันทึกผลลัพธ์

|  |  |  |  |
| --- | --- | --- | --- |
|  | texture 1 | Texture 2 | texture 3 |
| Fedgeness | 0.8 | 0.82 | 0.57 |

3. เขียนฟังก์ชัน calTextureHis(img) รับภาพอินพุต เพื่อสร้าง Texture Histogram ของภาพ โดยประกอบด้วยค่า normalize mean intensity และ Fedgeness ในรูปแบบ numpy array

[normalize mean intensity , Fedgeness] เช่น [0.27, 0.15]

โดยมีขั้นตอนการทำงานตามลำดับดังนี้

3.1 สร้าง numpy array ว่าง

3.2 เพิ่มค่า Normalize Mean intensity ของภาพลงใน array โดยใช้ calMeanGray()

3.3 เพิ่มค่า Fedgeness ลงใน array โดยใช้ calFEdgeness() กำหนด threshold = 100

|  |
| --- |
| def calTextureHis(img):      texture\_histogram = np.array([])      texture\_histogram = np.append(texture\_histogram, calMeanGray(img))      texture\_histogram = np.append(texture\_histogram, calFEdgeness(img, 100))      return texture\_histogram |

เรียกใช้ฟังก์ชัน และบันทึกผลลัพธ์

|  |  |  |  |
| --- | --- | --- | --- |
|  | texture 1 | Texture 2 | texture 3 |
| Texture Histogram | [0.33 0.81] | [0.49 0.82] | [0.41 0.57] |

4. เขียนฟังก์ชัน calL1Dist(his1,his2) รับ histogram 2 ชุด เพื่อคำนวณค่า L1 distance

|  |
| --- |
| def calL1Dist(his1,his2):      L1Distnp = np.abs(his1[0]-his2[0])+np.abs(his1[1]-his2[1])      return L1Distnp |

เรียกใช้ฟังก์ชัน และบันทึกผลลัพธ์

|  |  |  |  |
| --- | --- | --- | --- |
|  | texture 1 vs texture2 | Texture 1 vs texture3 | Texture2 vs texture 3 |
| L1 | 0.17 | 0.32 | 0.33 |

5. เขียนฟังก์ชัน textureOverlay(img, texture, L1Threshold) รับภาพอินพุต ภาพ texture และค่าL1 distance Threshold เพื่อทำการ overlay ภาพอินพุต ที่ตรงกับภาพ texture ตามค่า threshold ที่กำหนด โดยมีการทำงานดังนี้

|  |
| --- |
| def textureOvelay(img, texture, threshold) :      imgOut = np.copy(img)      hisTexture = calTextureHis(texture)      windowSize = 21      r, c = img.shape      for i in range(r-windowSize+1):          for j in range(c-windowSize+1):              subImg = img[i:i+windowSize, j:j+windowSize]              hisSubImg = calTextureHis(subImg)              if calL1Dist(hisTexture, hisSubImg) < threshold:                  imgOut[i+windowSize//2][j+windowSize//2] = 255      return imgOut |

เรียกใช้ฟังก์ชัน textureOvelay ระหว่างภาพ กับ texture โดยเปลี่ยนค่า L1Threshold และแสดงภาพผลลัพธ์ ดังนี้

|  |  |  |  |
| --- | --- | --- | --- |
| L1  Threshold | Img1+texture 1 | Img2+Texture 2 | Img3+Texture 3 |
| 0.2 |  |  |  |
| 0.3 |  |  |  |
| 0.4 |  |  |  |

|  |
| --- |
| NOTE : Numpy function  np.average() – หาค่าเฉลี่ย  np.abs() – ค่า absolute  np.sum() – ผลรวม  np.copy() – copy numpy array  np.count\_nonzero(condition) นับจำนวน element ที่ตรงกับเงื่อนไข  np.append – เพิ่ม element ลงใน numpy array |

**Homework**

ปรับปรุงการนำอัลกอริธึมจากโค้ดตัวอย่าง ไปใช้งานโดย

* กำหนดโจทย์ปัญหา ภาพตัวอย่าง 3-4 ภาพ
* แสดงภาพ Overlay ก่อนปรับปรุง
* เลือกดำเนินการปรับปรุงอย่างน้อยหนึ่งแนวทาง จากแนวทางดังต่อไปนี้
* เปรียบเทียบภาพ ต้นฉบับ ภาพ Overlay ก่อนปรับปรุง Overlay หลับปรับปรุง
* สรุปการดำเนินงานสั้นๆ 3-4 บรรทัด

แนวทางปรับปรุงอัลกอริธึม

|  |
| --- |
| 1. เพิ่ม feature Histogram ด้วยการวัดค่า texture ด้วยวิธีอื่นๆ แสดงภาพตัวอย่าง และ |
| 2. ใช้ภาพ texture ที่คล้ายกันหลายๆ ภาพ ปรับโปรแกรม โดยการทำ overlay จากหลาย Texture ร่วมกัน เพื่อให้การทำ segmentation ออกมาดีที่สุด |
| 3. ทดลองปรับค่า parameter ในหลายๆ แบบ และสรุปค่า parameter ที่เหมาะสม |
| 4 อื่นๆ ปรึกษากับอาจารย์ |