**Segmentasi Citra**

Lembar Kerja Praktikum 7

|  |
| --- |
| Nama : Muhamad Hilmy Haidar |
| NIM : G64170030 |
| Tanggal : 27 Februari 2020 |
| Nama Asisten :   1. Hilmi Farhan Ramadhani G64160048 2. Kautsar Ibrahim Hilmi G64160073 |

Pelajari dan jalankan code Python mengenai image segmentation di bawah ini. Apa keluaran dari program tersebut? Jelaskan cara kerja image segmentation pada kode tersebut.

|  |
| --- |
| """ ================================================== Comparing edge-based and region-based segmentation ================================================== In this example, we will see how to segment objects from a background. We use the ``coins`` image from ``skimage.data``, which shows several coins outlined against a darker background. """  import numpy as np import matplotlib.pyplot as plt  from skimage import data from skimage.exposure import histogram  coins = data.coins() hist, hist\_centers = histogram(coins) fig, axes = plt.subplots(1, 2, figsize=(8, 3)) axes[0].imshow(coins, cmap=plt.cm.gray) axes[0].axis('off') axes[1].plot(hist\_centers, hist, lw=2) axes[1].set\_title('histogram of gray values')  ###################################################################### # Thresholding # ============ # A simple way to segment the coins is to choose a threshold based on the # histogram of gray values. Unfortunately, thresholding this image gives a # binary image that either misses significant parts of the coins or merges # parts of the background with the coins:  fig, axes = plt.subplots(1, 2, figsize=(8, 3), sharey=True) axes[0].imshow(coins > 100, cmap=plt.cm.gray) axes[0].set\_title('coins > 100') axes[1].imshow(coins > 150, cmap=plt.cm.gray) axes[1].set\_title('coins > 150')  for a in axes:  a.axis('off')  plt.tight\_layout()  ###################################################################### # Edge-based segmentation # ======================= # Next, we try to delineate the contours of the coins using edge-based # segmentation. To do this, we first get the edges of features using the # Canny edge-detector.  from skimage.feature import canny  edges = canny(coins) fig, ax = plt.subplots(figsize=(4, 3)) ax.imshow(edges, cmap=plt.cm.gray) ax.set\_title('Canny detector') ax.axis('off')  ###################################################################### # These contours are then filled using mathematical morphology.  from scipy import ndimage as ndi  fill\_coins = ndi.binary\_fill\_holes(edges) fig, ax = plt.subplots(figsize=(4, 3)) ax.imshow(fill\_coins, cmap=plt.cm.gray) ax.set\_title('filling the holes') ax.axis('off')   ###################################################################### # Small spurious objects are easily removed by setting a minimum size for # valid objects.  from skimage import morphology  coins\_cleaned = morphology.remove\_small\_objects(fill\_coins, 21) fig, ax = plt.subplots(figsize=(4, 3)) ax.imshow(coins\_cleaned, cmap=plt.cm.gray) ax.set\_title('removing small objects') ax.axis('off')  ###################################################################### # However, this method is not very robust, since contours that are not # perfectly closed are not filled correctly, as is the case for one unfilled # coin above. # # Region-based segmentation # ========================= # We therefore try a region-based method using the watershed transform. # First, we find an elevation map using the Sobel gradient of the image.  from skimage.filters import sobel  elevation\_map = sobel(coins) fig, ax = plt.subplots(figsize=(4, 3)) ax.imshow(elevation\_map, cmap=plt.cm.gray) ax.set\_title('elevation map') ax.axis('off')  ###################################################################### # Next we find markers of the background and the coins based on the extreme # parts of the histogram of gray values.  markers = np.zeros\_like(coins) markers[coins < 30] = 1 markers[coins > 150] = 2  fig, ax = plt.subplots(figsize=(4, 3)) ax.imshow(markers, cmap=plt.cm.nipy\_spectral) ax.set\_title('markers') ax.axis('off')  ###################################################################### # Finally, we use the watershed transform to fill regions of the elevation # map starting from the markers determined above:  segmentation = morphology.watershed(elevation\_map, markers) fig, ax = plt.subplots(figsize=(4, 3)) ax.imshow(segmentation, cmap=plt.cm.gray) ax.set\_title('segmentation') ax.axis('off')  ###################################################################### # This last method works even better, and the coins can be segmented and # labeled individually.  from skimage.color import label2rgb  segmentation = ndi.binary\_fill\_holes(segmentation - 1) labeled\_coins, \_ = ndi.label(segmentation) image\_label\_overlay = label2rgb(labeled\_coins, image=coins) fig, axes = plt.subplots(1, 2, figsize=(8, 3), sharey=True) axes[0].imshow(coins, cmap=plt.cm.gray) axes[0].contour(segmentation, [0.5], linewidths=1.2, colors='y') axes[1].imshow(image\_label\_overlay)  for a in axes:  a.axis('off')  plt.tight\_layout() plt.show() |

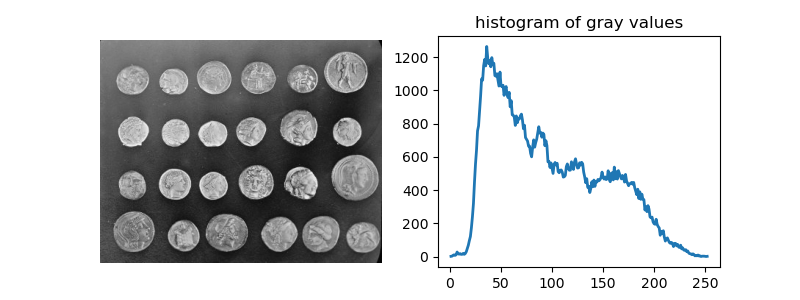
Kode di atas merupakan kode segmentasi citra pada data coins. Keluaran dari program tersebut adalah gambar-gambar hasil pemrosesan yang memiliki hasil akhir berupa pendeteksian citra-citra koin yang berbeda menjadi warna-warna yang berbeda.

Berikut merupakan penjelasan cara kerja dari kode tersebut :

1. Gunakan data coins pada library skimage.data, Lalu buat plot grayscale nya.

|  |
| --- |
| import numpy as np import matplotlib.pyplot as plt  from skimage import data from skimage.exposure import histogram  coins = data.coins() hist, hist\_centers = histogram(coins) fig, axes = plt.subplots(1, 2, figsize=(8, 3)) axes[0].imshow(coins, cmap=plt.cm.gray) axes[0].axis('off') axes[1].plot(hist\_centers, hist, lw=2) axes[1].set\_title('histogram of gray values') |

Gambar keluaran dari hasil plot menggunakan matplotlib.pyplot adalah sebagai berikut :

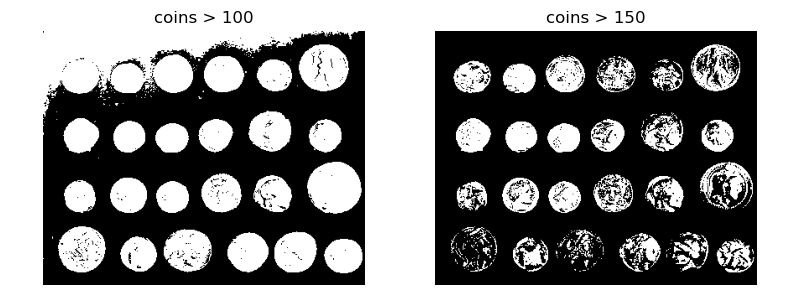


Dari gambar di atas, kita dapat mendapatkan informasi bahwa gambar coins tersebut memiliki perbedaan yang signifikan dengan background nya. Cara termudah untuk memisah-misahkan gambar (segmentasi) dari citra tersebut adalah dengan memberikan nilai ambang batas (threshold).

1. Lakukan percobaan threshold pada gambar tersebut :

|  |
| --- |
| fig, axes = plt.subplots(1, 2, figsize=(8, 3), sharey=True) axes[0].imshow(coins > 100, cmap=plt.cm.gray) axes[0].set\_title('coins > 100') axes[1].imshow(coins > 150, cmap=plt.cm.gray) axes[1].set\_title('coins > 150') |

Hasil thresholding dari citra coin dengan nilai 100 dan 150 pada nilai grayscale :

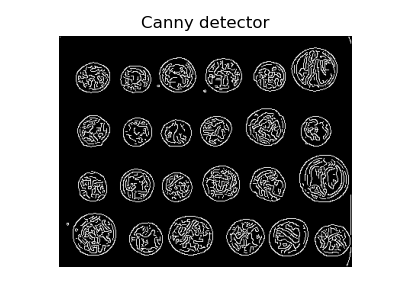


Sayangnya hasil threshold tersebut kurang memuaskan. Jika kita melakukan threshold pada nilai di atas 100, background akan terpotong. Sedangkan dengan threshold nilai 150, gambar koin-koin bisa terseleksi namun memiliki lubang-lubang dari pixel yang tidak terseleksi. Diperlukan adanya teknik seleksi yang bisa menyeleksi gambar tanpa meninggalkan lubang-lubang pixel yang tidak terseleksi.

1. Edge-based segmentation untuk menyeleksi tepian koin, yang paling optimal kita gunakan Canny Edge Detection :

|  |
| --- |
| from skimage.feature import canny  edges = canny(coins) fig, ax = plt.subplots(figsize=(4, 3)) ax.imshow(edges, cmap=plt.cm.gray) ax.set\_title('Canny detector') ax.axis('off') |

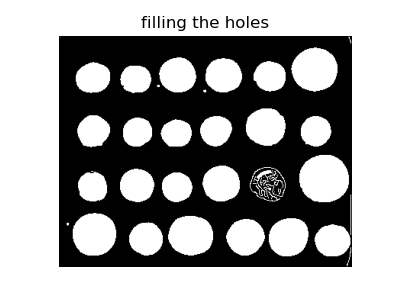
Berikut hasil deteksi tepi dari algoritma canny :



Setelah mengidentifikasi tepian dari koin, kita melakukan pengisian pada bagian dalam koin :

|  |
| --- |
| from scipy import ndimage as ndi  fill\_coins = ndi.binary\_fill\_holes(edges) fig, ax = plt.subplots(figsize=(4, 3)) ax.imshow(fill\_coins, cmap=plt.cm.gray) ax.set\_title('filling the holes') ax.axis('off') |

Gambar setelah tepi-tepi koin diisi dengan menggunakan fungsi binary fill holes :

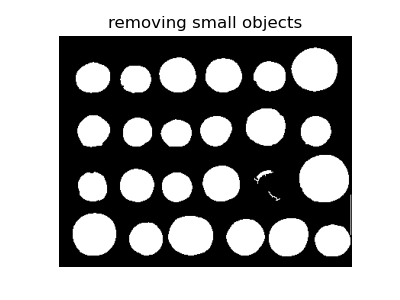


Terdapat kecacatan pada salah satu koin yang menandakan bahwa proses mengisi lubang tersebut masih kurang optimal. Hal tersebut kemungkinan terjadi karena adanya permukaan yang tidak tertutup dengan tepian secara sempurna.

Kita coba lakukan bla bla bla dengan menghapus objek-objek yang kecil dari gambar :

|  |
| --- |
| from skimage import morphology  coins\_cleaned = morphology.remove\_small\_objects(fill\_coins, 21) fig, ax = plt.subplots(figsize=(4, 3)) ax.imshow(coins\_cleaned, cmap=plt.cm.gray) ax.set\_title('removing small objects') ax.axis('off') |

Hasil gambarnya yaitu :

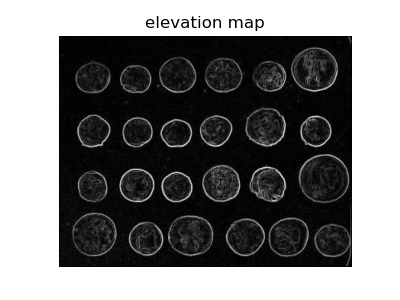


Dengan Edge-based segmentation, kita berhasil mendapatkan gambar koin yg cukup optimal. Namun, terdapat satu koin yang tidak bisa diidentifikasi karena saat dideteksi tepinya ternyata tidak tertutup secara sempurna.

1. Segmentasi menggunakan Region-based. Ubah terlebih dahulu citra koin menjadi bentuk sobel :

|  |
| --- |
| from skimage.filters import sobel  elevation\_map = sobel(coins) fig, ax = plt.subplots(figsize=(4, 3)) ax.imshow(elevation\_map, cmap=plt.cm.gray) ax.set\_title('elevation map') ax.axis('off') |

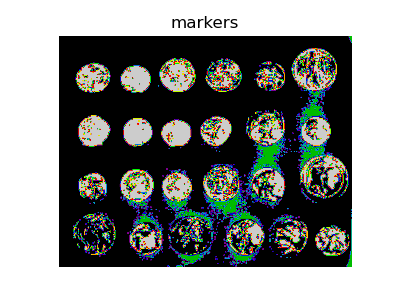
Gambar coins yang telah diberi filter sobel :



Terlihat kalau tepian koin semakin menebal sehingga tepian tersebut dapat membedakan koin dari backgroundnya. Selanjutnya coba untuk mencari penanda(marker) keberadaan tiap koin :

|  |
| --- |
| markers = np.zeros\_like(coins) markers[coins < 30] = 1 markers[coins > 150] = 2  fig, ax = plt.subplots(figsize=(4, 3)) ax.imshow(markers, cmap=plt.cm.nipy\_spectral) ax.set\_title('markers') ax.axis('off') |

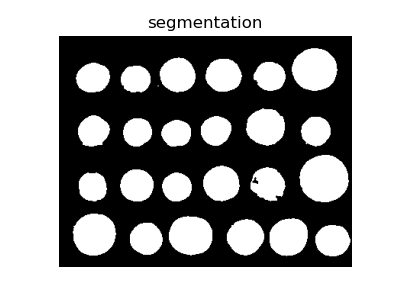
Keluaran marker :



Setelah marker diperoleh, kita lanjutkan dengan segmentasi watershed :

|  |
| --- |
| segmentation = morphology.watershed(elevation\_map, markers) fig, ax = plt.subplots(figsize=(4, 3)) ax.imshow(segmentation, cmap=plt.cm.gray) ax.set\_title('segmentation') ax.axis('off') |

Hasil segmentasi nya :

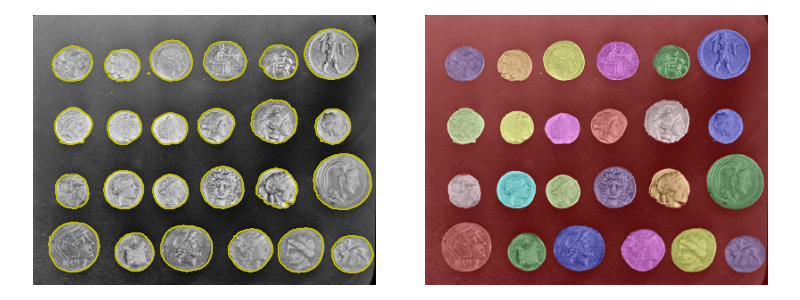


Gambar segmentasi yang diberikan dengan menggunakan transformasi watershed memiliki hasil yang lebih baik daripada sebelumnya karena hanya terdapat sedikit lubang pada satu koin.

1. Pada metode terakhir ini kita melanjutkan segmentasi hasil Region-based tadi :

|  |
| --- |
| from skimage.color import label2rgb  segmentation = ndi.binary\_fill\_holes(segmentation - 1) labeled\_coins, \_ = ndi.label(segmentation) image\_label\_overlay = label2rgb(labeled\_coins, image=coins) fig, axes = plt.subplots(1, 2, figsize=(8, 3), sharey=True) axes[0].imshow(coins, cmap=plt.cm.gray) axes[0].contour(segmentation, [0.5], linewidths=1.2, colors='y') axes[1].imshow(image\_label\_overlay) |

Dari segmentasi sebelumnya, kita dapat melihat masih ada satu koin yang memiliki lubang. Oleh karena itu kita lakukan lagi pengisian lubang dengan binary fill holes. Setelah semua lubang terisi, kita beri label berbeda pada setiap koin. Setelah itu setiap label diberi warna yang berbeda. Berikut hasilnya :



Gambar sebelah kanan di atas adalah citra hasil segmentasi yang telah diberi label berbeda pada setiap koin. Koin-koin tersebut teridentifikasi secara optimal oleh segmentasi yang berjenis Region-based.