**Implemnetasi Hough Transform Line untuk Mendeteksi Garis Menggunakan Python 2.7**

Diajukan untuk memenuhi tugas Matakuliah Visi Komputer yang diampu oleh Bapak Drs. Muh. Arif Rahman, M. Kom



**Disusun oleh:**

|  |  |
| --- | --- |
| Akhmad Muzanni Safi`i | 155150200111270 |
| Achmad Dewanto Aji W. | 155150207111167 |
| Ofi Eka Novyanti | 155150207111101 |
| Thio Marta Elisa Y.B.B | 145150200111015 |
| Risda Nur Ainum | 155150201111261 |

**PROGRAM STUDI TEKNIK INFORMATIKA**

**FAKULTAS ILMU KOMPUTER**

**UNIVERSITAS BRAWIJAYA**

**MALANG**

**2018**

# BAB I KAJIAN TEORI

## Hough Transform

*Hough Transform* (HT) didesain untuk menemukan garis pada suatu citra. HT diperkenalkan oleh Paul Hough pada tahun 1962. Hough Transform adalah sebuah metode yang dapat digunakan untuk mengisolasi feature tertentu dalam sebuah citra. Metode Hough Transform biasanya digunakan untuk mendeteksi bentuk geometri yang dapat dispesifikasikan dalam bentuk parametrik seperti garis, lingkaran, elips, dan lain–lain.

Metode yang digunakan dalam HT adalah membuat persamaan dari suatu piksel dan mempertimbangkan semua pasangan yang memenuhi persamaan tersebut. Semua pasangan ditempatkan pada suatu larik akumulator, yang disebut larik transformasi (Mc Andrew, 2004).

## Hough Transform Line

Teori dasar *Hough Transform* menggunakan bentuk parametrik dan menggunakan pemungutan suara terbanyak (*voting*) untuk menentukan nilai parameter yang tepat. Apabila dalam citra terdapat beberapa garis yang saling berpotongan pada suatu titik, maka apabila kemudian titik tersebut ditransformasi ke dalam ruang parameter ***m - c*** kemudian untuk mengetahui transformasi titik tersebut adalah sebuah garis lurus atau bukan dilakukan *voting* dengan persamaan garis yang dinyatakan dalam persamaan (1.1).

***yi = mxi + c*** (1.1)

Di dalam sebuah citra yang terdapat garis lurus maka garis tersebut mempunyai persamaan yang sama dengan persamaan (1.1), maka apabila ditransformasi ke dalam ruang parameter ***m - c*** akan diperoleh hasil beberapa garis yang saling berpotongan dalam suatu titik. Persamaan transformasi diperoleh dengan cara memanipulasi persamaan garisnya yaitu persamaan (1.1) menjadi bentuk parametrik menjadi persamaan (1.2).

***c = yi - mxi*** (1.2)

Misalnya dari persamaan garis ***4x + 3y = 12*** kita dapat menggambar grafiknya pada **koordinat Cartesius**, dengan cara:

* Langkah pertama adalah membuat table koordinat dan cukup dipilih dua pasangan koordinat yang mudah, yaitu titik potong terhadap sumbu x dan sumbu y.
* Jika diubah menjadi persamaan ***c* = *y* – *mx*** menjadi: ***4 = y – ( -4/3x)*** atau ***4***

## = y + 4/3x

* Titik potong terhadap sumbu x, y = 0

## y + 4/3x = 4

***0 + 4/3x = 4***

***4/3x = 4***

***x = 3,*** didapat titik ***(3, 0)***

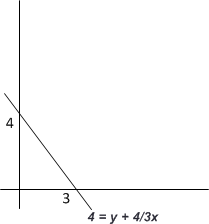
* Titik potong terhadap sumbu y, x = 0

## y + 4/3x = 4

***y + 0 = 4***

***y = 4,*** didapat titik ***(0, 4)***

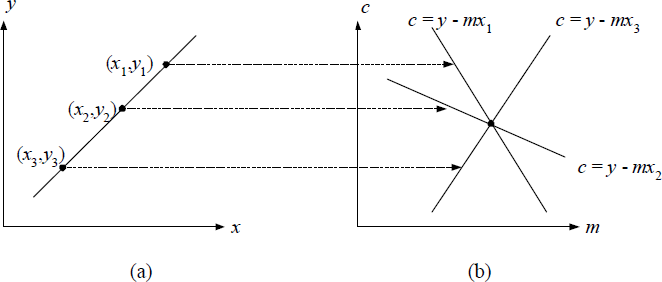
* Jadi hasil grafik kartesian persamaan ***4x + 3y = 12*** terlihat seperti gambar 1.



**Gambar 1.** Grafik kartesian persamaan *4x + 3y =12*

Permisalan lain, terdapat 3 buah titik pada sebuah garis lurus. Sembarang garis yang melalui titik (x1, y1) berkoresponden dengan garis c = y1 – mx1 pada ruang parameter m-c. Begitu juga, sembarang garis lurus yang melalui (x2, y2) berkoresponden dengan garis c = y2 – mx2 dan sembarang garis lurus yang melalui (x3, y3) berkoresponden dengan garis c = y3 – mx3 pada ruang m-c. Perpotongan (m’, c’) dari ketiga garis pada ruang m-c tersebut menentukan garis unik yang melalui (xi, yi), i = 1, 2, 3, di bidang X-Y.

Dengan cara ini, maka setiap pixel pada garis lurus di bidang citra berkoresponden dengan sejumlah garis lurus yang melalui satu ititik tertentu di ruang parameter m-c. Sifat ini dimanfaatkan untuk mendeteksi garis lurus. Jika setiap pixel tepi melakukan “pemungutan suara” pada ruang parameter, maka keberadaan garis lurus pada citra ditandai dengan penumpukan suara pada tempat- tempat tertentu di ruang parameter.\



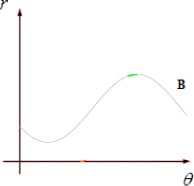
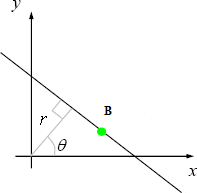
**Gambar 2.** (a) Garis lurus pada ruang X-Y; (b) representasinya dalam ruang parameter m-c.

Dalam kenyataannya, apabila ditemui sebuah garis vertikal, maka akan terjadi masalah dalam penghitungannya dikarenakan garis vertikal mempunyai nilai gradien kemiringan yang besarnya tak berhingga ( **∞ )**. Sebagai alternatifnyadigunakan persamaan (1.3).

***r = x cos θ + y sin θ*** (1.3)

Dimana gambar koordinat kartesiannya ditunjukkan pada Gambar 1. Sebuah titik B dalam gambar 1 apabila ditransformasi ke dalam ruang parameter ***r***

**- *θ*** maka akan menjadi seperti gambar 2.



**Gambar 3.** Citra sebuah garis dalam

koordinat Kartesian dengan satu titik uji

**Gambar 4.** Hasil transformasi dari satu titik

uji dalam ruang parameter ***r - θ***

Apabila di dalam citra terdapat suatu garis lurus, maka jika garis lurus citra ditransformasi kedalam ruang parameter ***r - θ*** akan terjadi suatu titik penumpukan antar kurva sinusoida hasil pentransformasian masing-masing komponen piksel garis yang membentuk garis lurus tersebut.

Metode proses yang digunakan dalam *Hough Transform Line* mendeteksi suatu garis pada objek berdasarkan algoritma *flowchart* pada gambar 5 dapat dijelaskan sebagai berikut :

1. Pembacaan berkas citra

Pembacaan berkas citra dilakukan untuk membaca citra asli yang menjadi source gambar suatu objek. Citra (*image*) itu sendiri memiliki arti secara harfiah yaitu sebagai gambar pada bidang dua dimensi (dwi matra). Ditinjau dari sudut pandang matematis, citra merupakan fungsi menerus (*continue*) dari intensitas cahaya pada bidang dua dimensi.

1. Pembacaan citra asli menjadi citra aras keabuan

Setelah dilakukan pembacaan citra, dilanjutkan dengan pengubahan citra menjadi citra aras keabuan. Supaya citra digital bisa diolah oleh komputer, maka citra digital harus mempunyai format tertentu. Format citra digital yang dipakai adalah citra skala keabuan (*Grayscale*). Format citra ini disebut skala keabuan karena pada umumnya warna yang dipakai warna hitam sebagai warna minimal (0) dan warna putih (255) sebagai warna maksimalnya, sehingga warna antaranya adalah abu-abu.



1. Deteksi tepi

**Gambar 9.** Citra *Gr*ay*scale*

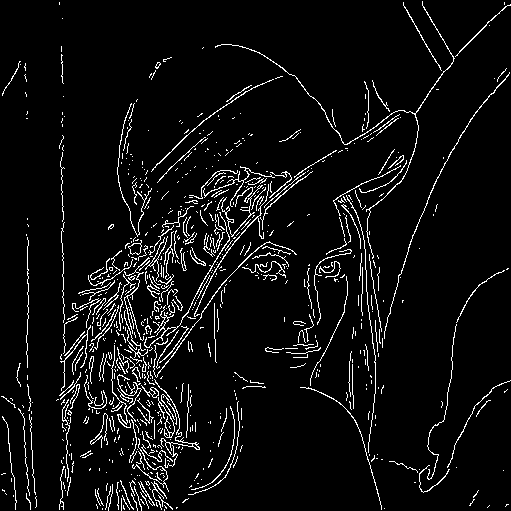
Deteksi tepi bertujuan untuk meningkatkan penampakan tepi pada citra. Suatu objek dapat dengan mudah dideteksi pada suatu citra jika objek cukup kontras dari latak belakangnya. Pada proses deteksi tepi ini hasilnya adalah berupa citra biner yang hanya memiliki dua nilai aras

keabuan saja yaitu hitam dan putih, dimana aras keabuan putih menunjukkan batas tepi. Metode deteksi tepi yang digunakan adalah metode *Canny*. Mengapa demikian ? Karena deteksi tepi *Canny* memiliki algoritma yang optimal dalam melakukan pendeteksian tepi. Untuk meningkatkan metode-metode yang telah ada dalam pendeteksian tepi, algortima deteksi tepi *Canny* mengikuti beberapa kriteria sebagai berikut:

* 1. Tingkat *error* yang rendah. *Error* terjadi bila ada tepi yang penting tetapi tidak muncul, atau bila ada yang bukan tepi tetapi muncul.
  2. Titik-titik pada tepi dilokasikan dengan benar. Dengan kata lain, jarak antara piksel- piksel tepi yang ditemukan algoritma dengan tepi sesungguhnya diminimumkan.
  3. Hanya satu respon (lebar 1 piksel) untuk setiap sebuah tepi.

Berdasarkan kriteria di atas, algoritma deteksi tepi *Canny* dilakukan dengan langkah- langkah sebagai berikut:

1. Pertama-tama dilakukan penghalusan (*smoothing*) citra untuk menghilangkan *noise*. Contohnya menggunakan *filtering* dengan *Gaussian Filter*.
2. Selanjutnya dicari *gradient magnitude* citra untuk melihat daerah-daerah yang memiliki turunan spasial yang tinggi. Pencarian *gradient magnitude* dapat menggunakan metode Sobel, Prewitt, dan lain lain.
3. Ditentukan arah dari tepi dengan menggunakan invers tangen dari *gradient magnitude* Y (Gy) dibagi *gradient magnitude* X (Gx). Arah yang diperoleh dari perhitungan ini kemudian dipetakan ke 0, 45, 90, atau 135 derajat berdasarkan kedekatannya dengan keempat derajat arah tadi.
4. Kemudian dilakukan *Non Maximum Suppression*. Yaitu, penghilangan nilai-nilai yang tidak maksimum. Ditelusuri daerah yang ditemukan pada langkah 2 (dengan arah seperti yang ditemukan pada langkah 3), dan menghilangkan (suppress) setiap piksel yang tidak maksimum.
5. Selanjutnya dilakukan *Hysteresis*. *Hysteresis* menggunakan dua *threshold* T1 (*threshold* bawah) dan T2 (*threshold* atas). Bila *magnitude* ada di bawah T1, titik tersebut di-set nol (dijadikan non-tepi). Bila *magnitude* ada di atas T2, maka termasuk tepi. Bila *magnitude* ada diantara T1 dan T2, di-*set* nol kecuali jika ada jalan (*path*) dari titik tersebut ke titik yang memiliki magnitude di atas T2.



**Gambar 10.** Citra Hasil Deteksi Tepi

1. Operasi *Hough Transform Line*

Pada proses operasi *hough transform* ini berfungsi untuk menampung citra dalam bentuk koordinat berdasarkan hasil deteksi tepi menggunakan metode *canny*. Sebelum menentukan garis lurus, citra yang telah melalui proses deteksi tepi akan terlebih dahulu di tampung di dalam variabel *storage* pada program.

1. Deteksi dan rekonstruksi garis lurus

Proses selanjutnya adalah deteksi dan rekonstruksi garis lurus. Deteksi garis lurus dilakukan dengan pembandingan antara besar tiap-tiap R yang ada terhadap besar R maksimun yang telah dikalikan dengan nilai ambang yang telah ditentukan yaitu sebesar 0,75. Besarnya R maksimum yang telah dikalikan dengan nilai ambang dijadikan sebagai batas minimal besar R yang harus dipenuhi sebagai syarat dideteksi dan direkonstruksi sebagai sebuah garis lurus.

Perekonstruksian garis lurus pada program ini dilakukan pada tiap-tiap titik penumpukan yang telah dideteksi sebagai garis lurus. Perekonstruksian garis lurus pada program ini dimulai dengan penghitungan dan penentuan letak titik asal terhadap titik tengah citra hasil pendeteksian tepi yang dijadikan sebagai titik referensi. Titik tengah citra adalah suatu titik yang berada persis di tengah matriks suatu citra. Penentuan letak titik asal terhadap titik tengah citra yang dijadikan sebagai titik referensi dijadikan sebagai alat bantu untuk mempermudah proses perekonstruksian selanjutnya. Setelah itu ditentukan batas titik *x* sesuai batas kolom matriks pada citra asli. Selanjutnya dihitung dan ditentukan batas titik *y* dengan mengacu pada letak titik asal dan batas titik *x*. Setelah itu direkonstruksi garis lurus dan titik asalnya sesuai dengan batas titik *x* dan batas titik *y* yang bersesuaian yang telah dihitung sebelumnya.

# BAB II SOURCECODE PROGRAM

1. **SOURCE CODE**

|  |  |
| --- | --- |
| houghline.py | |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52 | import cv2  import numpy as np  import matplotlib.pyplot as plt  img = cv2.imread('Monas.jpg')  gray = cv2.cvtColor(img,cv2.COLOR\_BGR2GRAY)  edges = cv2.Canny(gray,50,150,apertureSize = 3)  np.set\_printoptions(threshold='nan')  baris,kolom = edges.shape  print edges.shape  cv2.namedWindow("Monas.jpg",cv2.WINDOW\_NORMAL)  cv2.imshow('Monas.jpg',edges)  def hough\_line(img, akurasi\_pixel=1):  arraytheta = np.deg2rad(np.arange(-90.0, 90.0, akurasi\_pixel))  panjangdiagonal = int(np.round(np.hypot(baris, kolom)))  arrayrho = np.linspace (-panjangdiagonal, panjangdiagonal, panjangdiagonal \* 2)  panjang\_theta = len(arraytheta)  cos\_t = np.cos (arraytheta)  sin\_t = np.sin (arraytheta)  acumulator = np.zeros ((2 \* panjangdiagonal, panjang\_theta), dtype=np.uint8)  yindeks, xindeks = np.nonzero (img)  for i in range (len (xindeks)):  x = xindeks[i]  y = yindeks[i]  for t\_idx in range (panjang\_theta):  rho = panjangdiagonal + int(round (x \* cos\_t[t\_idx] + y \* sin\_t[t\_idx]))  acumulator[rho, t\_idx] += 1  return acumulator, arraytheta, arrayrho  def show\_hough\_line(accumulator, thetas, rhos):  plt.imshow(accumulator, cmap='jet',aspect='auto',  extent=[np.rad2deg(thetas[-1]), np.rad2deg(thetas[0]), rhos[-1], rhos[0]],)  plt.show()  a, t, r = hough\_line(edges)  hasil = []  b,k = a.shape  for i in range (b):  for j in range (k):  if a[i][j] > 15:  indexi =np.abs(np.round(r[i]))  indexj =(t[j])  hasil.append([(indexi, indexj)])  hasilarray = np.asarray(hasil)  for rho,theta in hasilarray[:,0,:]:  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))  cv2.line(img,(x1,y1),(x2,y2),(255,0,0),2)  cv2.imwrite('hasil.jpg',img)  print 'Selesai' |

1. **PENJELASAN**

|  |  |
| --- | --- |
| houghline.py | |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52 | Proses import library cv2  Proses import library numpy  Proses import library  Proses pembacaan file gambar dengan nama Monas.jpg  Mengubah variabel gambar img menjadi grayscale dan disimpan pada variabel gray  Proses deteksi tepi variabel gambar gray dengan algoritma Canny dan disimpan pada variabel edges  Proses pengaturan print dengan threshold NaN  Proses menyimpan ukuran gambar dan disimpan pada variabel baris dan kolom  Proses mencetak ukuran gambar  Proses menampilkan window dengan nama Monas.jpg berukuran normal  Proses menampilkan variabel gambar edges dengan nama Monas.jpg  Deklarasi method houghline dengan parameter img dan akurasi 1  Proses membuat range theta dari -90 sampai 90 dengan selisih tertentu dan dikonversi ke dalam satuan radian  Proses menyimpan panjang diagonal gambar  Proses membuat range dimensi rho dengan panjang 2 kali diagonal gambar  Proses menyimpan panjang range theta dan disimpan pada variabel panjang\_theta  Proses menghitung menyimpan nilai cos theta  Proses menghitung menyimpan nilai sin theta  Proses pembuatan array akumulator dengan dimensi rho 2 kali panjang diagonal dan dimensi theta sebesar panjang\_theta  Proses menyimpan yindeks dan xindeks  Inisialisasi perulangan dengan range xindeks  Proses menyimpan xindeks indeks ke i ke dalam variabel x  Proses menyimpan yindeks indeks ke i ke dalam variabel y  Inisialisasi perulangan dengan range panjang\_theta  Proses penghitungan rho dengan rumus x\*cos(theta) + y\*sin(theta)  Proses increment pada akumulator indeks ke rho, t\_idx  Proses pengembalian nilai acumulator, arraytheta dan arrayrho  Deklarasi method show\_hough\_line untuk menampilkan grafik houghline dengan parameter accumulator, thetas dan rhos  Proses menampilkan array accumulator dalam bentuk grafik  Proses eksekusi tampilan grafik accumulator  Proses pemanggilan method hough\_line dengan argumen gambar edges dan disimpan pada variabel a, t dan r  Deklarasi variabel list dengan nama hasil  Proses menyimpan ukuran array accumulator dalam variabel b dan k  Inisialisasi perulangan dengan range sepanjang variabel b  Inisialisasi perulangan dengan range sepanjang variabel k  Inisialisasi percabangan dengan syarat nilai accumulator > 15  Proses menyimpan nilai mutlak accumulator dalam variael indexi  Proses menyimpan nilai theta indeks ke j dalam variabel indexj  Proses penambahan elemen array hasil dengan tuple berisi variabel indexi dan indexj  Proses sorting variabel hasil dan disimpan pada variabel hasilarray  Inisialisasi perulangan untuk hasil rho dan theta yang ditemukan dengan metode houghline  Proses menghitung dan menyimpan nilai cos(theta) dalam variabel a  Proses menghitung dan menyimpan nilai sin(theta) dalam variabel b  Proses menghitung a\*rho dan disimpan pada variabel x0  Proses menghitung b\*rho dan disimpan pada variabel y0  Proses menghitung x1  Proses menghitung y1  Proses menghitung x2  Proses menghitung y2  Proses membuat garis pada image dengan titik awal (x1,y1) dan titik akhir (x2,y2) dengan warna biru dan tebal 2 pt  Proses menyimpan variabel gambar menjadi file gambar dengan nama hasil.jpg  Proses mencetak keterangan Selesai |

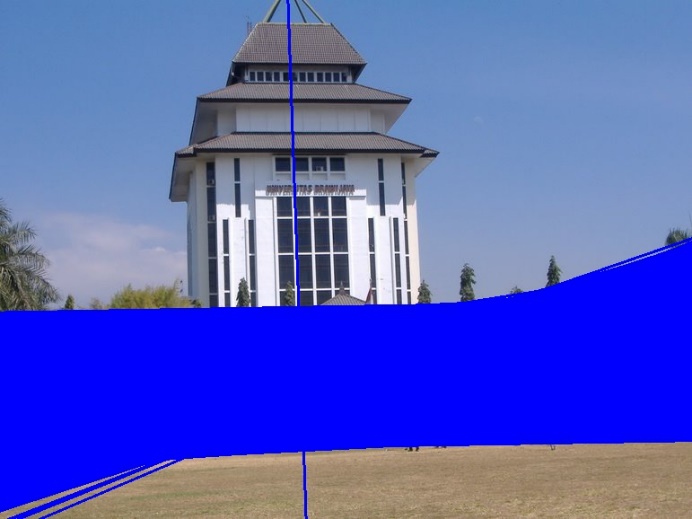
# BAB III ANALISIS PROGRAM



Output method myHoughLine untuk Gambar 1



Output method myHoughLine untuk Gambar 2



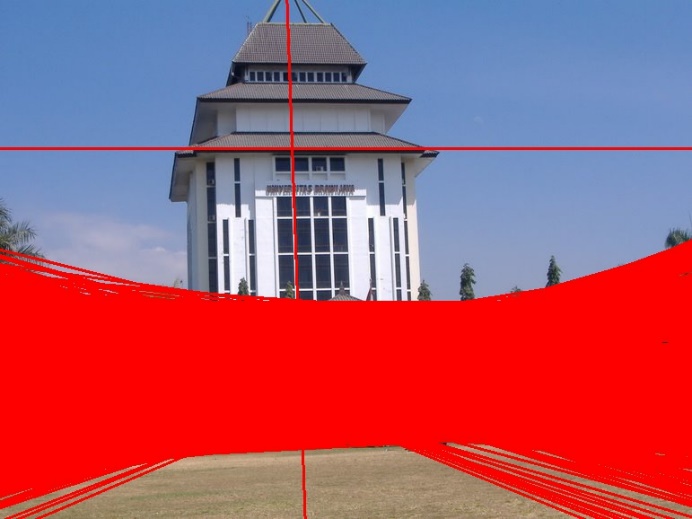
Output method myHoughLine untuk Gambar 3



Output method HoughLines dari library untuk Gambar 1



Output method HoughLines dari library untuk Gambar 3



Output method HoughLines dari library untuk Gambar 3

Dalam pengimplementasian method myHoughline masih terdapat beberapa kekurangan. Hal ini disebabkan beberapa hal, antara lain :

1. Komputasi dan pemilihan variabel yang kurang optimal sehingga terdapat perbedaan hasil rho dan theta yang dihasilkan
2. Algoritma houghline yang belum optimal sehingga mempengaruhi waktu eksekusi program
3. Pemilihan ukuran dimensi grafik houghline yang belum optimal sehingga menambah beban memori saat eksekusi.

# BAB IV DAFTAR RUJUKAN

[Gusti Bimo](http://bimomarlaw.github.io/). (2015, 24 November). [Tranformasi Hough untuk deteksi garis lurus dan lingkaran](http://bimomarlaw.github.io/Praktikum%20PPCD%209/). Diperoleh 16 Maret 2018, dari <http://bimomarlaw.github.io/Praktikum%20PPCD%209/>

Widi Hapsari. 2015. *Transformasi Hough Linear Untu Analisi dan Pengenalan Batif Motif Paraf*

,Jakarta: Jurnal IPTEK. Vol. 11,No. 2, November 2015.

Syahri Muharom. 2017. *Penerapan Metode Hough Line Transform Untuk Mendeteksi Pintu Ruangan Menggunakan Kamera,* Surabaya: Jurnal IPTEK. Vol. 21,No. 1, Mei 2017.