21 - Kontur

IF4073 Interpretasi dan Pengolahan Citra

Oleh: Rinaldi Munir

Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung
2021

Kontur

- Pendeteksi tepi menghasilkan citra tepi yang berupa citra biner
- *Pixel-pixel* tepi berwarna putih, sedangkan *pixel* bukan-tepi berwarna hitam.

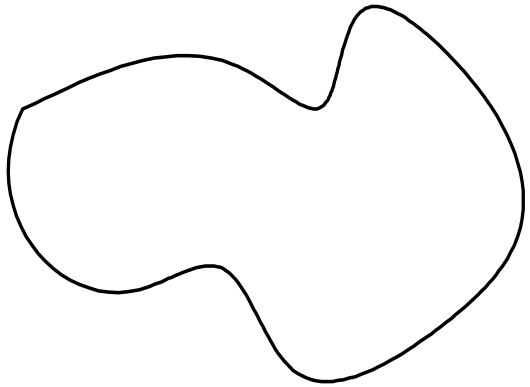
- Tetapi, tepi belum memberikan informasi yang berguna karena belum ada keterkaitan antara suatu tepi dengan tepi lainnya.

- Oleh karena itu, citra tepi ini harus diproses lebih lanjut untuk menghasilkan informasi yang lebih berguna yang dapat digunakan dalam mendeteksi bentuk-bentuk sederhana.

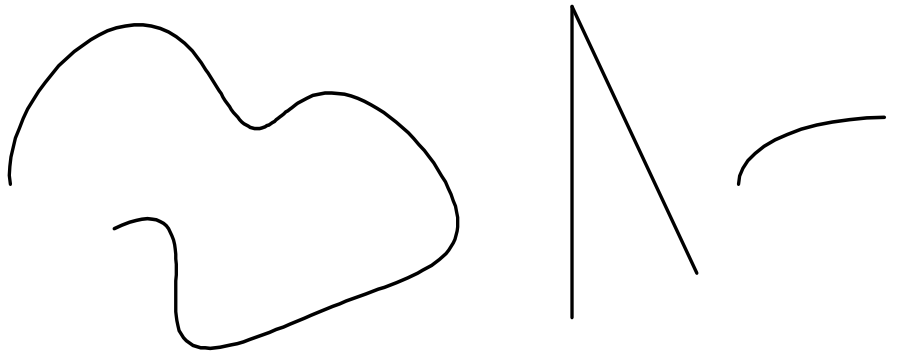
- Bentuk-bentuk sederhana misalnya garis lurus, lingkaran, elips, dan sebagainya. Bentuk-bentuk ini berguna di dalam proses analisis citra.

- Rangkaian *pixel-pixel* tepi yang membentuk batas daerah (*region boundary*) disebut **kontur**

- Kontur dapat terbuka atau tertutup.



(a) Kontur tertutup



(b) Kontur terbuka

- Kontur tertutup berkoresponden dengan batas yang mengelililingi suatu daerah (*region*).

- *Pixel-pixel* di dalam daerah tertutup dapat ditemukan dengan algoritma pengisian (*filling algorithm*).

- Batas daerah berguna untuk mendeskripsikan bentuk objek dalam tahap analisis citra (misalnya untuk mengenali objek).

- Kontur terbuka dapat berupa fragmen garis atau bagian dari batas daerah yang tidak membentuk sirkuit

Representasi Kontur

- Representasi kontur dapat berupa senarai tepi (*edge list*) atau berupa kurva.

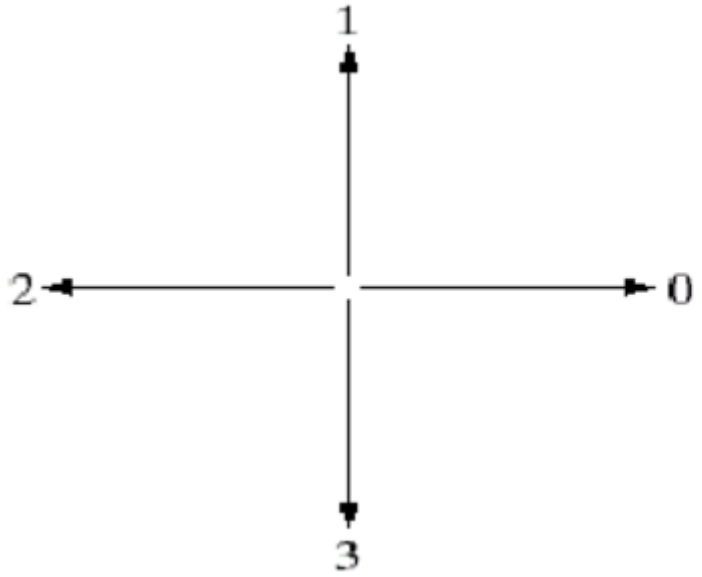
- Senarai tepi merupakan himpunan terurut *pixel-pixel* tepi.

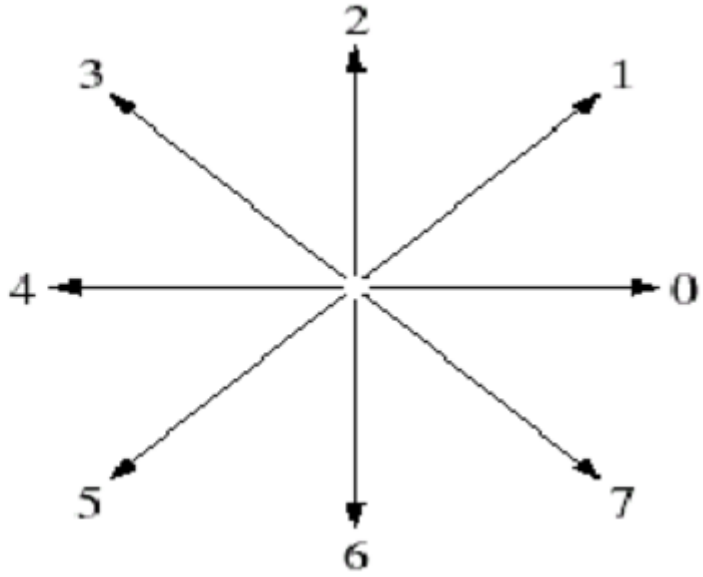
- Representasi kontur ke dalam kurva merupakan representasi dalam bentuk persamaan. Misalnya, rangkaian *pixel* tepi yang membentuk garis dapat direpresentasikan hanya dengan sebuah persamaan garis lurus.

- Representasi semacam ini menyederhanakan perhitungan selanjutnya seperti arah dan panjang garis

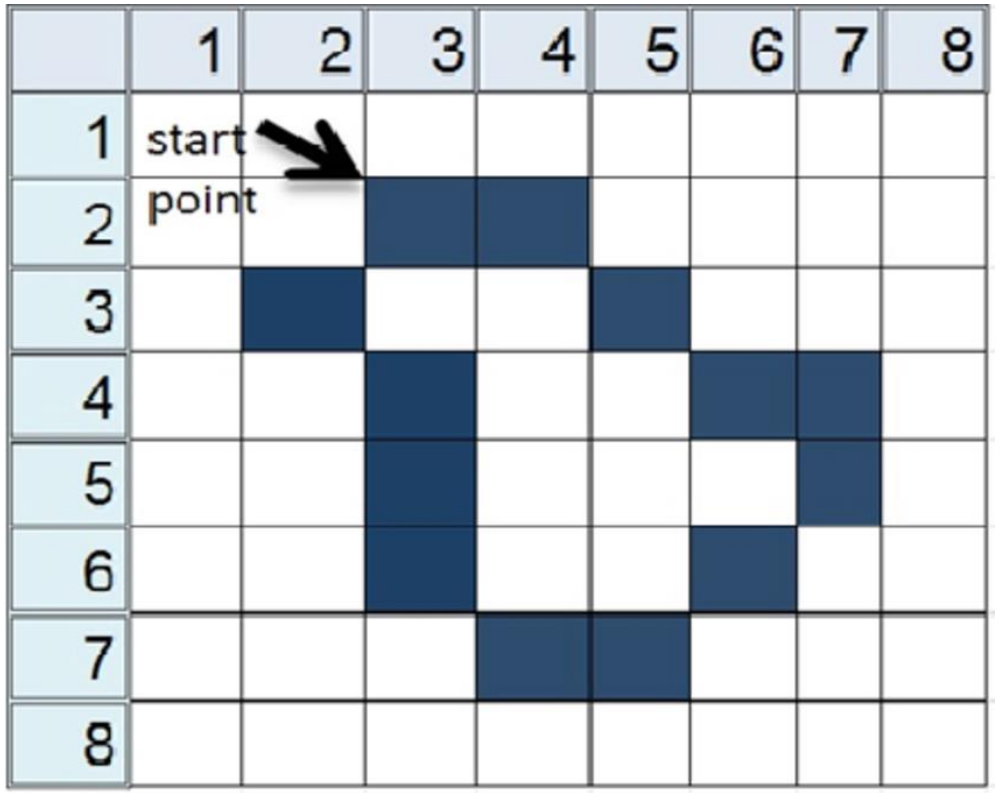
Kode Rantai

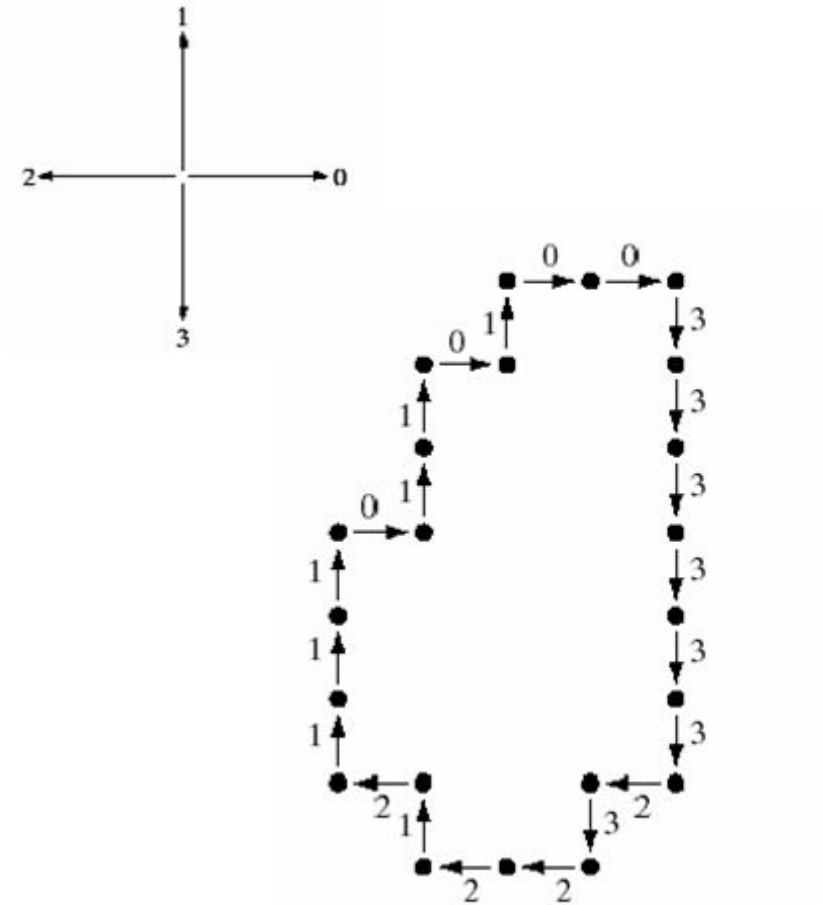
- **Kode rantai** (*chain code*)adalah notasi untuk mengkodekan senarai tepi yang membentuk batas daerah.
- Kode rantai menspesifikasikan arah setiap *pixel* tepi di dalam senarai tepi. Arah yang digunakan adalah 4 arah mata angin atau 8 arah mata angin.



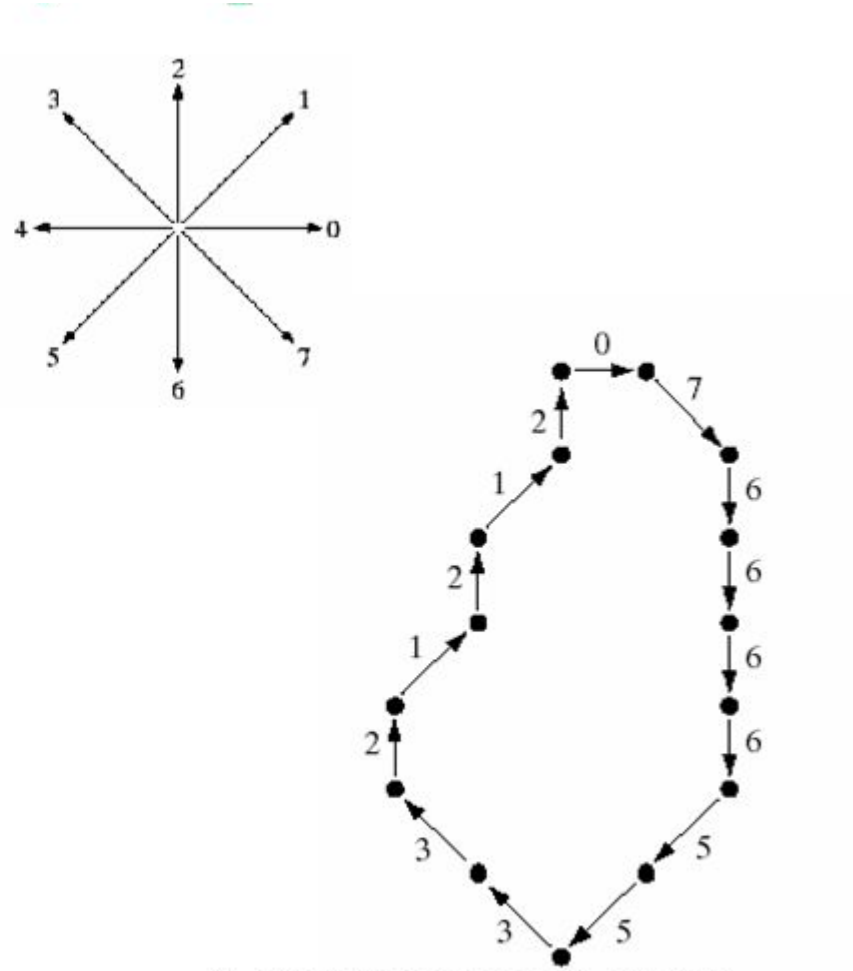


- Dimulai dari sebuah *pixel* tepi dan searah jarum jam, arah setiap *pixel* tepi yang membentuk batas objek dikodekan dengan salah satu dari empat atau delapan kode rantai.
- Kode rantai merepresentasikan batas objek dengan koordinat *pixel* tepi pertama lalu diikuti dengan senarai kode rantai



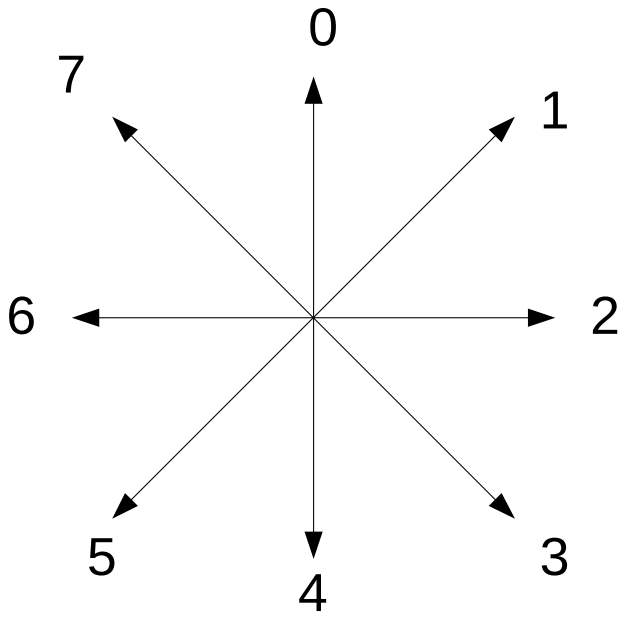


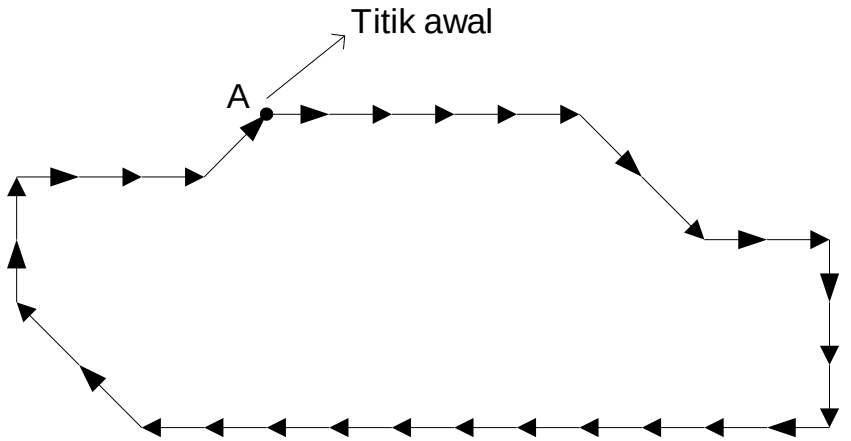
4-directional chain code:
00333333232212111101101

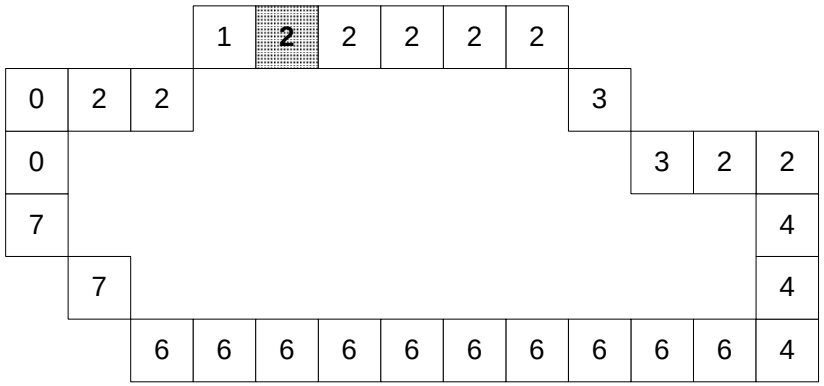


8-directional chain code:
076666553321212

- Aturan penomoran kode rantai tergantung konvensi yang digunakan.







Kode rantai: 22222332244466666666667700221

Transformasi Hough

- Kurva yang merepresentasikan kontur dicari dengan teknik pencocokan kurva (*curve fitting*).
- Ada dua macam teknik pencocakan kurva: interpolasi dan **penghampiran** (*approximation*).
- Interpolasi kurva adalah mencari kurva yang melalui **semua** pixel tepi.
- Penghampiran kurva dengan mencari kurva yang paling dekat melalui pixel-pixel tepi, tetapi tidak perlu melalui semua *pixel* tersebut.
- Transformasi Hough adalah teknik untuk menghampiri kurva parameterik (garis, lingkaran, elips, dll).

- Transformasi Hough menspesifikasikan kurva dalam bentuk parametrik. Kurva dinyatakan sebagai bentuk parametrik

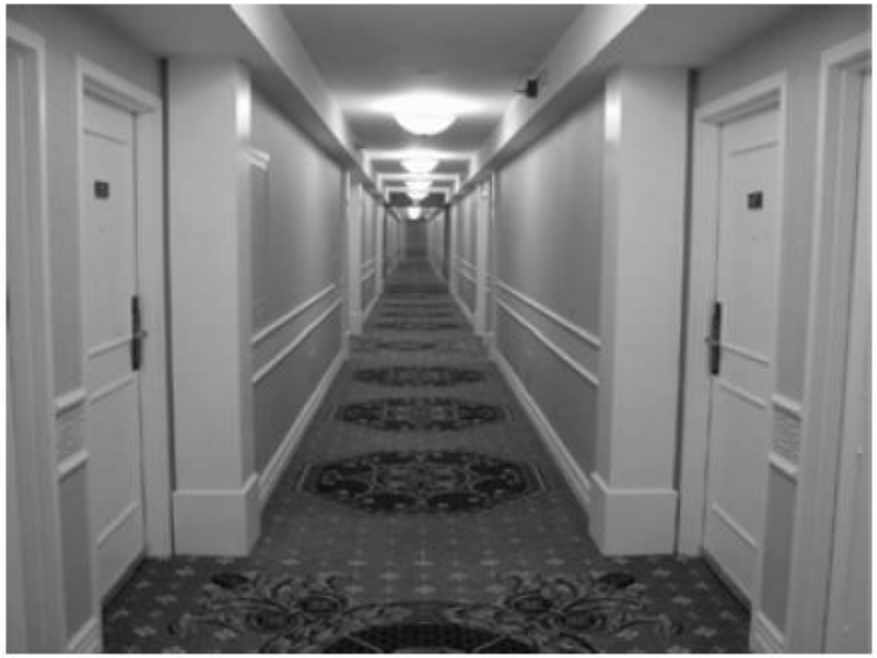
$$(x(u), y(u))$$

dari parameter u.

- Transformasi Hough menggunakan mekanisme *voting* untuk mengestimasi nilai parameter.

- Setiap titik di kurva menyumbang suara untuk beberapa kombinasi parameter. Parameter yang memperoleh suara terbanyak terpilih sebagai pemenang

A. Mendeteksi garis lurus



Contoh citra dengan struktur linier

Citra hasil deteksi tepi

Sumber: INF 4300 – Hough transform, Anne Solberg

- Persamaan garis lurus:

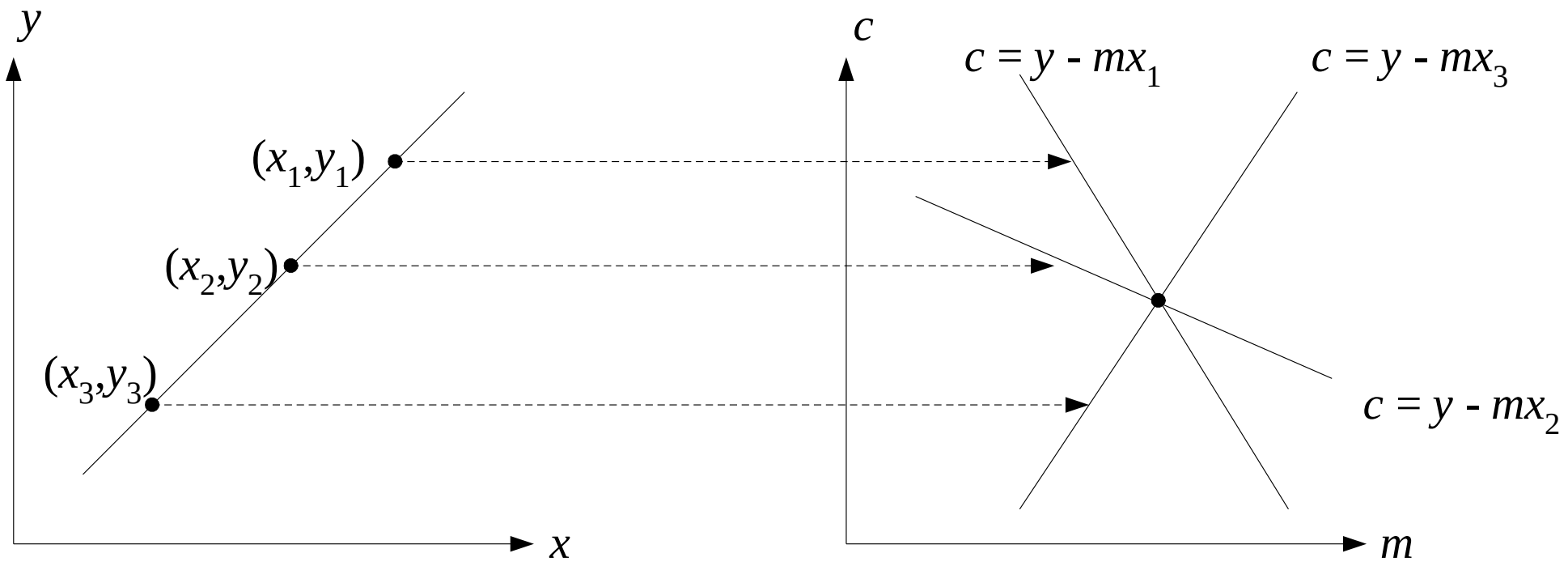
$$y = mx + c \qquad (1)$$

- Dalam bentuk parametrik, setiap garis dinyatakan di dalam ruang parameter *m*-*c*. Persamaan garis lurus dalam ruang m-c ditulis menjadi

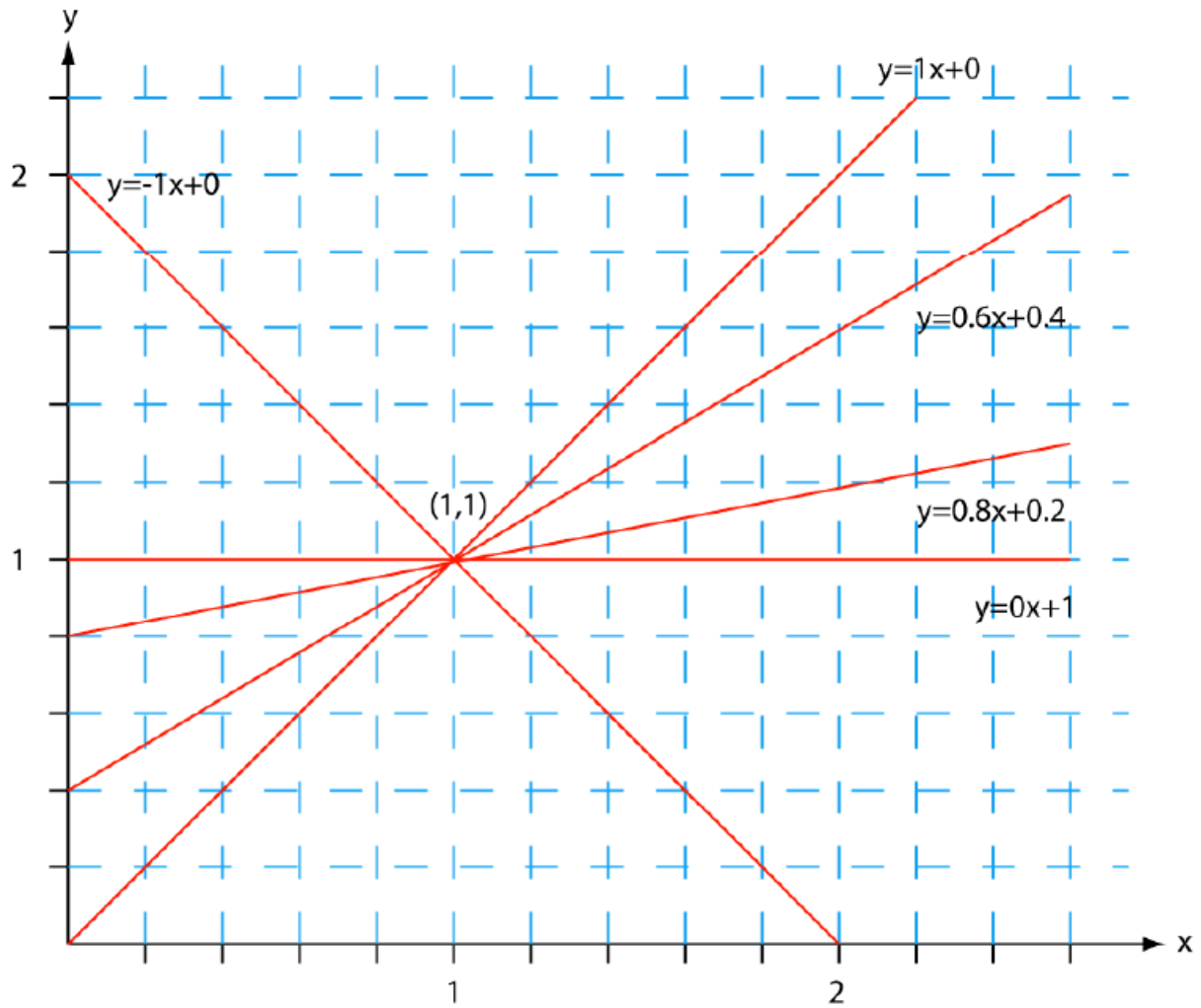
$$c = y - mx \qquad (2)$$

- Sembarang titik (*x*,*y*) pada bidang planar *X*-*Y* berkoresponden dengan sebuah garis lurus pada ruang parameter *m*–*c*.

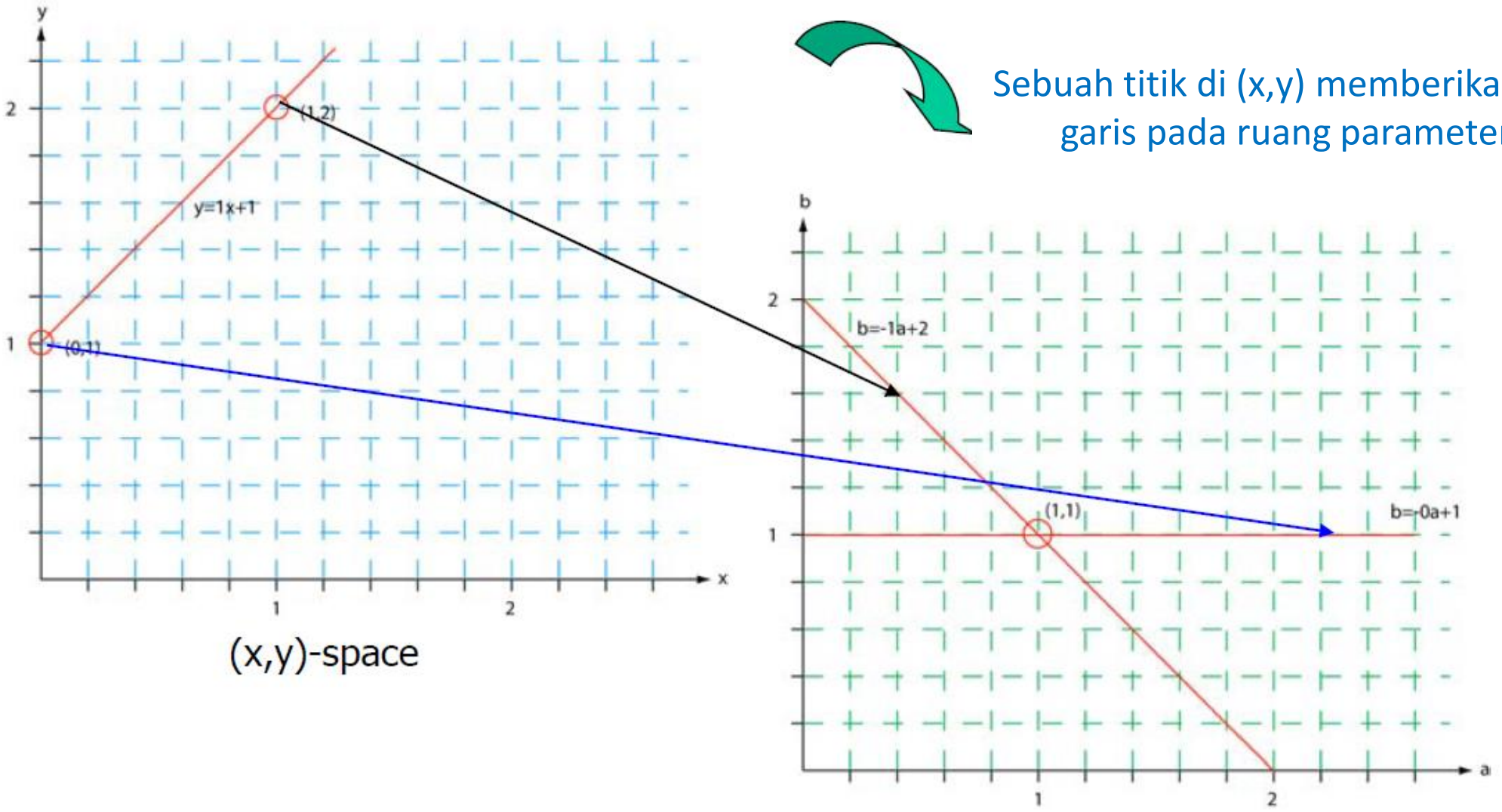
- Setiap *pixel* pada garis lurus di bidang citra berkoresponden dengan sejumlah garis lurus yang melalui **satu** titik tertentu di ruang parameter *m*-*c*.



- 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.

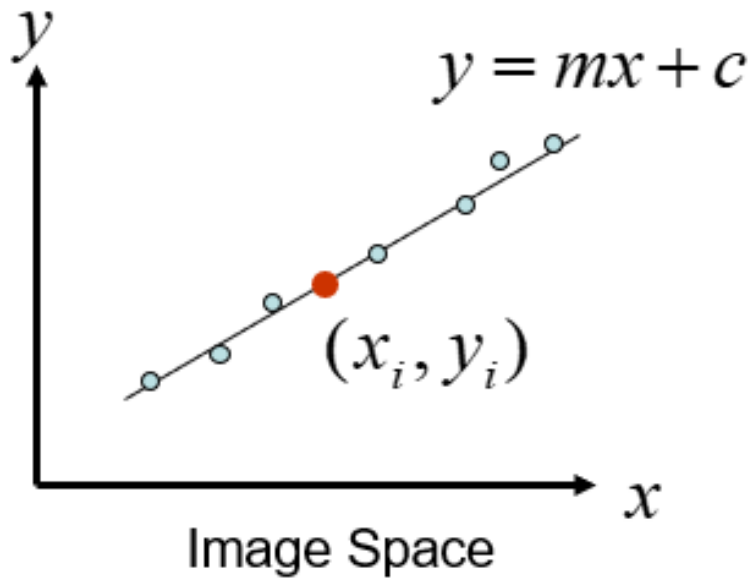


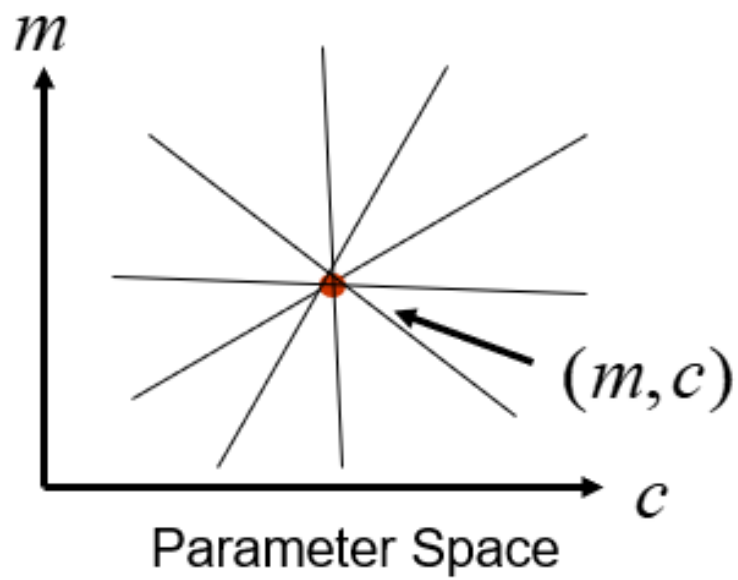
Sumber: INF 4300 – Hough transform, Anne Solberg



Sebuah titik di (x,y) memberikan sebuah garis pada ruang parameter m-c

Sumber: INF 4300 – Hough transform, Anne Solberg





Sumber: *Computer Vision*, S. Narasimhan

Algoritma Transformasi Hough untuk mendeteksi garis lurus:

1. Ruang parameter didiskritkan sebagai matriks $P(m, c)$, yang dalam hal ini $m_{\min} \leq m \leq m_{max}$ dan $c_{\min} \leq c \leq c_{max}$.

2. Tiap elemen pada ruang parameter diasumsikan sebagai akumulator. Inisialisasi setiap elemen $P(m, c)$ dengan 0.

3. Untuk setiap *pixel* tepi (x_i, y_i) hitung nilai $c = y_i - mx_i$. Untuk setiap nilai parameter m yang berkoresponden dengan nilai c, maka elemen matriks $P(m, c)$ yang bersesuaian dinaikkan satu:

$$P(m, c) = P(m, c) + 1$$

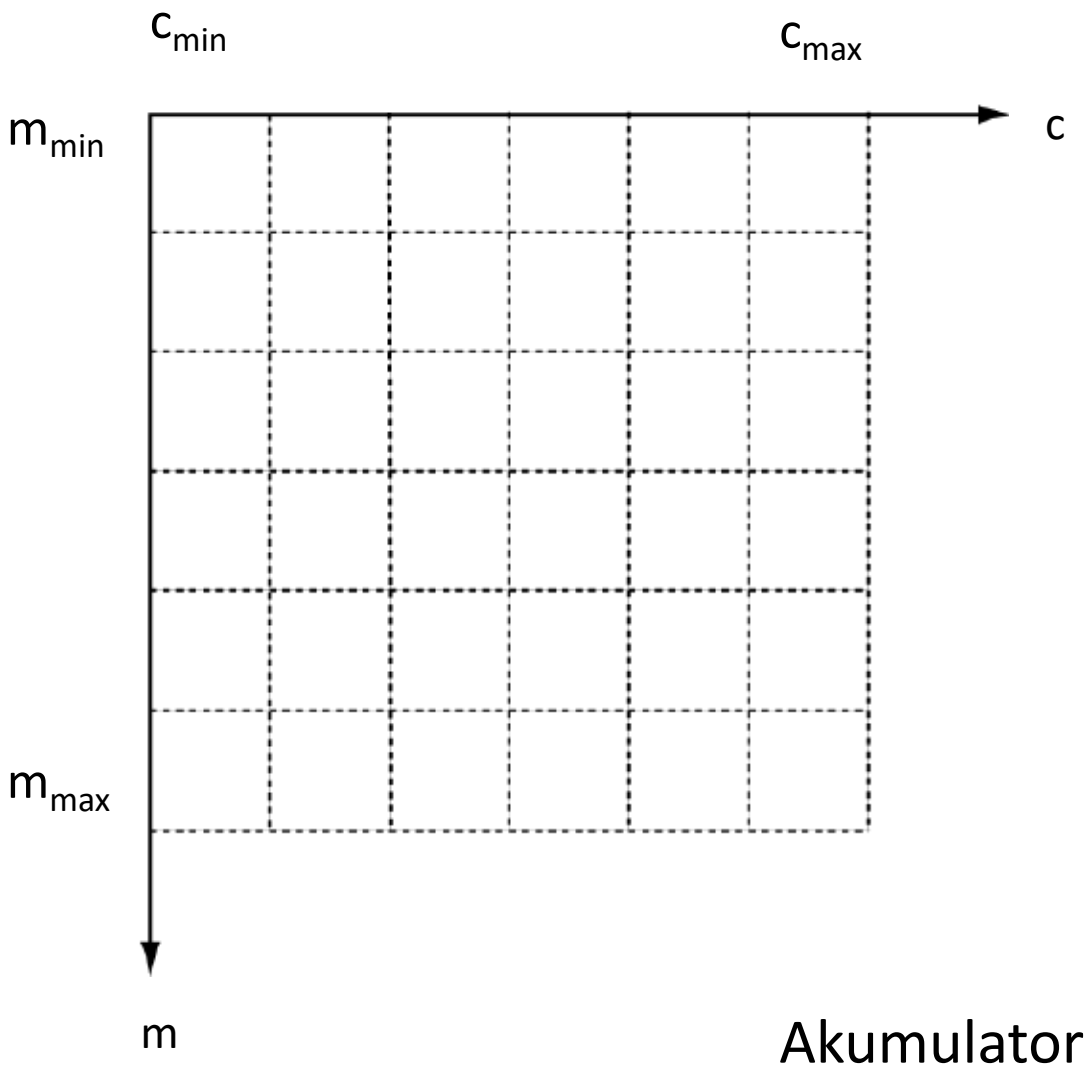
Dengan kata lain, tambahkan satu suara pada ruang parameter m-c.

4. Ulangi langkah 3 sampai seluruh *pixel* di dalam citra tepi ditelusuri.

5. Pada akhir prosedur, tiap elemen matriks $P(m, c)$ menyatakan jumlah *pixel* tepi yang memenuhi persamaan (1). Tentukan elemen matriks yang memiliki penumpukan suara cukup besar (yang nilainya di atas nilai ambang tertentu). Misalkan tempat-tempat itu adalah

   $\{(m_1, c_1), (m_2, c_2), \ldots, \{(m_k, c_k),$

Hal ini berarti terdapat *k* garis lurus yang terdeteksi pada citra.



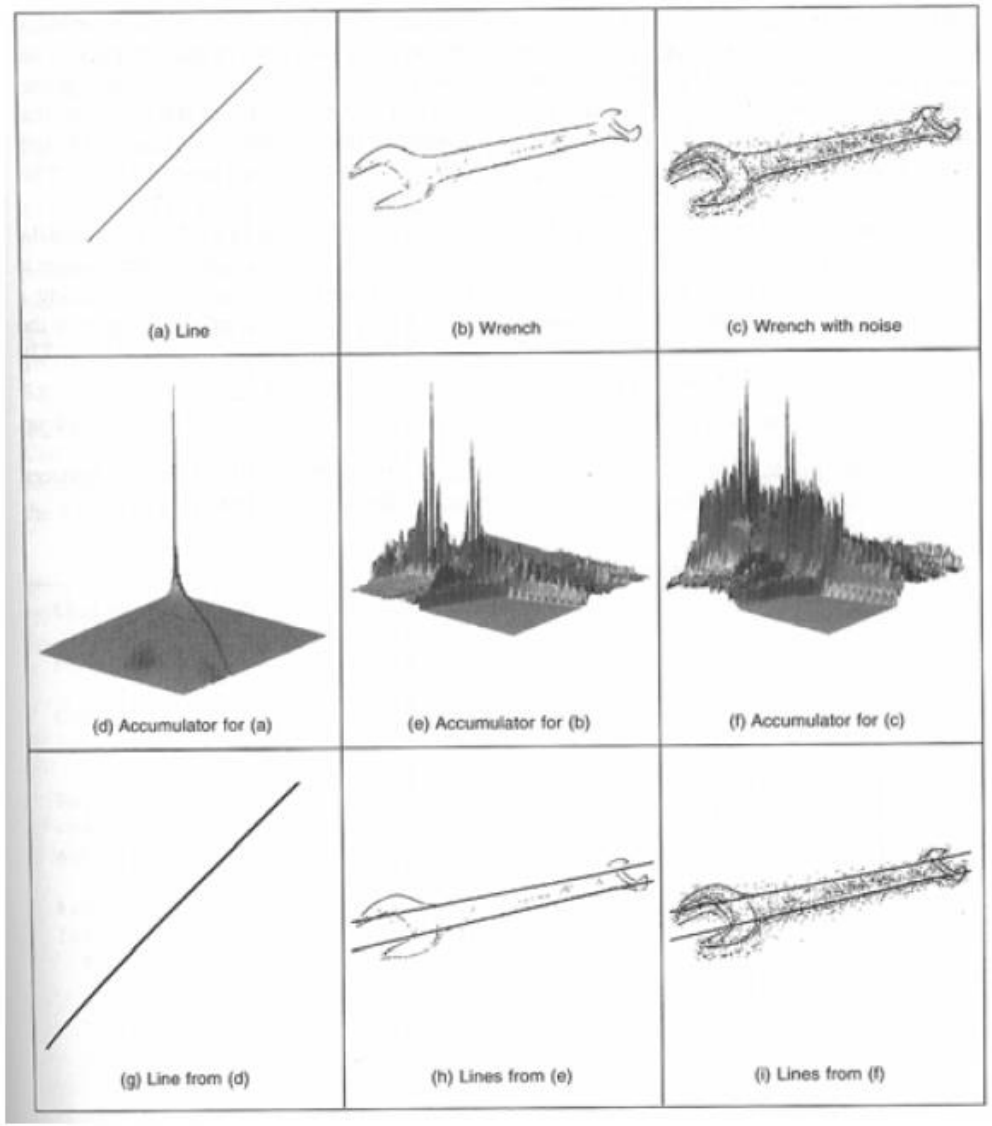
Akumulator Hough, P(m, c)

Transformasi Hough dapat mendeteksi garis meskipun citra mengandung derau sekalipun

Thresholded edge images

Visualizing the accumulator space
The height of the peak will be defined by the number of pixels in the line.

Thresholding the accumulator space and superimposing this onto the edge image



Note how noise effects the accumulator. Still with noise, the largest peaks correspond to the major lines.

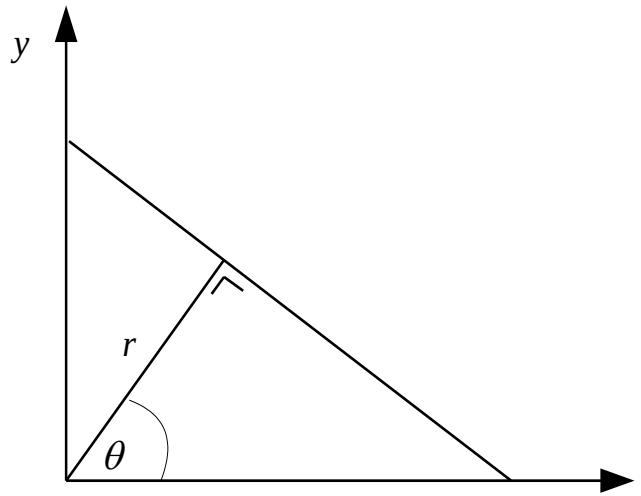
Sumber: INF 4300 – Hough transform, Anne Solberg

- Model parametrik pada persamaan (2) tidak dapat digunakan untuk mendeteksi garis vertikal karena gradiennya (*m*) nilai tak-berhingga.
- Selain itu, kebutuhan memori untuk akumulatornya besar, karena $-\infty \leq m \leq \infty$

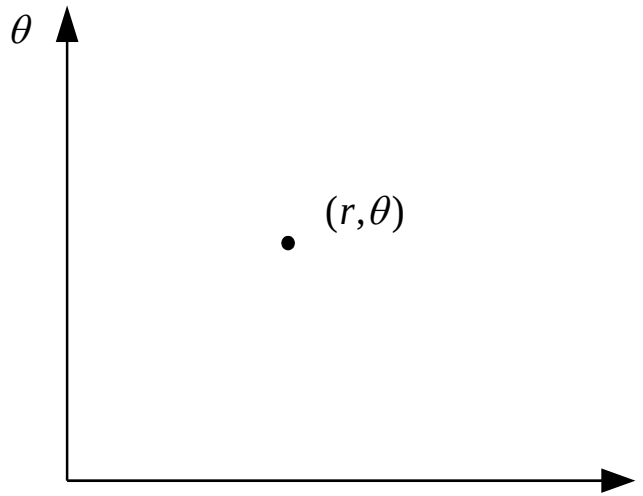
- Oleh karena itu, garis dinyatakan dalam representasi polar:

$$r = x \cos \theta + y \sin \theta \qquad (3)$$

yang dalam hal ini *r* adalah jarak garis ke titik asal



Persamaan polar



Ruang parametrik r - θ

- Sembarang garis yang melalui (x_1,y_1) pada ruang x–y berkoresponden dengan kurva sinusoida $r = x_1 \cos \theta + y_1 \sin \theta$ pada ruang r–θ.

- *Pixel-pixel* yang terletak segaris pada citra tepi berkoresponden dengan titik potong seluruh kurva sinusoidanya pada ruang parameter r–θ.

- Prosedur yang sama untuk mendeteksi garis lurus dapat digunakan kembali dengan mengganti ruang parameter m–c menjadi ruang parameter r–θ, yang dalam hal ini,

$$-\sqrt{N^2 + M^2} \leq r \leq \sqrt{N^2 + M^2}$$

$$-\pi/2 \leq \theta \leq \pi/2$$

Kode program Hough Transform

Input: citra tepi (citra hasil pendeteksian tepi), disimpan di dalam matriks `Edge[0..N-1,0..M-1]`. Ruang parameter r–θ dinyatakan sebagai matriks `P` yang berukuran $n \times m$. Nilai *cosinus* dan *sinus* disimpan di dalam *lookup table* `COS[0..p-1]` dan `SIN[0..p-1]`

```
void Hough(citra Edge, int N, int M, matriks P, int n, int m,
           float *COS, float *SIN)
/* prosedur yang melakukan Transformasi Hough.
   Masukan: citra tepi Edge yang berukuran  N x M.
   Keluaran: matriks parameter P yang berukuran n x m
*/
{
  int k, l, i, j, kk, ll;
  float r, b;
  float SQRTD =sqrt((float)N*(float)N + (float)M*(float)M);

  /* inisialisasi P[0..p-1, 0,,q-1] dengan 0 */
  for(kk=0;kk<=p-1;kk++)
    for(ll=0;ll<=q-1;ll++)
       P[kk][ll]=0;
```

```
  /*telusuri citra tepi. Jika pixel merupakan tepi, lakukan pemungutan
    suara pada elemen matriks P yang bersesuaian.
    tetha dari -pi/2 sampai pi/2.
    r dari -sqrt(N*N+M*M) sampai sqrt(N*N+M*M).
  */

  for (k=0;k<=N-1;k++)
    for (l=0;l<=M-1;l++)
    {
      if (Edge[k][l]==1)
      {
        for (i=0;i<=p-1;i++)
         {
           r = k*COS[i] + l*SIN[i];
           b = SQRTD;
           r+=b; r/=(SQRTD*2.0); r*=(m-1); r+=0.5;
           j=floor(r);
           P[i][j]++;
         }
      }
    }
}
```

Nilai *cosinus* dan *sinus* disimpan di dalam *lookup table* COS[0..p-1] dan SIN[0..p-1] yang dibentuk dengan kode program berikut:

```
void LookUpTable(float *COS, *SIN, int m)
/* Membuat tabel cosinus dan sinus untuk fungsi COS dan SIN.
   Masukan: m adalah jumlah baris tabel
   Keluaran: tabel COS dan tabel SIN
*/
{
 int i;
 float th, R_TO_D = 0.017453

 for(i=0;i<=p-1;i++)
 {
   th = (float)i * 180.0/(m-1)-90.0;
   th = th * R_TO_D;
   COS[i] = (double) cos((double)th);
   SIN[i] = (double) sin((double)th);
 }
```

- Setelah Transformasi Hough selesai dilakukan, langkah berikutnya adalah melakukan operasi pengambangan (*thresholding*) untuk menentukan tempat-tempat pada ruang paramater r–θ yang mempunyai penumpukan suara lebih besar dari nilai ambang T.

- Elemen matriks P yang nilainya di atas nilai ambang tersebut menyatakan parameter garis lurus.

- Misalkan tempat-tempat itu adalah $\{(m_1, c_1), (m_2, c_2), \ldots, \{(m_k, c_k)$, hal ini berarti terdapat k garis lurus yang terdeteksi pada citra.

```
void threshold(imatriks P, int n, in m, int T)
/* Melakukan pengambangan pada matriks parameter P.
   Setiap elemen matriks P yang nilainya di atas T menyatakan
    parameter garis lurus.
   Masukan: matriks parameter P yang berukuran n x m.
   Keluaran: matriks parameter P yang sudah di-threshold.
*/
{ int i, j;
 for(i=0;i<n;i++)
   for(j=0;j<m;j++)
      if (P[i][j]>T)
        P[i][j]=1;
      else
        P[i][j=0;
}
```

Transformasi Hough Balikan

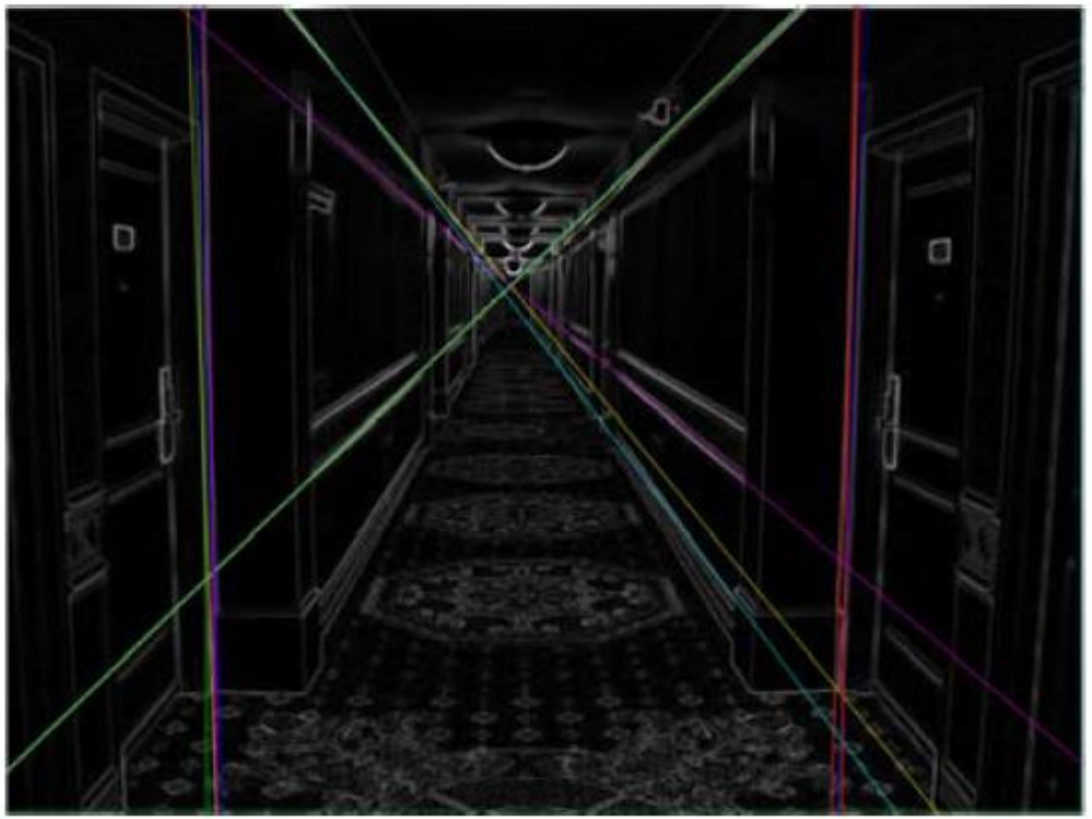
- *Pixel-pixel* tepi yang termasuk di dalam garis lurus hasil deteksi Transformasi Hough dapat dihasilkan dengan algoritma **Transformasi Hough Balikan** (*inverse Hough Transform*).

- Untuk setiap elemen matriks P yang bernilai 1, garis lurus yang bersesuaian (yang intensitasnya 1) digambarkan pada matriks luaran `Out`. Operasi `and` dengan citra tepi dilakukan untuk mengklarifikasi keberadaan *pixel* tepi.

```
void InverseHough(citra Edge, citra Out, int N, int M, imatriks P,
                  int n, int m, float *COS, float *SIN)
/* prosedur yang melakukan Transformasi Hough Balikan
   Masukan: 1. citra tepi Edge yang berukuran  N x M.
            2. matriks parameter P yang berukuran n x m
   Keluaran: citra Out yang berisi pixel-pixel pembentuk garis lurus.
*/
{
  int k, l, i, j;
  float r, y;
  float SQRTD =sqrt((float)N*(float)N + (float)M*(float)M);

  /* inisialisasi citra keluaran dengan 0 */
  for(kk=0;kk<=p-1;kk++)
    for(ll=0;ll<=q-1;ll++)
       Out[p][q]=0;

  /* Matriks parameter P telah dilakukan operasi thresholding.
     Untuk setiap elemen P yang bernilai 1, garis lurus yang
     bersesuaian digambarkan ke dalam matriks Out.
     and dengan citra tepi dikerjakan.   */
```

```
for (k=0;k<=p-1;k++)
       for (l=0;l<=q-1;l++)
        {
         y=(float)0.0;
         if (P[k][l]==1)  /* atau P[k][l]== 255 */
          {
           for (i=0;i<=N-1;i++)
            {
              r = (float)l * 2.0 * SQRTD/(m-1) - SQRTD;
              if (SIN[k]==(float)0.0)
                y++;
              else
                y=(r - (float)i * COS[k])/SIN[k];

              y+=0.5; j=floor(y);
              if (j >=0 && j < M)
                if (Edge[i][j]==1) Out[i][j]++;
            }
          }
        }
}
```

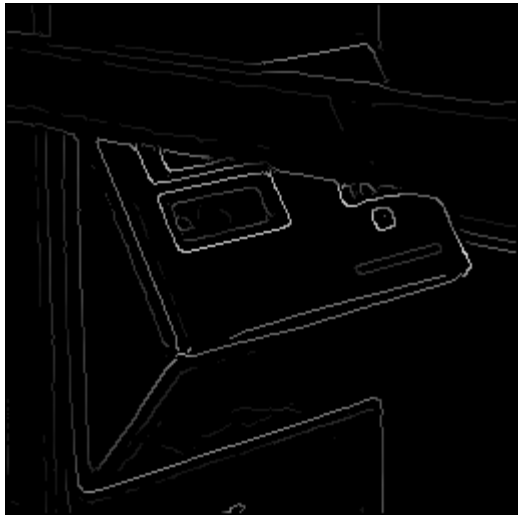
- Example 3: Original image and 20 most prominent lines:



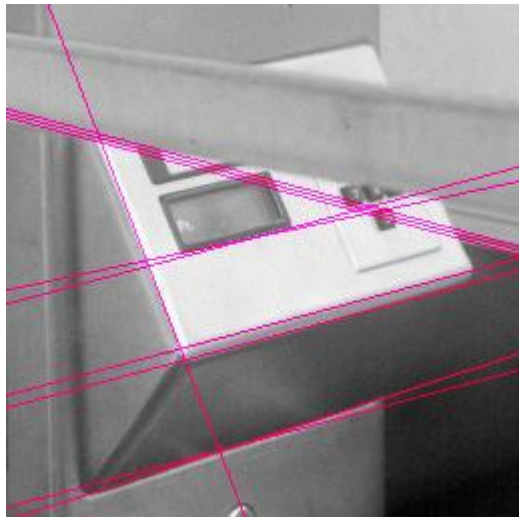
Sumber: INF 4300 – Hough transform, Anne Solberg

Real World Example

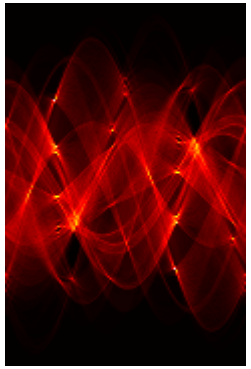
Original



Edge Detection



Found Lines



Parameter Space

Sumber: *Computer Vision*, S. Narasimhan

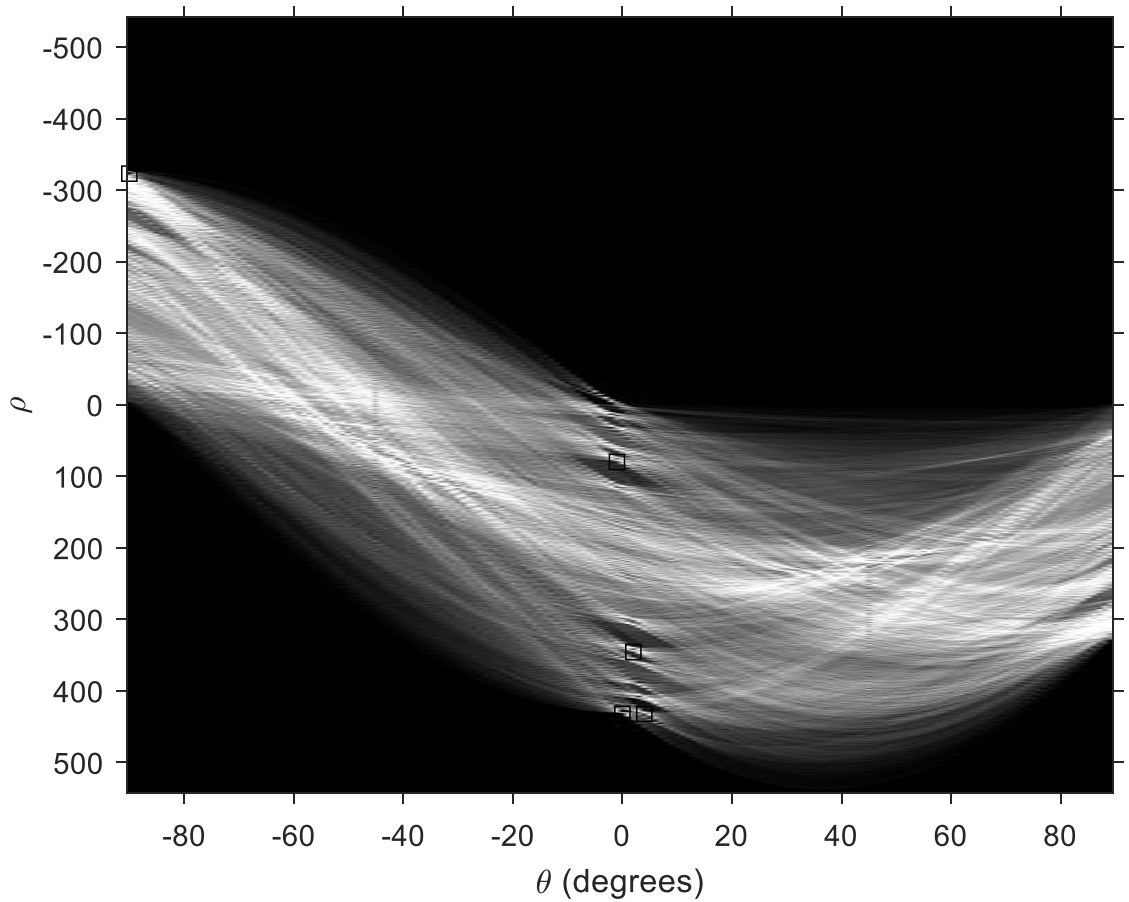
Program Matlab

```
% Baca citra
I = imread('hotel.bmp');
imshow(I);
```

```
% Deteksi tepi dengan operator canny
BW = edge(I,'canny');
figure, imshow(BW);
```

Sumber program: Mathworks

```
% Hitung transformasi Hough
[H,theta,rho] = hough(BW);

% Tampilkan H
figure
imshow(imadjust(mat2gray(H)),[],...
       'XData',theta,...
       'YData',rho,...
       'InitialMagnification','fit');
xlabel('\theta (degrees)')
ylabel('\rho')
axis on
axis normal
hold on
colormap(hot)
```

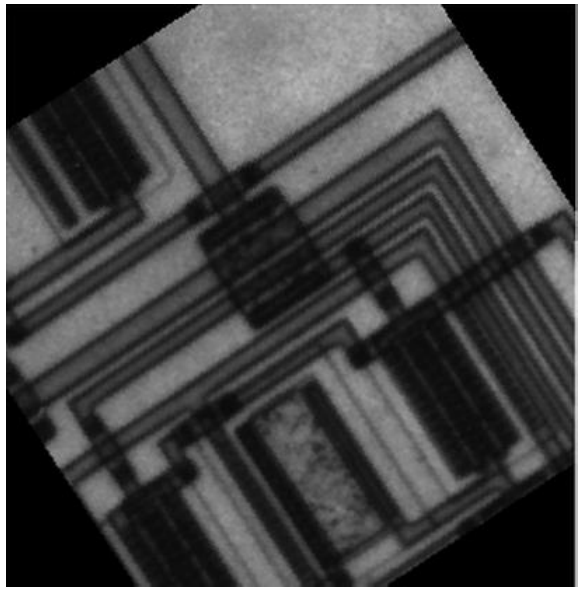


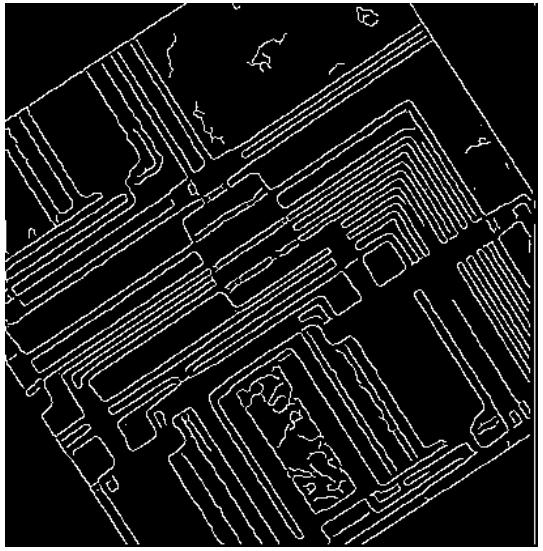

```
%Temukan hasil pemungutan suara yang memperoleh suara
% terbanyak dengan menggunakan fungsi houghpeaks
P = houghpeaks(H,5,'threshold',ceil(0.3*max(H(:))));

%Identifikasi puncak-puncak suara
x = theta(P(:,2));
y = rho(P(:,1));
plot(x,y,'s','color','black');
```

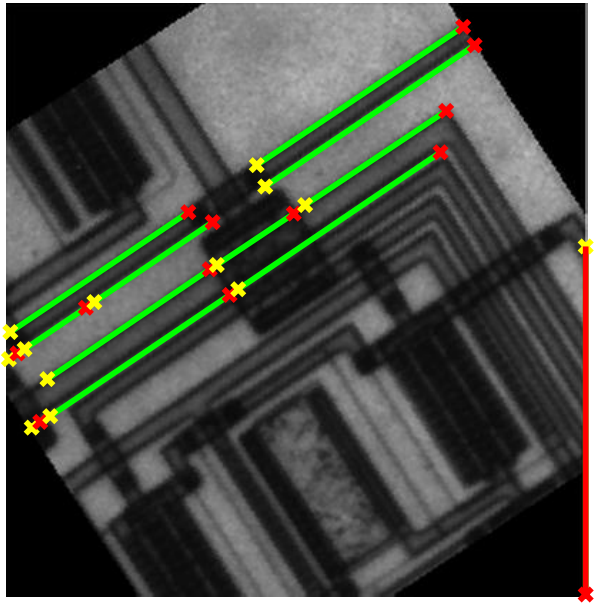
```
%Temukan hasil pemungutan suara yang memperoleh suara terbanyak dengan
% menggunakan fungsi houghpeaks
P = houghpeaks(H,5,'threshold',ceil(0.3*max(H(:))));

%Identifikasi puncak-puncak suara
x = theta(P(:,2));
y = rho(P(:,1));
plot(x,y,'s','color','black');

%Temukan garis-garis dengan fungsi houghlines
lines = houghlines(BW,theta,rho,P,'FillGap',5,'MinLength',7);
```

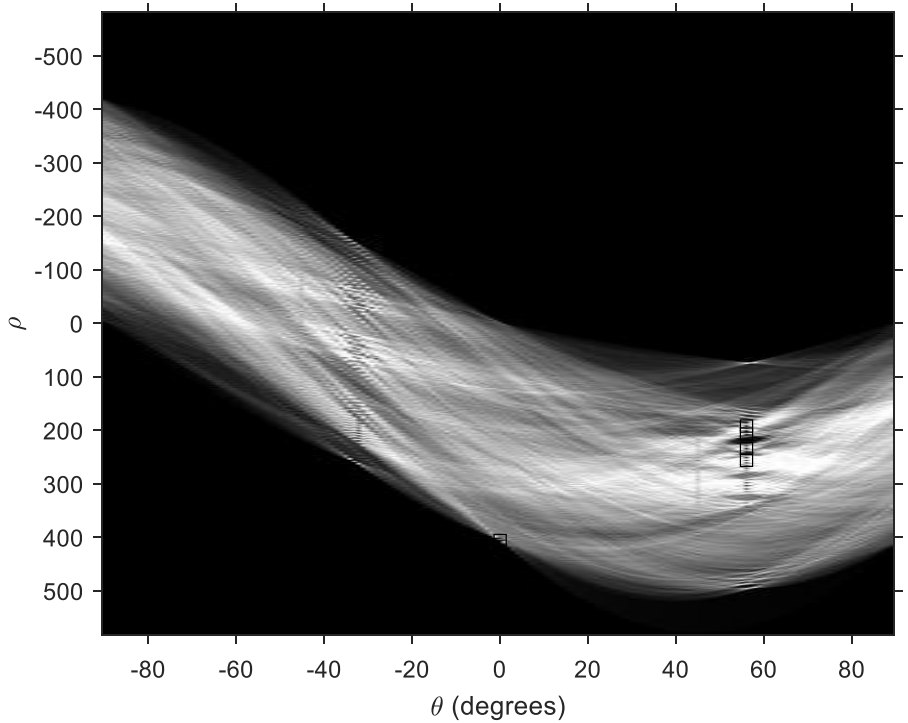
```
% Tampilkan citra semula dengan garis-garis hasil deteksi Hough
figure, imshow(I), hold on
max_len = 0;
for k = 1:length(lines)
   xy = [lines(k).point1; lines(k).point2];
   plot(xy(:,1),xy(:,2),'LineWidth',2,'Color','green');

   % Plot beginnings and ends of lines
   plot(xy(1,1),xy(1,2),'x','LineWidth',2,'Color','yellow');
   plot(xy(2,1),xy(2,2),'x','LineWidth',2,'Color','red');

   % Determine the endpoints of the longest line segment
   len = norm(lines(k).point1 - lines(k).point2);
   if ( len > max_len)
      max_len = len;
      xy_long = xy;
   end
end
% highlight the longest line segment
plot(xy_long(:,1),xy_long(:,2),'LineWidth',2,'Color','red');
```

Contoh lain:








-500
-400
-300
-200
-100
0
100
200
300
400
500
ρ
-80
-60
-40
-20
0
20
40
60
80
θ (degrees)

B. Mendeteksi lingkaran

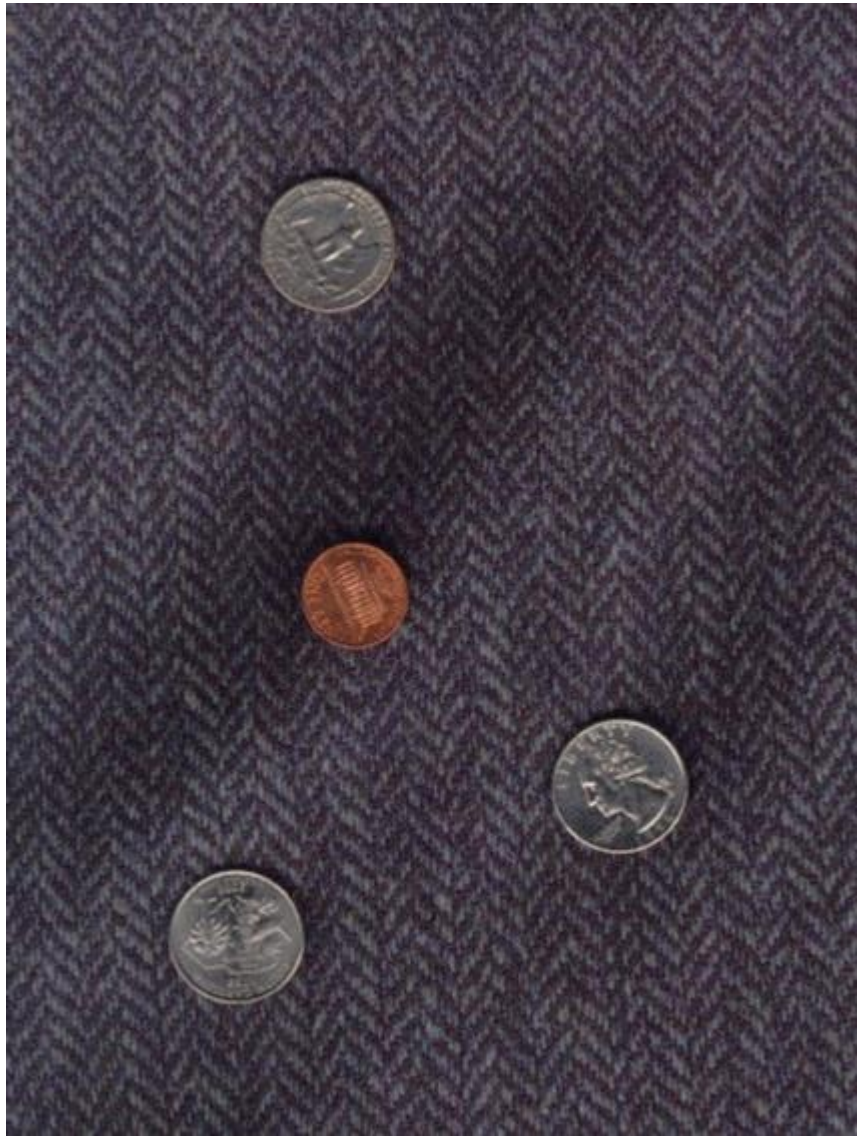
Contoh lingkaran dalam dunia nyata:

Sumber: *Computer Vision*, S. Narasimhan

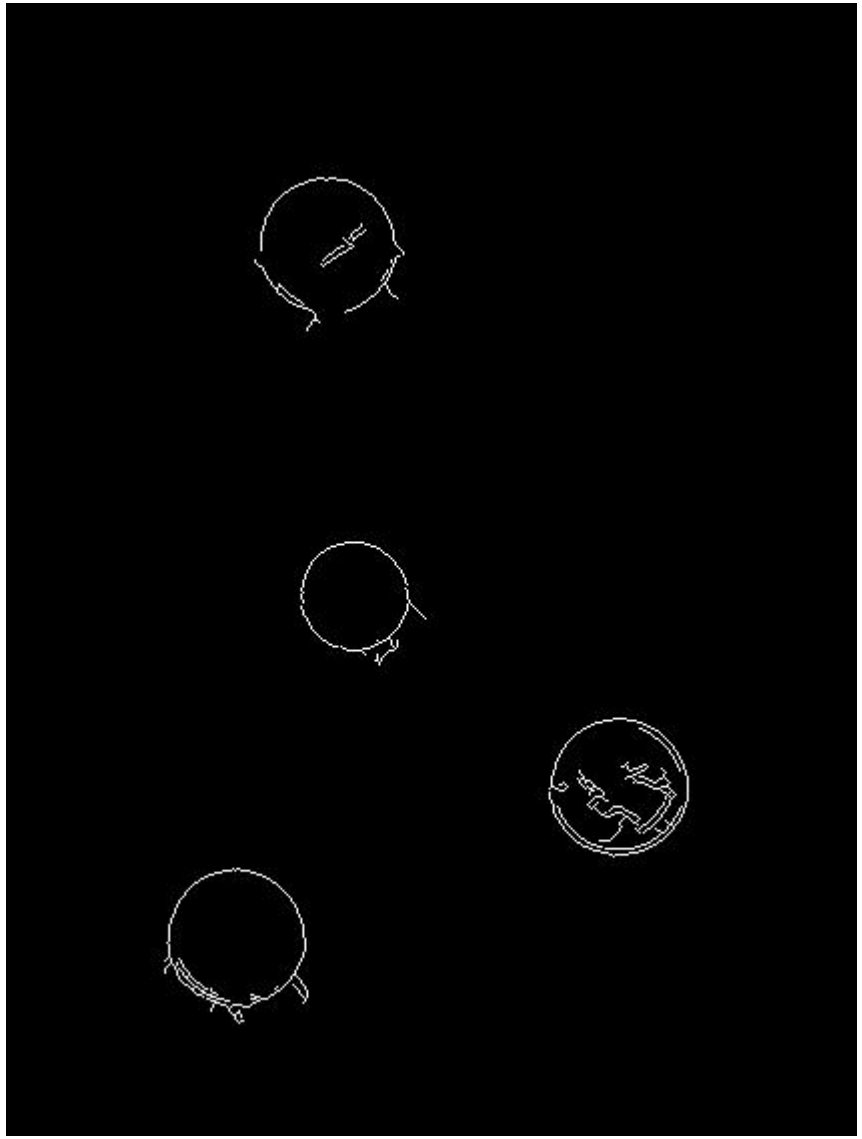
Finding Coins

Sumber: *Computer Vision*, S. Narasimhan

Original



Edges (note noise)



B. Mendeteksi lingkaran

- Transformasi Hough dapat juga digunakan untuk mendeteksi bentuk lingkaran di dalam citra tepi.

- Persamaan lingkaran yang berpusat di titik (*a*, *b*) dengan jari-jari *r* adalah

$$(x-a)^2+(y-b)^2=r^2$$

- Jadi, ruang parameter untuk lingkaran adalah *r*–*a*–*b* , sehingga matriks trimatra *P*(*r*, *a*, *b*) dibutuhkan untuk menyimpan perhitungan suara.

- Persamaan polar untuk setiap titik (x, y) di lingkaran:

$$x = a + r\cos\theta \quad (4)$$

$$y = b + r\sin\theta \quad (5)$$

Persamaan (4) dan (5) dapat ditulis menjadi persamaan

$$a = x - r\cos\theta \quad (6)$$

$$b = y - r\sin\theta \quad (7)$$

Pada operasi deteksi tepi, selain magnitudo *pixel* tepi, juga dihasilkan arah tepi, θ. Karena itu, cos θ dan sin θ dapat dihitung.

- Misalkan (x_i, yi) adalah pixel tepi dan θ adalah arah tepi . Ada dua kasus yang akan ditinjau:

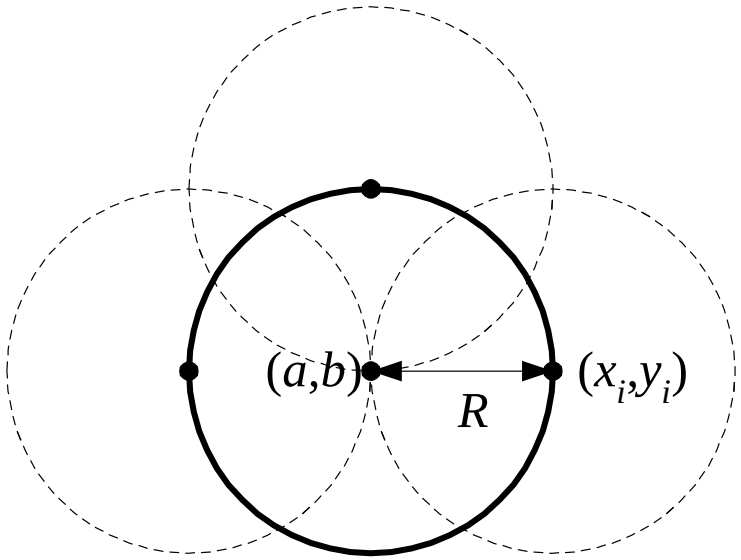
  (i) jika jari-jari lingkaran diketahui,

  (ii) jika jari-jari lingkaran tidak diketahui.

- Tinjau satu per satu kasusnya

Kasus 1: Jari-jari lingkaran diketahui

- Jika jari-jari jari-jari lingkaran diketahui $r = R$, maka ruang parametrik trimatra, $P(r,a,b)$, dapat direduksi menjadi ruang dwimatra, $P(a,b)$.



Proses penumpukan suara untuk mendeteksi lingkaran

- Titik pusat lingkaran, (a,b), yang mempunyai jari-jari $r = R$ dan melalui titik (x_i, y_i) dapat dihitung dengan persamaan

$$a = x_i - R \cos \theta$$
$$b = y_i - R \sin \theta$$

seperti yang ditunjukkan pada gambar, lalu tambahkan elemen $P(a, b)$ yang bersesuaian dengan satu.

- Proses ini diulangi untuk *pixel-pixel* tepi yang lain.

- Elemen matriks $P(a, b)$ yang memiliki jumlah suara di atas nilai ambang tertentu menyatakan lingkaran yang terdapat di dalam citra tepi.

Kasus 2: Jari-jari lingkaran tidak diketahui

- Jika jari-jari lingkaran tidak diketahui, maka penumpukan suara dilakukan untuk semua nilai r, $0 < r \leq r_{max}$, nilai a dan b untuk *pixel* tepi (x_i, y_i) dihitung dengan persamaan

$$a = x_i - r \cos \theta \quad (8)$$

$$b = y_i - r \sin \theta \quad (9)$$

dan elemen $P(r, a, b)$ yang bersesuaian dinaikkan satu (P(r, a, b) = P(r, a, b) + 1)

- Proses ini diulangi untuk *pixel-pixel* tepi yang lain.

- Elemen matriks $P(r, a, b)$ yang memiliki jumlah suara di atas nilai ambang tertentu menyatakan lingkaran yang terdapat di dalam citra tepi.

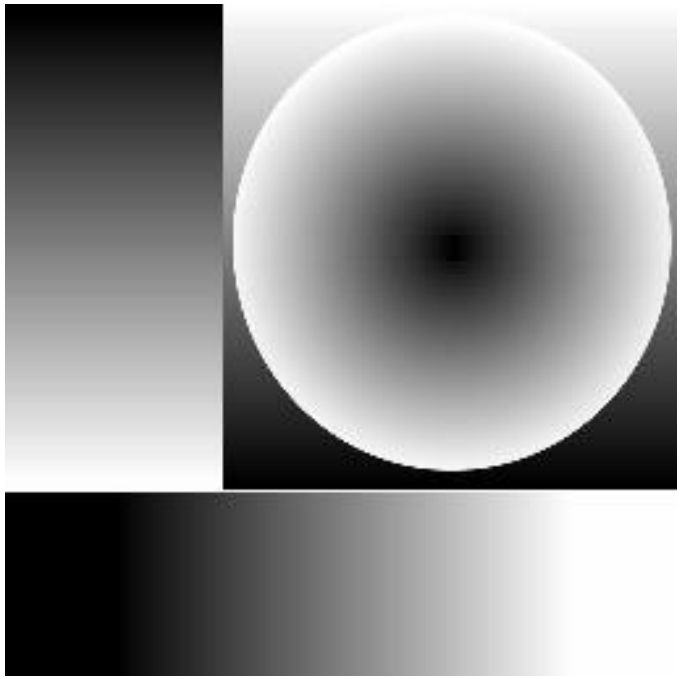
- Persamaan (8) dan (9) dapat dimanipulasi dengan mengeliminasi r dari kedua persamaan:

$$a = x - r\cos\theta \rightarrow r = \frac{(x-a)}{\cos\theta}$$

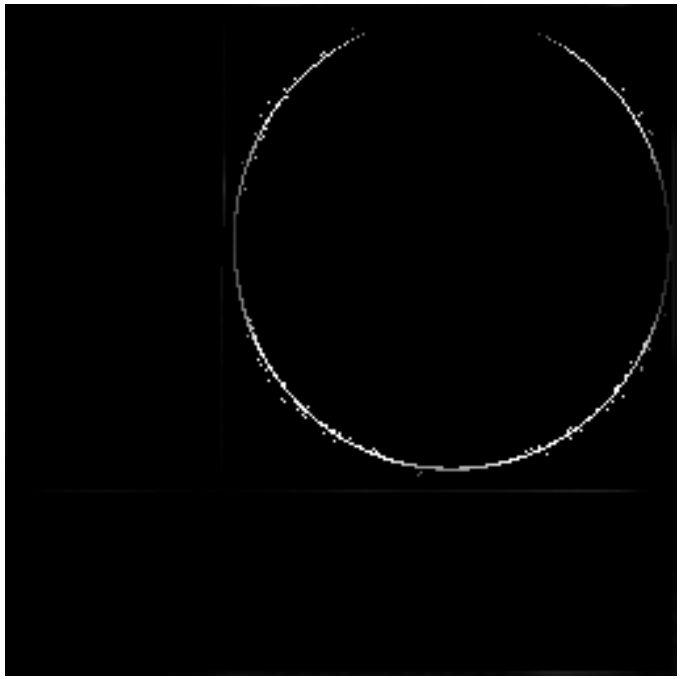
$$b = y - r\sin\theta \rightarrow b = y - \frac{(x-a)}{\cos\theta}\sin\theta = y - (x-a)\tan\theta$$

$$b = a\tan\theta - x\tan\theta + y \qquad (10)$$

- Dengan demikian, maka ruang parametrik trimatra, $P(r,a,b)$, dapat direduksi menjadi ruang dwimatra, $P(a,b)$.
- Untuk untuk semua nilai a, yang dalam hal ini $a_1 < a \leq a_K$, nilai ordinat b dari titik pusat lingkaran (a,b) yang melalui titik (x_i, y_i) dapat dihitung dengan persamaan (10), lalu tambahkan elemen $P(a, b)$ yang bersesuaian dengan satu ($P(a, b) = P(a, b) + 1$) .
- Proses ini diulangi untuk *pixel-pixel* tepi yang lain. Elemen matriks $P(a, b)$ yang memiliki jumlah suara di atas nilai ambang tertentu menyatakan lingkaran yang terdapat di dalam citra tepi



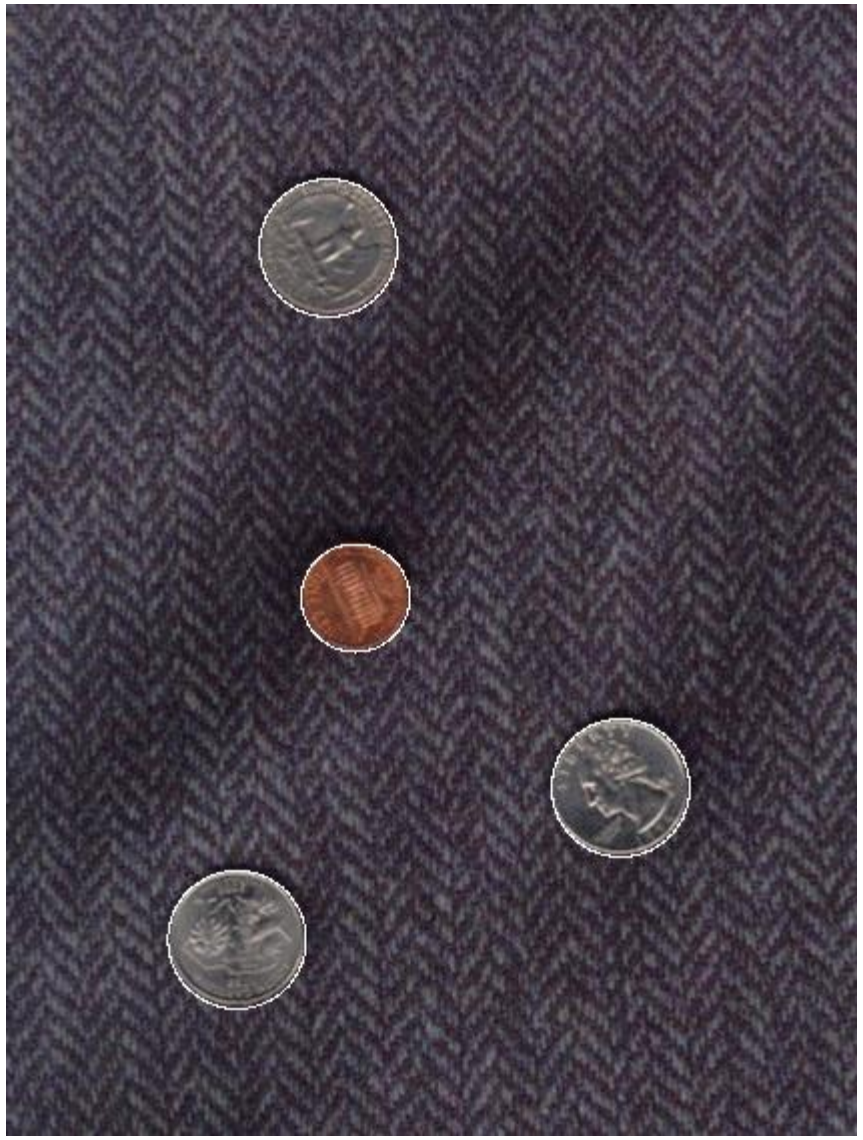
Citra slope



Hasil deteksi lingkaran dengan Transformasi Hough (T = 30)

Finding Coins (Continued)

Sumber: *Computer Vision*, S. Narasimhan



Note that because the quarters and penny are different sizes, a different Hough transform (with separate accumulators) was used for each circle size.

Coin finding sample images from: Vivek Kwatra

C. Mendeteksi kurva sembarang

- Transformasi Hough dapat dirampatkan untuk mendeteksi sembarang kurva yang berbentuk $f(\mathbf{x}, \mathbf{a}) = 0$, yang dalam hal ini **x** adalah vektor peubah dan **a** adalah vektor parameter.

- Tahapan yang dilakukan adalah:
  1. tentukan lokasi pusat penumpukan suara;
  2. tentukan fungsi jarak dari setiap *pixel* tepi ke pusat pemungutan suara.

- Sebagai contoh, lihat gambar, titik (a, b) adalah lokasi pusat penumpukan suara. Fungsi jarak *r* dari setiap titik (x, y) dan nilai α merupakan fungsi dari arah vektor normal N.

- Untuk setiap pixel tepi (x,y) dengan sudut arah tepi θ, lokasi pusat penumpukan suara dihitung dengan rumus

$$a = x - r(\theta)\cos(\alpha(\theta))$$

$$b = y - r(\theta)\sin(\alpha(\theta))$$