**“TRANSFORMASI HOUGH CIRCLES PADA PYTHON”**



Oleh :

GDE ANGGA PRATAMA NUGRAHA (1504505013)

I NYOMAN ADI TRIGINARSA (1504505026)

PUTU ANDRE MAHAPUTRA (1504505029)

**JURUSAN TEKNOLOGI INFORMASI**

**FAKULTAS TEKNIK**

**UNIVERSITAS UDAYANA**

**CONNECTED COMPONENT LABELING**

*Connected component labeling* merupakan teknik yang juga bisa digunakan untuk mengklasifikasikanregion atau objek dalam citra digital. Teknik ini memanfaatkan teori *connectivity* piksel pada citra. Piksel-piksel dalam region disebut *connected* (ada konektifitasnya atau *connectivity*) bila mematuhi aturan *adjacency* atau aturan “kedekatan” piksel. Aturan kedekatan piksel ini memanfaatkan sifat ketetanggaan piksel. Dengan demikian piksel-piksel yang di katakan *connected* pada dasarnya memiliki sifat *adjacency* satu sama lain karena mereka masih memiliki hubungan *neighbourhood* atau ketetanggaan. Perlu diingat, bahwa citra yang bisa diolah dengan menggunakan metode ini adalah citra grayscale atau citra biner. Ketetanggaan harus memiliki panjang atau jarak 1 unit (langsung antara piksel dengan piksel tanpa ada perantara nya). Menurut Gonzales dan Woods (1992, p40), ada dua jenis konektivitas yang digunakan pada citra dua dimensi yakni 4-Konektivitas (4-Connected Neighbors) dan 8-Konektivitas (8-Connected Neighbors).

1. 4-Konektivitas (4-Connected Neighbors)

Berikut ini merupakan contoh gambar dari 4-Konektivitas (4-Connected Neighbors) yang digunakan pada citra dua dimensi.

|  |
| --- |
|  |

**Gambar 1** 4-Konektivitas (4-Connected Neighbors)

Piksel-piksel yang berdekatan dikatakan memiliki hubungan 4-konektivitas jika piksel-piksel tersebut terletak berdampingan secara horizontal dan vertical N4(P). Kumpulan dari piksel-piksel ini disebut dengan 4 neighbors of P. Pada konsep 4-Connected Neighbors bila terdapat 2 piksel yang bersinggungan secara diagonal maka akan dianggap 2 objek.

1. 8-Konektivitas (8-Connected Neighbors)

Berikut ini merupakan contoh gambar dari 8-Konektivitas (8-Connected Neighbors) yang digunakan pada citra dua dimensi.

|  |
| --- |
|  |

**Gambar 2** 8-Konektivitas (8-Connected Neighbors)

Piksel-piksel yang berdekatan dikatakan memiliki hubungan 8-konektivitas jika piksel-piksel tersebut terletak berdampingan secara horizontal dan vertical N8(P) atau disebut juga empat diagonal neighbors. Pada konsep 8-Connected Neighbors bila terdapat 2 piksel yang bersinggungan baik secara diagonal maupun secara horizontal dan vertical maka akan dianggap 1 objek.

**PERBANDINGAN 4-KONEKTIVITAS DAN 8-KONEKTIVITAS**

Dengan konfigurasi 4 konektivitas, pengecekan label hanya dilakukan pada 2 tetangga terdekat saja yaitu atas dan bawah. Sebaliknya dengan konfigurasi 8 konektivitas, pengecekan label dilakukan dengan 4 tetangga terdekat. Kelebihan dan kekurangan dari masing-masing konfigurasi jumlah konektivitas ini bergantung pada penerapan aplikasi metode CCL itu sendiri. Sebagai contoh, metode CCL dengan 4 konektivitas akan tidak efektif apabila digunakan untuk mendeteksi pola garis miring pada citra biner. Hal ini dikarenakan metode CCL dengan 4 konektivitas hanya mampu melabeli piksel terdekat yang berada pada baris ataupun kolom yang sama dengan piksel yang ditinjau. Penggunaan konfigurasi 8 konektivitas juga akan memerlukan waktu pemrosesan lebih lama dikarenakan jumlah tetangga terdekat yang dicek lebih banyak dibandngkan dengan konfigurasi 4 konektivitas. Untuk lebih jelasnya bisa dilihat pada program yang telah kami buat.

**IMPLEMENTASI PROGRAM CCL**

Berikut ini merupakan kode program dari implementasi program connected component labeling dengan menggunakan dua buah label yakni label 4-connectivity dan label 8-connectivity.

|  |
| --- |
| import cv2  import numpy as np  from matplotlib import pyplot as plt  #prosedur 8-connectivity  def delapanlabel(gambar):  #membuat 8-connectivity pada gambar  ret, labels = cv2.connectedComponents(gambar, connectivity=8)  # memberi nilai hue pada gambar yang telah di labelkan  label\_hue = np.uint8(179 \* labels / np.max(labels))  #mengalikan warna hue dengan nilai 255 untuk mendapatkan nilai maksimal  nilai\_maks = 255 \* np.ones\_like(label\_hue)  #menggabungkan warna hue awal dengan warna maksimal agar memiliki 3 nilai untuk di rubah ke bentuk RGB  label\_rgb = cv2.merge([label\_hue, nilai\_maks, nilai\_maks])  # mengubah gambar hue menjadi RGB  label\_rgb = cv2.cvtColor(label\_rgb, cv2.COLOR\_HSV2RGB)  # mengatur agar background tetap hitam  label\_rgb[label\_hue == 0] = 0  return label\_rgb  def empatlabel(gambar):  #membuat 8-connectivity pada gambar  ret, labels = cv2.connectedComponents(gambar, connectivity=4)  # memberi nilai hue pada gambar yang telah di labelkan  label\_hue = np.uint8(179 \* labels / np.max(labels))  #mengalikan warna hue dengan nilai 255 untuk mendapatkan nilai maksimal  nilai\_maks = 255 \* np.ones\_like(label\_hue)  #menggabungkan warna hue awal dengan warna maksimal agar memiliki 3 nilai untuk di rubah ke bentuk RGB  label\_rgb = cv2.merge([label\_hue, nilai\_maks, nilai\_maks])  # mengubah gambar hue menjadi RGB  label\_rgb = cv2.cvtColor(label\_rgb, cv2.COLOR\_HSV2RGB)  # mengatur agar background tetap hitam  label\_rgb[label\_hue == 0] = 0  return label\_rgb  #main program  #proses membaca gambar  gambar = cv2.imread('a.png', 0)  #menyalin gambar menjadi gambar\_asli  gambar\_asli = gambar  #mengubah gambar menjadi biner dengan threshold diatas 127 menjadi nilai 1  gambar = cv2.threshold(gambar, 127, 255, cv2.THRESH\_BINARY)[1]  #memanggil prosedur empatlabel  hasil1 = empatlabel(gambar)  #memanggin prosedur delapanlabel  hasil2 = delapanlabel(gambar)  #membuat tampilan citra awal  plt.subplot(131),plt.imshow(gambar\_asli, cmap='gray')  plt.title('Citra Awal'), plt.xticks([]), plt.yticks([])  #membuat tampilan citra hasil 4-connectivity  plt.subplot(132),plt.imshow(hasil1)  plt.title('Hasil 4-Connectivity'), plt.xticks([]), plt.yticks([])  #membuat tampilan citra hasil 8-connectivity  plt.subplot(133),plt.imshow(hasil2)  plt.title('Hasil 8-Connectivity'), plt.xticks([]), plt.yticks([])  #menampilkan gambar  plt.show() |

**Kode Program 1** Program Connected Component Labelling

Pada program connected component labelling terdapat dua buah prosedur yang masing-masing bertujuan untuk menggunakan label 4-connectivity dan label 8-connectivity. Dalam prosedur terdapat beberapa langkah untuk melakukan proses labeling tergantung dengan tipe labelnya. Langkah pertama adalah melakukan labeling dengan tipe tertentu dengan kode program ret, labels = cv2.connectedComponents(gambar, connectivity=8). Selanjutnya memberi nilai HUE berdasarkan pengelompokan label yang telah dibuat dan mengubah warna HUE menjadi RGB. Pada main program langkah yang dilakukan adalah membaca gambar yang akan di proses. Kemudian mengubah gambar tersebut menjadi biner dengan threshold apabila nilai piksel dibawah 179 maka nilainya dirubah menjadi 0 apabila diatas 179 maka nilainya akan dirubah menjadi 1. Langkah berikutnya memanggil prosedur empatlabel() untuk proses labelling dengan 4-connectivity dan prosedur delapanlabel() untuk proses labelling dengan 8-connectivity dan menampilkannya dengan matplotlab. Berikut ini merupakan hasil keluaran program.

|  |
| --- |
|  |

**Gambar 3** Hasil Program CCL

Dari hasil yang diperoleh dapat dilihat perbedaan dari hasil label 4-connectivity dengan hasil label 8-connectivity. Pada hasil label 4-connectivity terdapat 4 label yang memiliki warna yang berbeda sedangkan hasil dari label 8-connectivity memiliki 3 label yang memiliki warna yang berbeda hal ini disebabkan karena 8-conectivity membaca piksel diagonal kemudian menjadikannya 1 label.

**DILASI**

Dilasi adalah teknik untuk memperbesar segmen objek (citra biner) dengan menambah lapisan disekeliling objek. Atau dengan menjadi titik latar (0) yang bertetangga dengan titik objek (1) menjadi titik objek (1). Berikut ini merupakan contoh dari proses dilasi.

|  |
| --- |
| https://i2.wp.com/devtrik.com/wp-content/uploads/2017/04/morfologi-dilasi.png?w=1013&ssl=1 |

**Gambar 4** Proses Dilasi

Berikut ini merupakan kode program untuk implementasi dilasi pada python menggunakan opencv.

|  |
| --- |
| import cv2  import numpy as np  from matplotlib import pyplot as plt  #membaca gambar  img = cv2.imread('contoh.png',0)  #konversi biner dengan threshold apabila minValue 127 dan MaxValue 255 pada sebuah pixel maka nilainya 1  ret, thresh = cv2.threshold(img, 127, 255, cv2.THRESH\_BINARY)  #membuat sebuah kernel 3x3 tipe 8bit  kernel = np.ones((3,3),np.uint8)  #proses dilasi dengan menggunakan thresh sebagai background dan kernel sebagai foreground dengan sekali iterasi 1  dilasi = cv2.dilate(thresh,kernel,iterations = 1)  #membuat tampilan citra awal  plt.subplot(131),plt.imshow(img,cmap = 'gray')  plt.title('Citra Awal'), plt.xticks([]), plt.yticks([])  #membuat tampilan citra biner saat di threshold  plt.subplot(132),plt.imshow(thresh,cmap = 'gray')  plt.title('Citra Biner'), plt.xticks([]), plt.yticks([])  #membuat tampilan citra hasil dilasi  plt.subplot(133),plt.imshow(dilasi,cmap = 'gray')  plt.title('Hasil Dilasi'), plt.xticks([]), plt.yticks([])  #menampilkan gambar  plt.show() |

**Kode Program 2** Program Dilasi

Pada proses dilasi langkah pertama adalah membaca citra yang akan di proses kemudian dilakukan konversi gambar ke biner dimana threshold dari nilainya apabila nilai sebuah piksel dibawah 127 maka akan bernilai 0 apabila nilai piksel diatas 127 maka akan bernilai 1. Langkah selanjutnya adalah membuat kernel 3x3 bertipe 8bit dan melakukan proses dilasi kemudian menampilkannya. Berikut ini merupakan hasil dari program dilasi.

|  |
| --- |
|  |

**Gambar 5** Hasil Proses Dilasi

Hasil dilasi pada program mengakibatkan gambar yang sebelumnya yang terdapat teks yang terdapat garis hitam dan tidak bersambung menjadikan garis hitam pada teks hilang dan tulisan menjadi menyambung. Ini dikarenakan saat melakukan proses dilasi nilai biner 1 bertambah atau mengembang.

**EROSI**

Erosi atau pengikisan adalah kebalikan dari dilasi yaitu teknik yang bertujuan untuk memperkecil atau mengikis tepi objek. Atau dengan menjadi titik objek (1) yang bertetangga dengan titik latar (0) menjadi titik latar (0). Berikut ini merupakan contoh dari proses erosi.

|  |
| --- |
| https://i1.wp.com/devtrik.com/wp-content/uploads/2017/04/morfologi-erosi.png?w=793&ssl=1 |

**Gambar 6** Proses Erosi

Berikut ini merupakan kode program untuk implementasi dilasi pada python menggunakan opencv.

|  |
| --- |
| import cv2  import numpy as np  from matplotlib import pyplot as plt  #membaca gambar  img = cv2.imread('contoh.png',0)  #konversi biner dengan threshold apabila minValue 127 dan MaxValue 255 pada sebuah pixel maka nilainya 1  ret, thresh = cv2.threshold(img, 127, 255, cv2.THRESH\_BINARY)  #membuat sebuah kernel 3x3 tipe 8bit  kernel = np.ones((3,3),np.uint8)  #proses erosi dengan menggunakan thresh sebagai background dan kernel sebagai foreground dengan sekali iterasi 1  erosi = cv2.erode(thresh,kernel,iterations = 1)  #membuat tampilan citra awal  plt.subplot(131),plt.imshow(img,cmap = 'gray')  plt.title('Citra Awal'), plt.xticks([]), plt.yticks([])  #membuat tampilan citra biner saat di threshold  plt.subplot(132),plt.imshow(thresh,cmap = 'gray')  plt.title('Citra Biner'), plt.xticks([]), plt.yticks([])  #membuat tampilan citra hasil dilasi  plt.subplot(133),plt.imshow(erosi,cmap = 'gray')  plt.title('Hasil Erosi'), plt.xticks([]), plt.yticks([])  #menampilkan gambar  plt.show() |

**Kode Program 3** Program Erosi

Pada proses erosi langkah pertama adalah membaca citra yang akan di proses kemudian dilakukan konversi gambar ke biner dimana threshold dari nilainya apabila nilai sebuah piksel dibawah 127 maka akan bernilai 0 apabila nilai piksel diatas 127 maka akan bernilai 1. Langkah selanjutnya adalah membuat kernel 3x3 bertipe 8bit dan melakukan proses erosi kemudian menampilkannya. Berikut ini merupakan hasil dari program erosi.

|  |
| --- |
|  |

**Gambar 7** Hasil Program Erosi

Hasil dilasi pada program mengakibatkan gambar yang sebelumnya terdapat teks yang bergaris hitam menjadikan garis tersebut lebih nampak. Disamping itu teks dan garis menjadi mengkerut. Ini disebabkan karena nilai biner 1 berkurang atau mengkerut.

**OPENING**

Opening adalah proses erosi yang diikuti dengan dilasi. Dimulai dengan melakukan erosi pada citra kemudian hasil tersebut kembali dilakukan erosi. Opening biasanya digunakan untuk menghilangkan objek-objek kecil dan kurus serta dapat membuat tepi citra lebih smooth (untuk citra berukuran besar).

|  |
| --- |
|  |

**Gambar 8** Proses Opening

Berikut ini merupakan kode program untuk implementasi opening pada python menggunakan opencv.

|  |
| --- |
| import cv2  import numpy as np  from matplotlib import pyplot as plt  #membaca gambar  img = cv2.imread('contoh.png',0)  #konversi biner dengan threshold apabila minValue 127 dan MaxValue 255 pada sebuah pixel maka nilainya 1  ret, thresh = cv2.threshold(img, 127, 255, cv2.THRESH\_BINARY)  #membuat sebuah kernel 3x3 tipe 8bit  kernel = np.ones((3,3),np.uint8)  #proses opening dengan melakukan erosi terlebih dahulu kemudian dilasi  opening = cv2.morphologyEx(thresh,cv2.MORPH\_OPEN, kernel, iterations = 2)  #membuat tampilan citra awal  plt.subplot(131),plt.imshow(img,cmap = 'gray')  plt.title('Citra Awal'), plt.xticks([]), plt.yticks([])  #membuat tampilan citra biner saat di threshold  plt.subplot(132),plt.imshow(thresh,cmap = 'gray')  plt.title('Citra Biner'), plt.xticks([]), plt.yticks([])  #membuat tampilan citra hasil opening  plt.subplot(133),plt.imshow(opening,cmap = 'gray')  plt.title('Hasil Opening'), plt.xticks([]), plt.yticks([])  #menampilkan gambar  plt.show() |

**Kode Program 4** Program Opening

Pada proses opening langkah pertama adalah membaca citra yang akan di proses kemudian dilakukan konversi gambar ke biner dimana threshold dari nilainya apabila nilai sebuah piksel dibawah 127 maka akan bernilai 0 apabila nilai piksel diatas 127 maka akan bernilai 1. Langkah selanjutnya adalah membuat kernel 3x3 bertipe 8bit dan melakukan proses opening kemudian menampilkannya. Berikut ini merupakan hasil dari program opening.

|  |
| --- |
|  |

**Gambar 9** Hasil Program Opening

Hasil dilasi pada program mengakibatkan gambar yang sebelumnya terdapat teks pada huruf “T” setelah dilakukan proses opening garis horizontal pada huruf “T” menjadi hilang sebagian ini diakibatkan saat proses erosi garis tersebut habis sehingga saat dilakukan dilasi tidak dapat dilakukan.

**CLOSING**

Closing merupakan kebalikan dari opening. Dimana citra terlebih dahulu dilakukan dilasi yang kemudian dilanjutkan dengan erosi. Closing bertujuan untuk mengisi lubang kecil pada objek, menggabungkan objek yang berdekatan.

|  |
| --- |
|  |

**Gambar 10** Proses Closing

Berikut ini merupakan kode program untuk implementasi closing pada python menggunakan opencv.

|  |
| --- |
| import cv2  import numpy as np  from matplotlib import pyplot as plt  #membaca gambar  img = cv2.imread('contoh.png',0)  #konversi biner dengan threshold apabila minValue 127 dan MaxValue 255 pada sebuah pixel maka nilainya 1  ret, thresh = cv2.threshold(img, 127, 255, cv2.THRESH\_BINARY)  #membuat sebuah kernel 3x3 tipe 8bit  kernel = np.ones((3,3),np.uint8)  #proses opening dengan melakukan dilasi terlebih dahulu kemudian erosi  closing = cv2.morphologyEx(thresh,cv2.MORPH\_CLOSE, kernel, iterations = 2)  #membuat tampilan citra awal  plt.subplot(131),plt.imshow(img,cmap = 'gray')  plt.title('Citra Awal'), plt.xticks([]), plt.yticks([])  #membuat tampilan citra biner saat di threshold  plt.subplot(132),plt.imshow(thresh,cmap = 'gray')  plt.title('Citra Biner'), plt.xticks([]), plt.yticks([])  #membuat tampilan citra hasil closing  plt.subplot(133),plt.imshow(closing,cmap = 'gray')  plt.title('Hasil Closing'), plt.xticks([]), plt.yticks([])  #menampilkan gambar  plt.show() |

**Kode Program 5** Program Closing

Pada proses closing langkah pertama adalah membaca citra yang akan di proses kemudian dilakukan konversi gambar ke biner dimana threshold dari nilainya apabila nilai sebuah piksel dibawah 127 maka akan bernilai 0 apabila nilai piksel diatas 127 maka akan bernilai 1. Langkah selanjutnya adalah membuat kernel 3x3 bertipe 8bit dan melakukan proses erosi kemudian menampilkannya. Berikut ini merupakan hasil dari program closing.

|  |
| --- |
|  |

**Gambar 11** Hasil Program Closing

Hasil dilasi pada program mengakibatkan gambar yang sebelumnya terdapat teks dan garis setelah dilakukan proses closing teks “tulisan” pada huruf “L” menjadi menyatu dan garis hitam di sekitar tulisan hilang dan teks menjadi bersambung.

**GRADIENT**

Gradient digunakan dalam pengolahan citra untuk mendeteksi tepi. Asumsi yang umum adalah bahwa tepi objek terletak di mana terdapat variasi nilai biner yang tinggi. Gradient mengembalikan variasi maksimum intensitas biner di lingkungan yang ditentukan oleh elemen penataan. Berikut ini merupakan kode program untuk implementasi gradient pada python menggunakan opencv.

|  |
| --- |
| import cv2  import numpy as np  from matplotlib import pyplot as plt  #membaca gambar  img = cv2.imread('contoh.png',0)  #konversi biner dengan threshold apabila minValue 127 dan MaxValue 255 pada sebuah pixel maka nilainya 1  ret, thresh = cv2.threshold(img, 127, 255, cv2.THRESH\_BINARY)  #membuat sebuah kernel 3x3 tipe 8bit  kernel = np.ones((3,3),np.uint8)  #proses gradient adalah menentukan letak variasi nilai biner yang berbeda  gradient = cv2.morphologyEx(thresh,cv2.MORPH\_GRADIENT, kernel, iterations = 1)  #membuat tampilan citra awal  plt.subplot(131),plt.imshow(img,cmap = 'gray')  plt.title('Citra Awal'), plt.xticks([]), plt.yticks([])  #membuat tampilan citra biner saat di threshold  plt.subplot(132),plt.imshow(thresh,cmap = 'gray')  plt.title('Citra Biner'), plt.xticks([]), plt.yticks([])  #membuat tampilan citra hasil gradient  plt.subplot(133),plt.imshow(gradient,cmap = 'gray')  plt.title('Hasil Gradient'), plt.xticks([]), plt.yticks([])  #menampilkan gambar  plt.show() |

**Kode Program 6** Program Gradient

Pada proses gradient langkah pertama adalah membaca citra yang akan di proses kemudian dilakukan konversi gambar ke biner dimana threshold dari nilainya apabila nilai sebuah piksel dibawah 127 maka akan bernilai 0 apabila nilai piksel diatas 127 maka akan bernilai 1. Langkah selanjutnya adalah membuat kernel 3x3 bertipe 8bit dan melakukan proses gradient kemudian menampilkannya. Berikut ini merupakan hasil dari program gradient.

|  |
| --- |
|  |

**Gambar 12** Hasil Program Gradient

Hasil dilasi pada program mengakibatkan gambar yang sebelumnya terdapat teks dan garis setelah dilakukan gradient teks dan garis membentuk sebuah garis tepi (outline).

**LANGKAH-LANGKAH ALGORITMA PENGENALAN UKIRAN BALI**

Berikut ini merupakan langkah-langkah yang dilakukan dalam algoritma pengenalan ukiran bali.

1. Data Capture

Data capture merupakan proses konversi dimana objek lukisan bali yang nyata dijadikan suatu file gambar digital agar dapat di proses.

1. Preprocessing

Preprocessing merupakan suatu proses untuk menghilangkan bagian-bagian yang tidak diperlukan pada gambar agar citra Ukiran Bali dapat diambil dan dijadikan sampling.

1. Segmentasi

dilakukan bila citra yang akan dianalisis memiliki noise sehingga perlu dihaluskan dengan tapis citra. Perancangan tapis dengan memanipulasi piksel-piksel tetangga membuat citra lebih halus, bentuk sudut, dan tepi citra tetap terjaga. Pada proses perekaman citra digital dapat terjadi gangguan yang bersifat frekuensi rendah, dimana terjadi proses pemerataan intensitas cahaya pada suatu titik sampel dengan titik-titik tetangganya. Gangguan lain yang sering terjadi pada proses perekaman citra digital adalah terjadinya gangguan berbentuk garis-garis akibat adanya kerusakan pada sebagian detektor sensor. Juga sering dijumpai gangguan lain dalam bentuk bercak hitam yang acak.

1. Normalisasi

digunakan untuk mengubah citra dengan format keabuan yang mempunyai nilai lebih dari dua ke format citra biner yang hanya memiliki dua nilai (0 atau 1). Dalam hal ini titik dengan rentang nilai keabuan tertentu diubah menjadi warna hitam dan sisanya menjadi warna putih atau sebaliknya.

1. Pengenalan

Prinsip metode ini adalah membandingkan antara image objek yang akan dikenali dengan image template yang ada. Image objek yang akan dikenali mempunyai tingkat kemiripan sendiri terhadap masing-masing image template. Pengenalan dilakukan dengan melihat nilai tingkatkemiripan tertinggi dan nilai batas ambang pengenalan dari image objek tersebut. Bila nilai tingkat kemiripan berada di bawah nilai batas ambang maka image objek tersebut dikategorikan sebagai objek tidak dikenal.

http://abikindo.blogspot.co.id/2010/06/template-matching.html

1. Post Processing