

20 - Pendeteksian Tepi (Bagian 3)

IF4073 Pemrosesan Citra Digital

Oleh: Dr. Ir. Rinaldi, M.T



Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung
2025

Citra Tepi

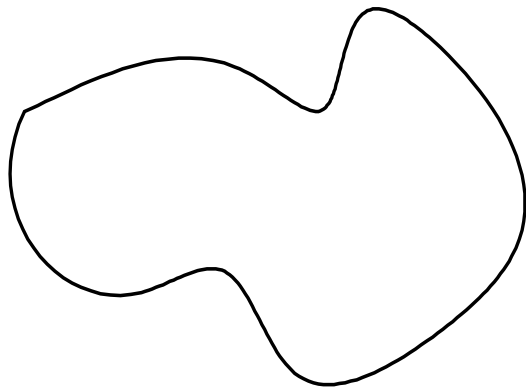
- Pendeteksi tepi menghasilkan citra tepi (*edges image*) yang berupa citra biner
- *Pixel-pixel* tepi berwarna putih, sedangkan *pixel* bukan-tepi berwarna hitam.



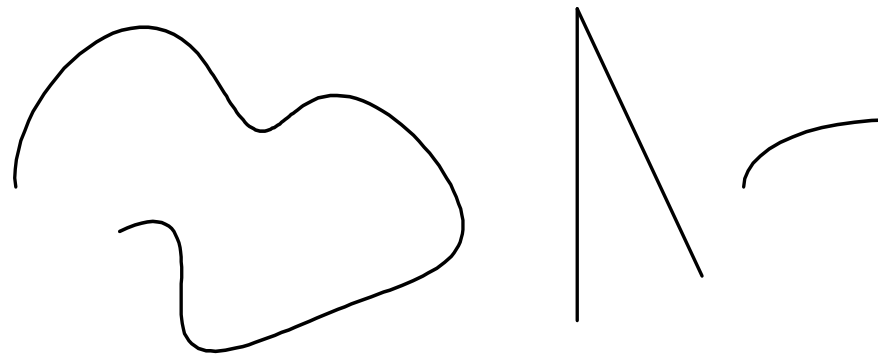
- Tetapi, tepi belum memberikan informasi yang berguna karena belum ada keterkaitan antara suatu tepi dengan tepi lainnya.
- Oleh karena itu, pixel-pixel tepi harus ditautkan satu sama lain agar menghasilkan informasi yang lebih berguna yang dapat digunakan dalam mendeteksi bentuk-bentuk sederhana.
- Bentuk-bentuk sederhana misalnya garis lurus, lingkaran, elips, dan sebagainya. Bentuk-bentuk ini berguna di dalam proses analisis citra.
- Oleh karena itu, algoritma deteksi tepi biasanya dilanjutkan dengan prosedur penautan (*linking procedure*) untuk menyusun *pixel-pixel* tepi menjadi representasi yang lebih berarti.

Kontur

- Rangkaian *pixel-pixel* tepi membentuk batas daerah (*region boundary*) yang dinamakan **kontur**
- Kontur dapat terbuka atau tertutup.



(a) Kontur tertutup



(b) Kontur terbuka

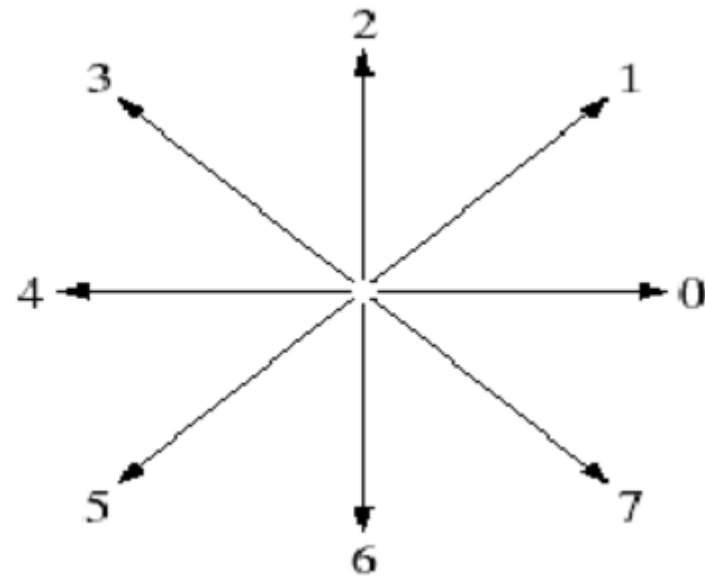
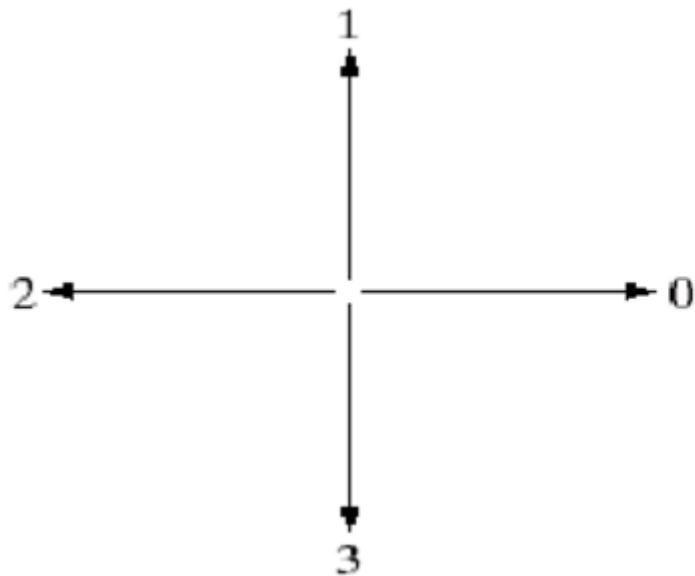
- Kontur tertutup berkoresponden dengan batas yang mengelilingi suatu daerah (*region*).
- *Pixel-pixel* di dalam daerah tertutup dapat ditemukan dengan algoritma pengisian (*filling algorithm*).
- Batas daerah berguna untuk mendeskripsikan bentuk objek dalam tahap analisis citra (misalnya untuk mengenali objek tersebut).
- Kontur terbuka dapat berupa fragmen garis atau bagian dari batas daerah yang tidak membentuk sirkuit

Representasi Kontur

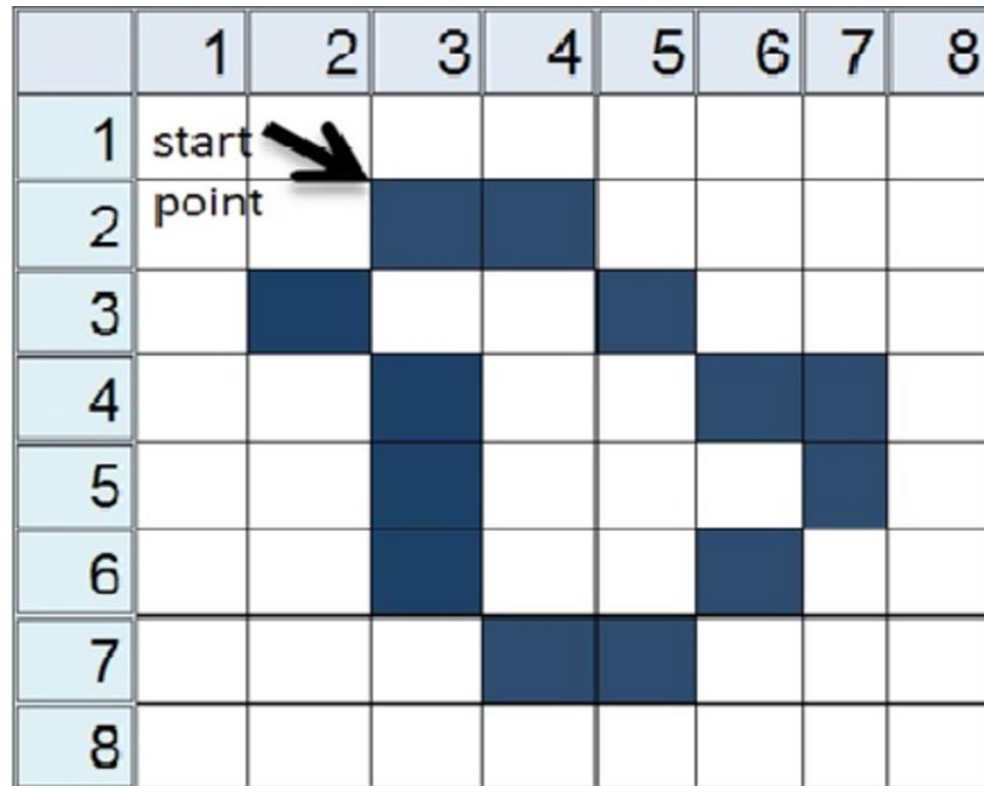
- Representasi kontur dapat berupa senarai tepi (*edge list*) atau berupa kurva.
- Senarai tepi merupakan himpunan terurut *pixel-pixel* tepi.
- Representasi kontur ke dalam kurva merupakan representasi dalam bentuk persamaan. Misalnya, rangkaian *pixel* tepi yang membentuk garis dapat direpresentasikan hanya dengan sebuah persamaan garis lurus.
- Representasi semacam ini menyederhanakan perhitungan selanjutnya seperti arah dan panjang garis

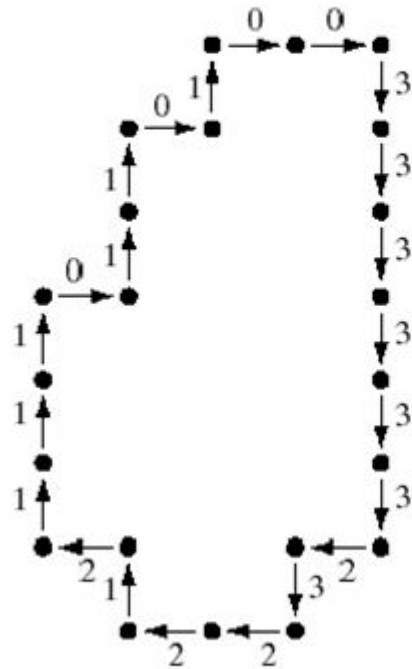
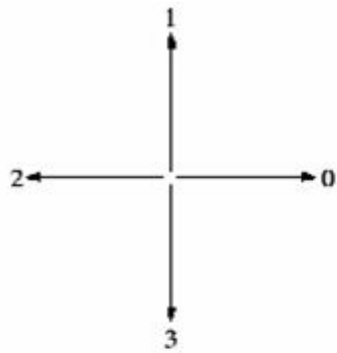
Kode Rantai

- **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.

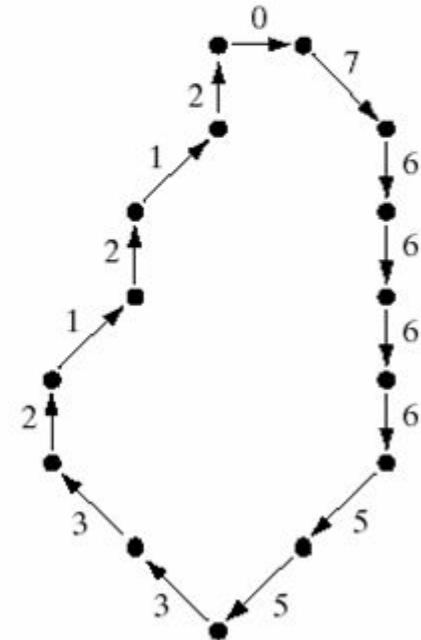
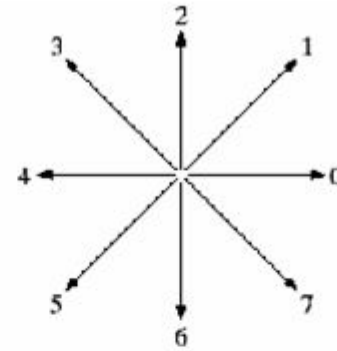


- Dimulai dari sebuah *pixel* tepi dan searah jarum jam, arah setiap *pixel* tepi yang membentuk batas objek dikodekan dengan salah satu dari empat atau delapan kode rantai.
- Kode rantai merepresentasikan batas objek dengan koordinat *pixel* tepi pertama lalu diikuti dengan senarai kode rantai



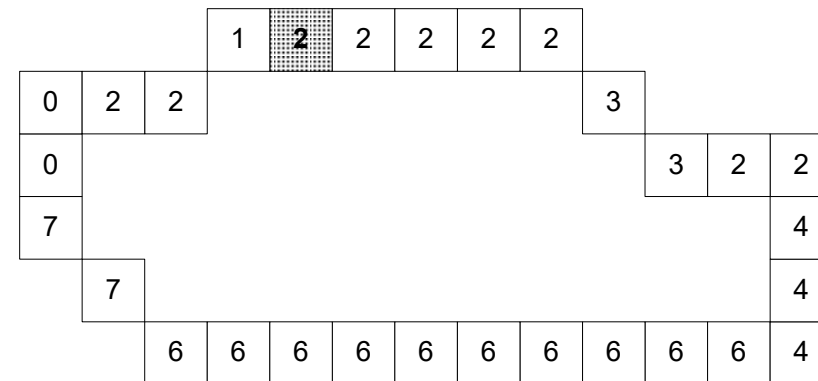
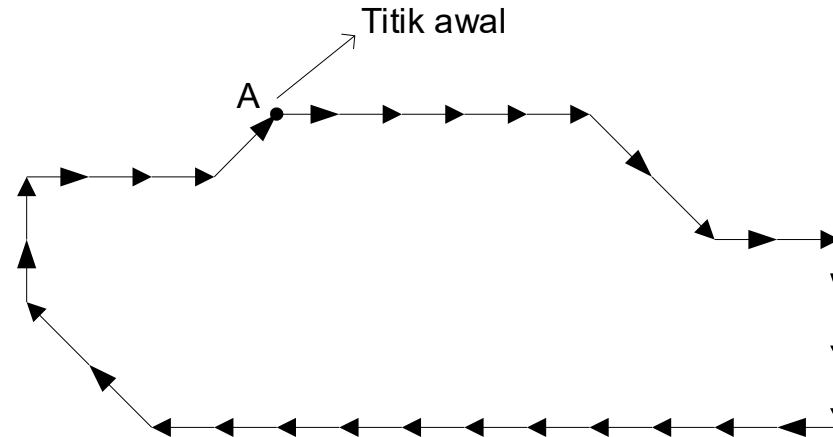
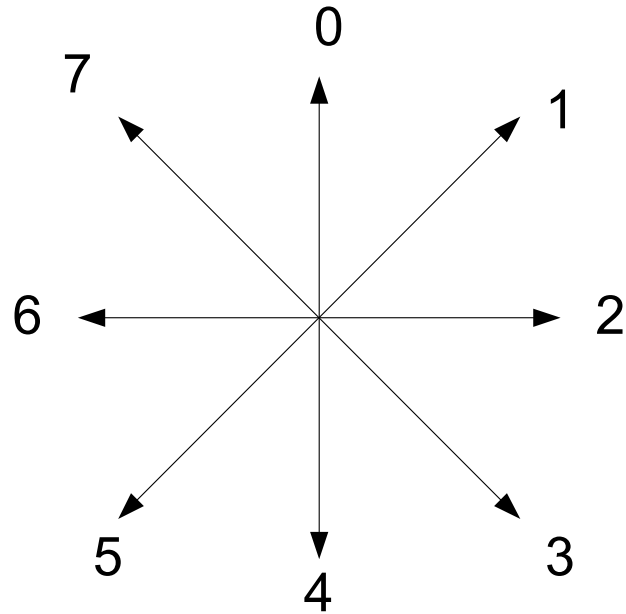


4-directional chain code:
003333323221211101101



8-directional chain code:
076666553321212

- Kode rantai yang dihasilkan tergantung dari titik mana dimulai



Kode rantai: 22223322444666666667700221

Transformasi Hough

- Kurva yang merepresentasikan kontur dicari dengan teknik pencocokan kurva (*curve fitting*).
- Ada dua macam teknik pencocokan 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 teratur dalam bentuk parameterik (kurva berupa 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. Parameter pemenang inilah yang menyatakan parameter persamaan kurva.

- Transformasi Hough dipatenkan oleh Paul Hough pada tahun 1962. Awalnya transformasi ini digunakan untuk mendeteksi garis lurus di dalam citra.
- Generalisasi transformasi Hough ditemukan oleh Richard Duda dan Peter Hart pada tahun 1972, yang dikenal dengan nama "*generalized Hough transform*" sehingga dapat digunakan untuk deteksi lingkaran dan bentuk-bentuk lainnya.
- Makalah Richard Duda dan Peter Hart di jurnal *Communications of the ACM* Volume 15 Issue 1 Jan. 1972

Duda, R.O.; Hart, P. E. (January 1972). ["Use of the Hough Transformation to Detect Lines and Curves in Pictures"](#). *Comm. ACM*. **15**: 11–15. [doi:10.1145/361237.361242](#). [S2CID 1105637](#).



Use of the Hough Transformation To Detect Lines and Curves in Pictures

Richard O. Duda and Peter E. Hart
Stanford Research Institute,
Menlo Park, California

Hough has proposed an interesting and computationally efficient procedure for detecting lines in pictures. This paper points out that the use of angle-radius rather than slope-intercept parameters simplifies the computation further. It also shows how the method can be used for more general curve fitting, and gives alternative interpretations that explain the source of its efficiency.

Key Words and Phrases: picture processing, pattern recognition, line detection, curve detection, colinear points, point-line transformation, Hough transformation

CR Categories: 3.63

1. Introduction

A recurring problem in computer picture processing is the detection of straight lines in digitized images. In the simplest case, the picture contains a number of discrete, black figure points lying on a white background. The problem is to detect the presence of groups of colinear or almost colinear figure points. It is clear that the problem can be solved to any desired degree of accuracy by testing the lines formed by all pairs of points. However, the computation required for n points is approximately proportional to n^2 , and may be prohibitive

A. Transformasi Hough untuk Mendeteksi Garis Lurus



Contoh citra dengan struktur linier



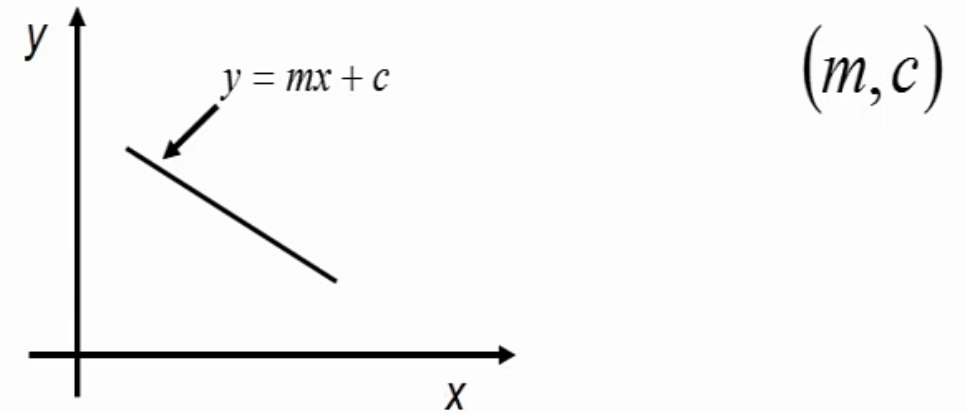
Citra hasil deteksi tepi

Sumber: INF 4300 – Hough transform, Anne Solberg

- Persamaan garis lurus:

$$y = mx + c$$

(1)



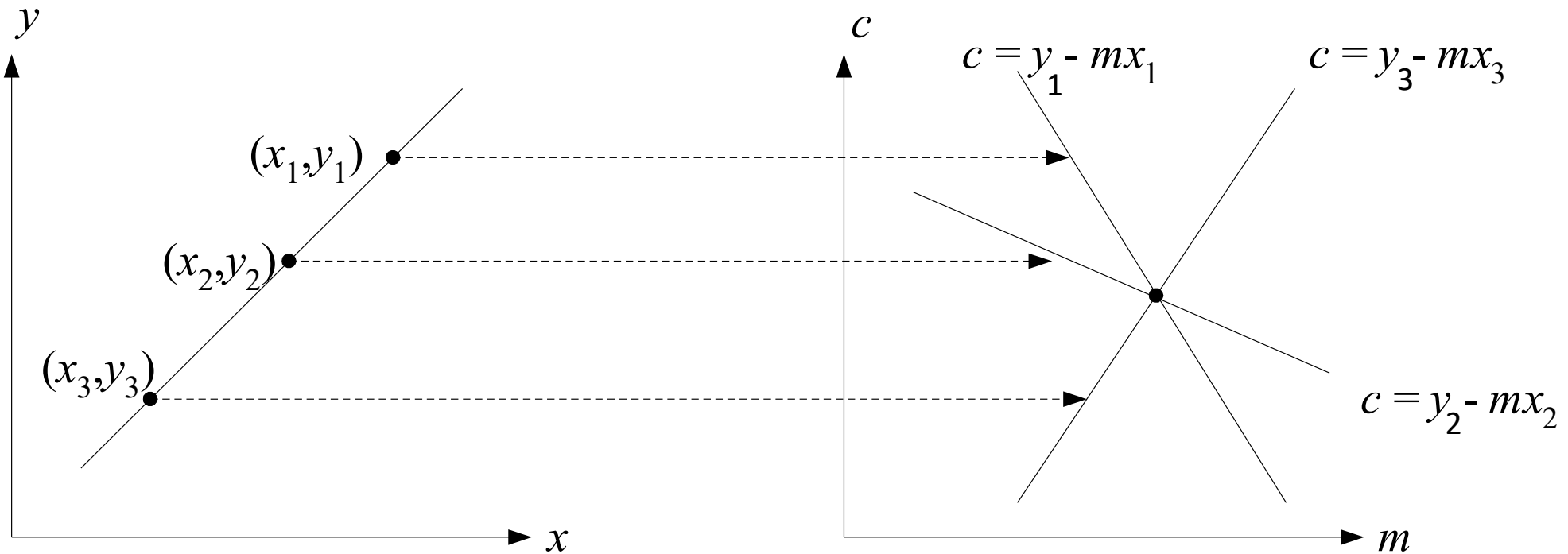
- Dalam bentuk parametrik, setiap garis dinyatakan di dalam ruang parameter (m, c)
Persamaan garis lurus dalam ruang (m, c) ditulis menjadi

$$c = y - mx$$

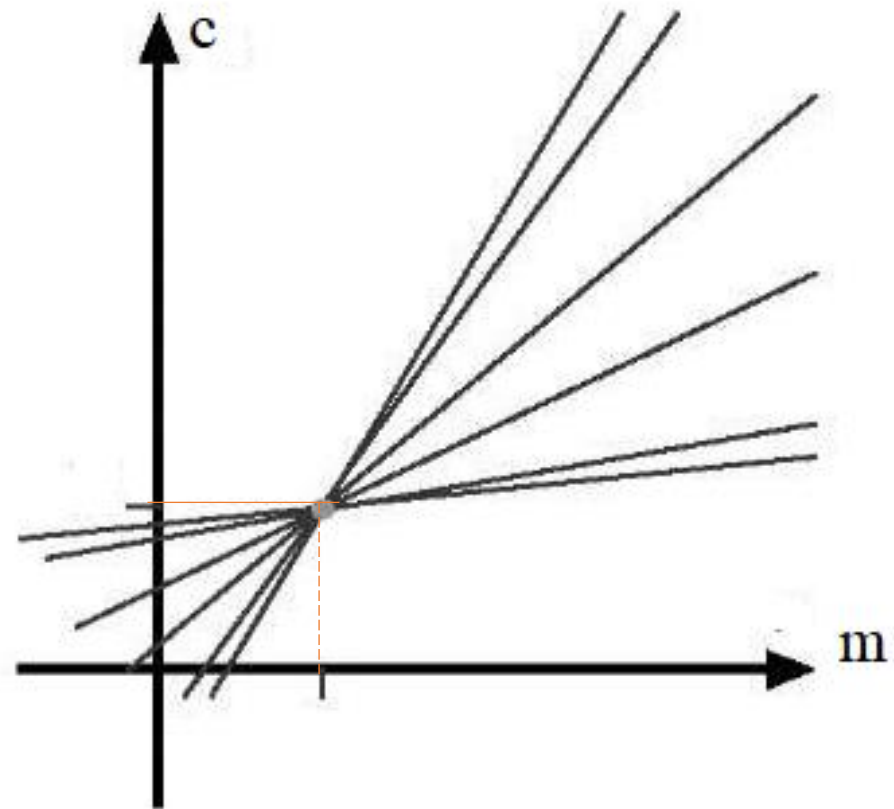
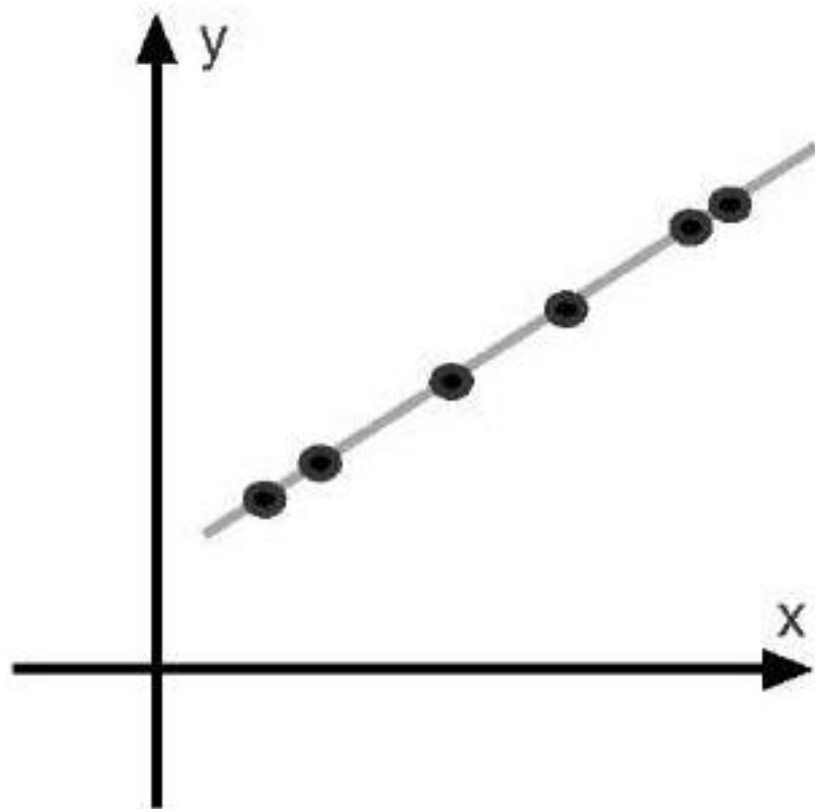
(2)

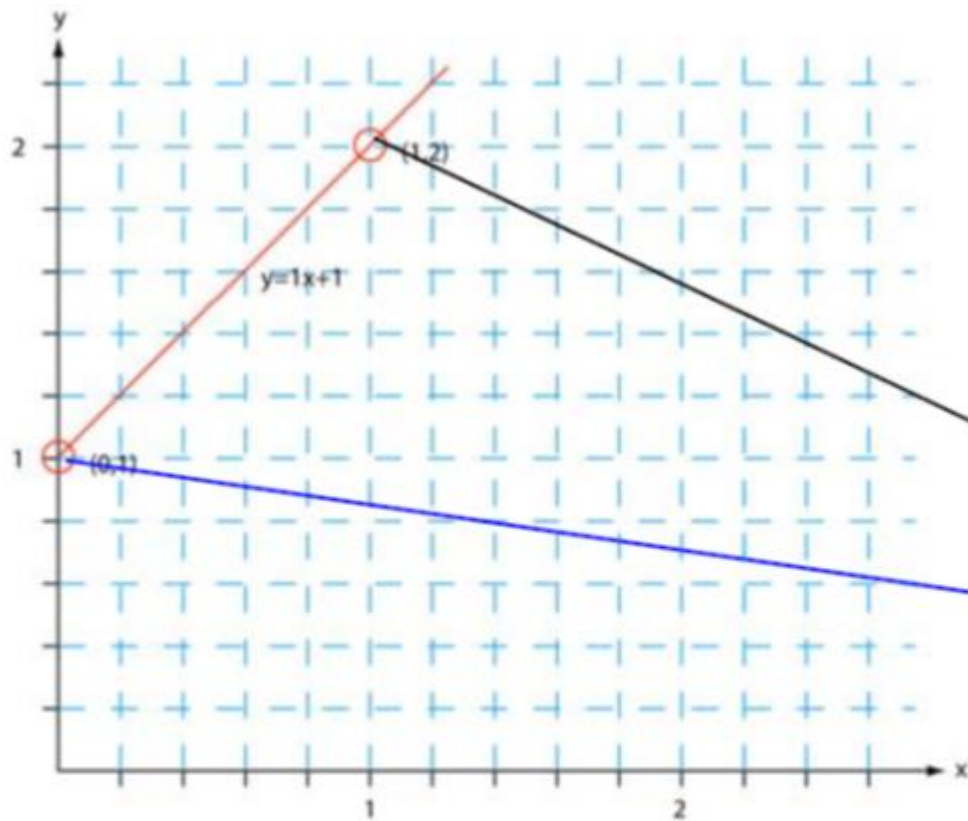
- Sembarang titik (x, y) pada bidang planar X - Y berkoresponden dengan sebuah garis lurus pada ruang parameter (m, c) .

- Setiap *pixel* pada garis lurus di bidang citra (bidang x-y) berkoresponden dengan sebuah garis lurus di dalam bidang m-c. Semua garis lurus itu berpotongan pada **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.

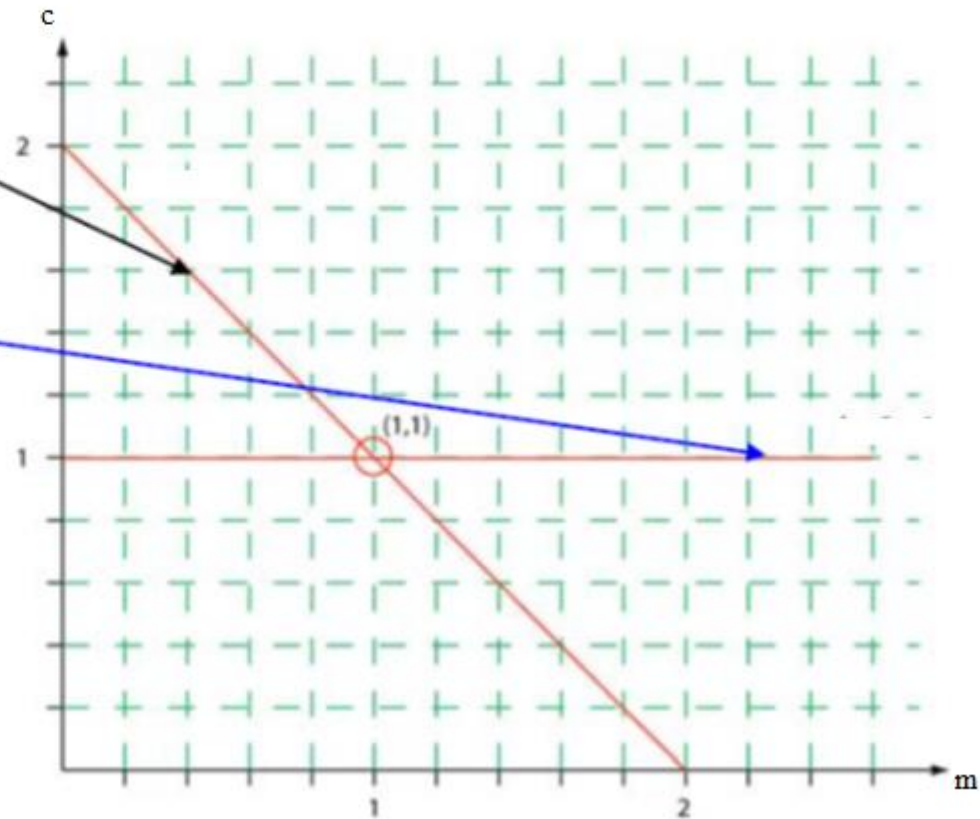




ruang (x,y)



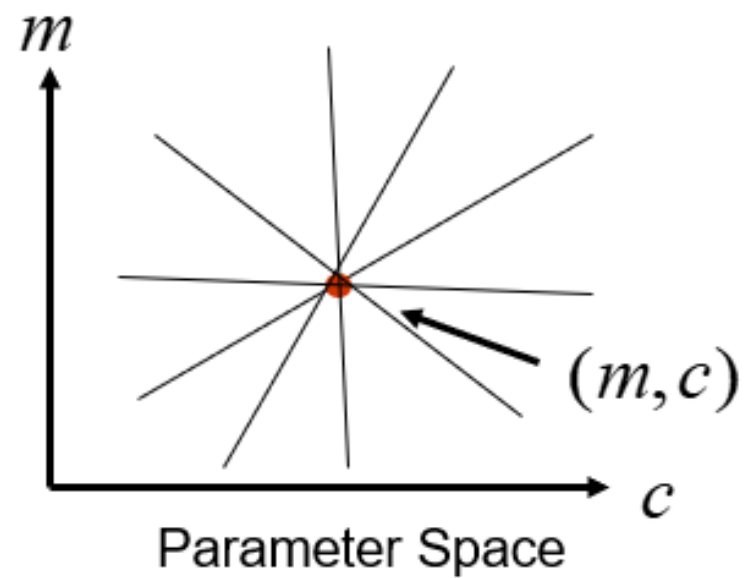
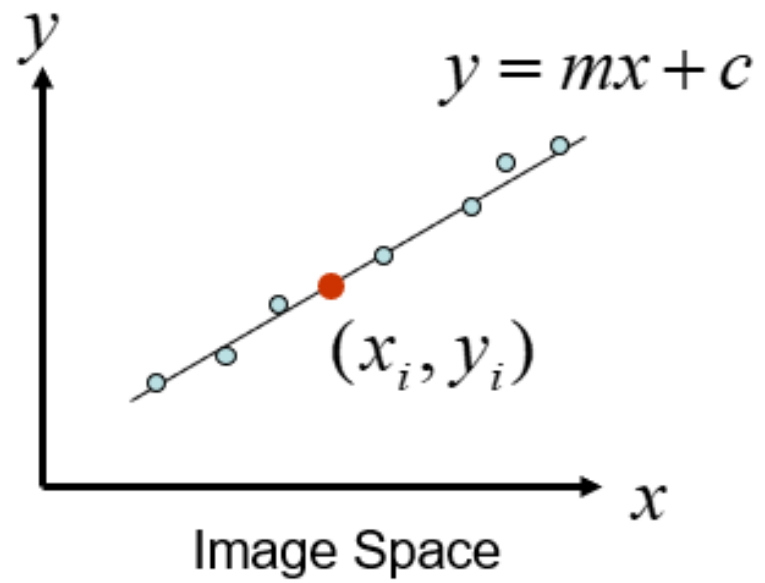
Sebuah titik di (x,y) memberikan sebuah garis pada ruang parameter (m,c)



ruang (m,c)

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 (m, c) 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:

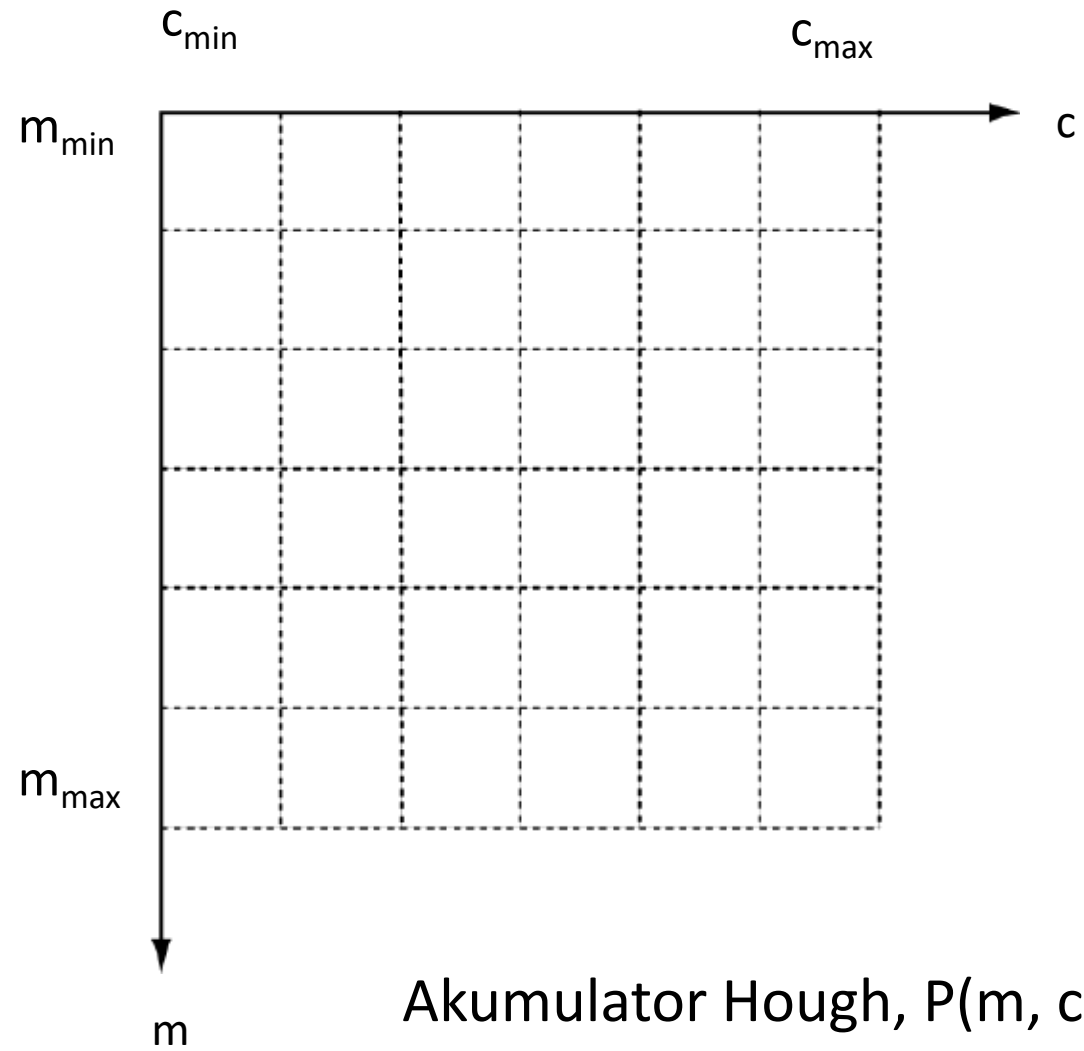
$$P(m, c) = P(m, c) + 1$$

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

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

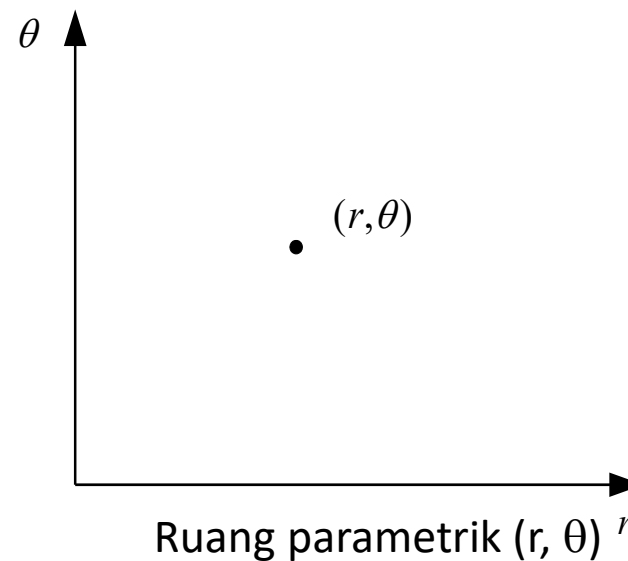
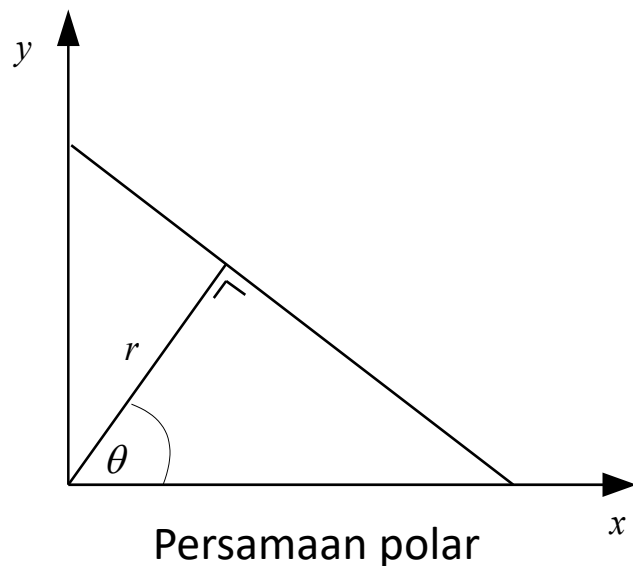
Hal ini berarti terdapat k garis lurus yang terdeteksi pada citra.



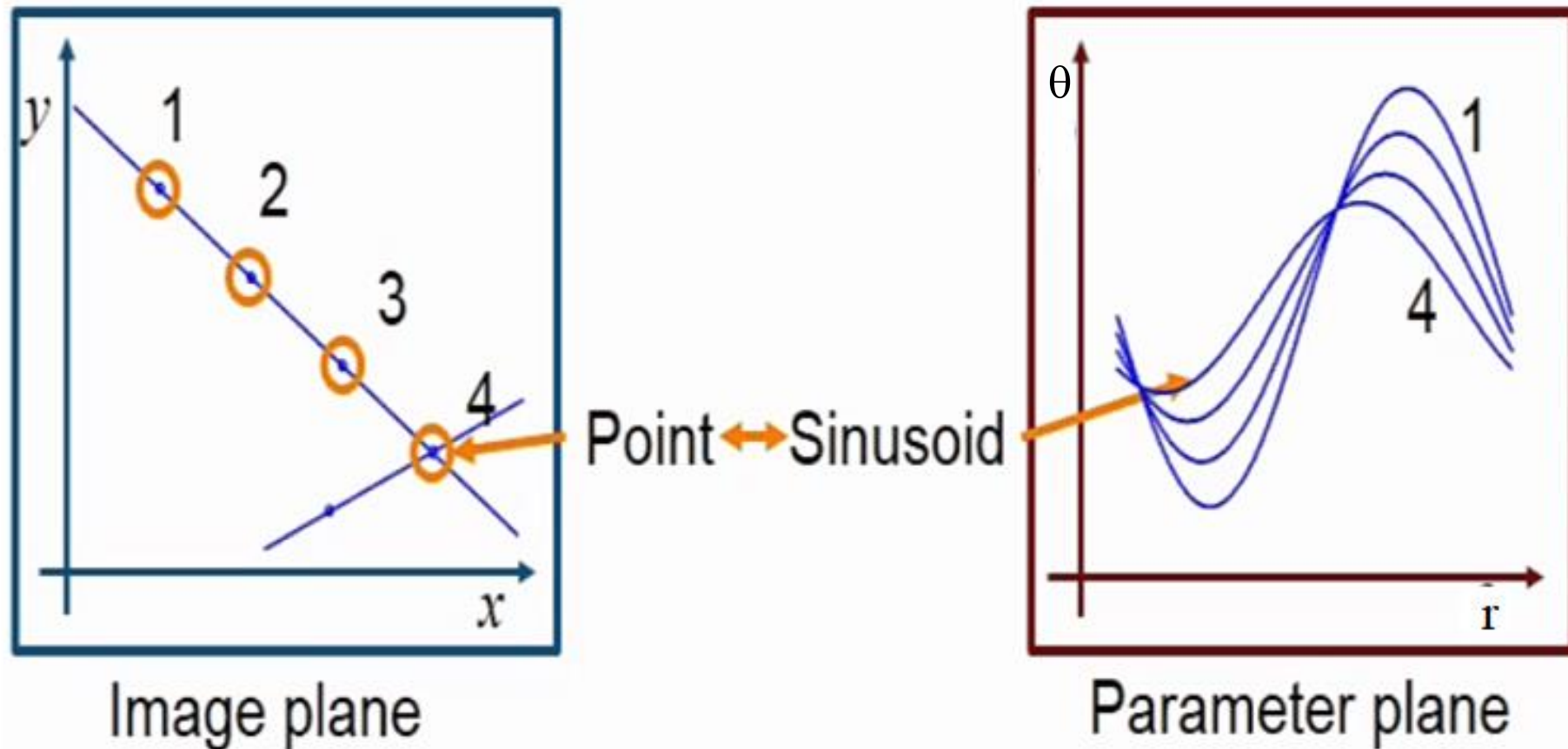
- Model parametrik pada persamaan (2) tidak dapat digunakan untuk mendeteksi garis vertikal karena gradiennya (m) nilai tak-berhingga.
- Selain itu, kebutuhan memori untuk akumulatornya besar, karena $-\infty \leq m \leq \infty$
- Oleh karena itu, garis dinyatakan dalam representasi polar:

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

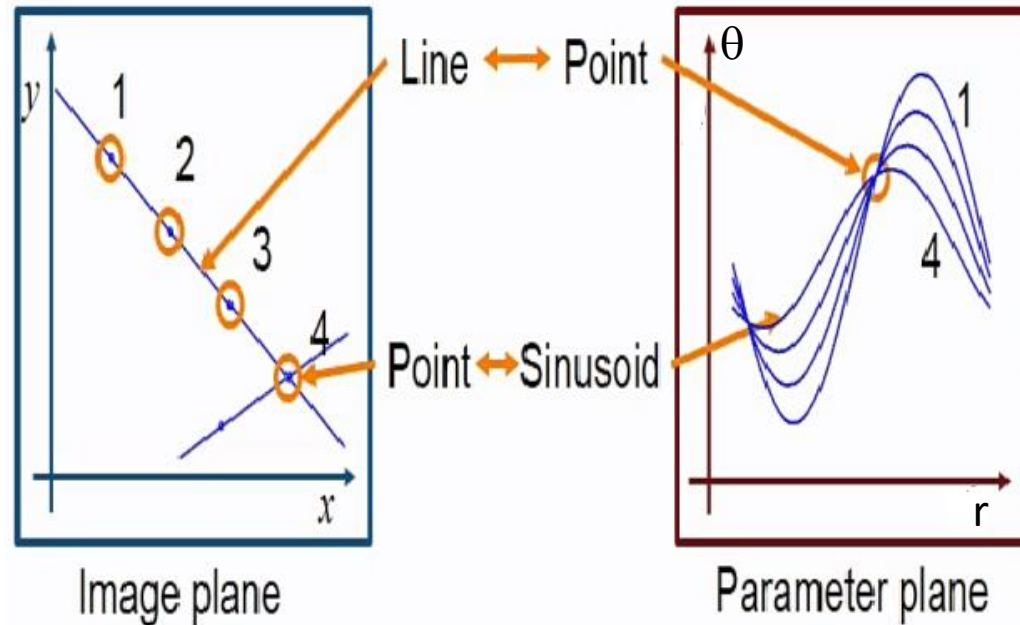
yang dalam hal ini r adalah jarak garis ke titik asal



- Setiap titik (x_1, y_1) pada garis lurus di 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 Bahasa C untuk Hough Transform

Input: citra tepi (citra hasil pendeteksian tepi), disimpan di dalam matriks `Edge [0..M-1, 0..N-1]`.

Ruang parameter $r-\theta$ dinyatakan sebagai matriks `P` yang berukuran $p \times q$. Nilai *cosinus* dan *sinus* disimpan di dalam *lookup table* `COS [0..p-1]` dan `SIN [0..p-1]`

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

    /* 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(M*M+N*N) sampai sqrt(M*M+N*N).
*/

for (k=0;k<=M-1;k++)
    for (l=0;l<=N-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<=m-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 parameter (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 $\{(r_1, \theta_1), (r_2, \theta_2), \dots, (r_k, \theta_k)\}$, hal ini berarti terdapat k garis lurus yang terdeteksi pada citra.

```
void threshold(imatriks P, int p, in q, int T)
/* Melakukan pengembangan pada matriks parameter P.
   Setiap elemen matriks P yang nilainya di atas T menyatakan
   parameter garis lurus.
   Masukan: matriks parameter P yang berukuran p x q.
   Keluaran: matriks parameter P yang sudah di-threshold.
*/
{ int i, j;
  for(i=0;i<p;i++)
    for(j=0;j<q;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 M, int N, imatriks P,
                  int p, int q, float *COS, float *SIN)
/* prosedur yang melakukan Transformasi Hough Balikan
Masukan: 1. citra tepi Edge yang berukuran M x N.
          2. matriks parameter P yang berukuran p x q
Keluaran: citra Out yang berisi pixel-pixel pembentuk garis lurus.
*/
{
    int k, l, i, j;
    float r, y;
    float SQRTD =sqrt((float)M*(float)M + (float)N*(float)N);

    /* 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<=M-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 < N)
                    if (Edge[i][j]==1) Out[i][j]++;
            }
        }
    }
}

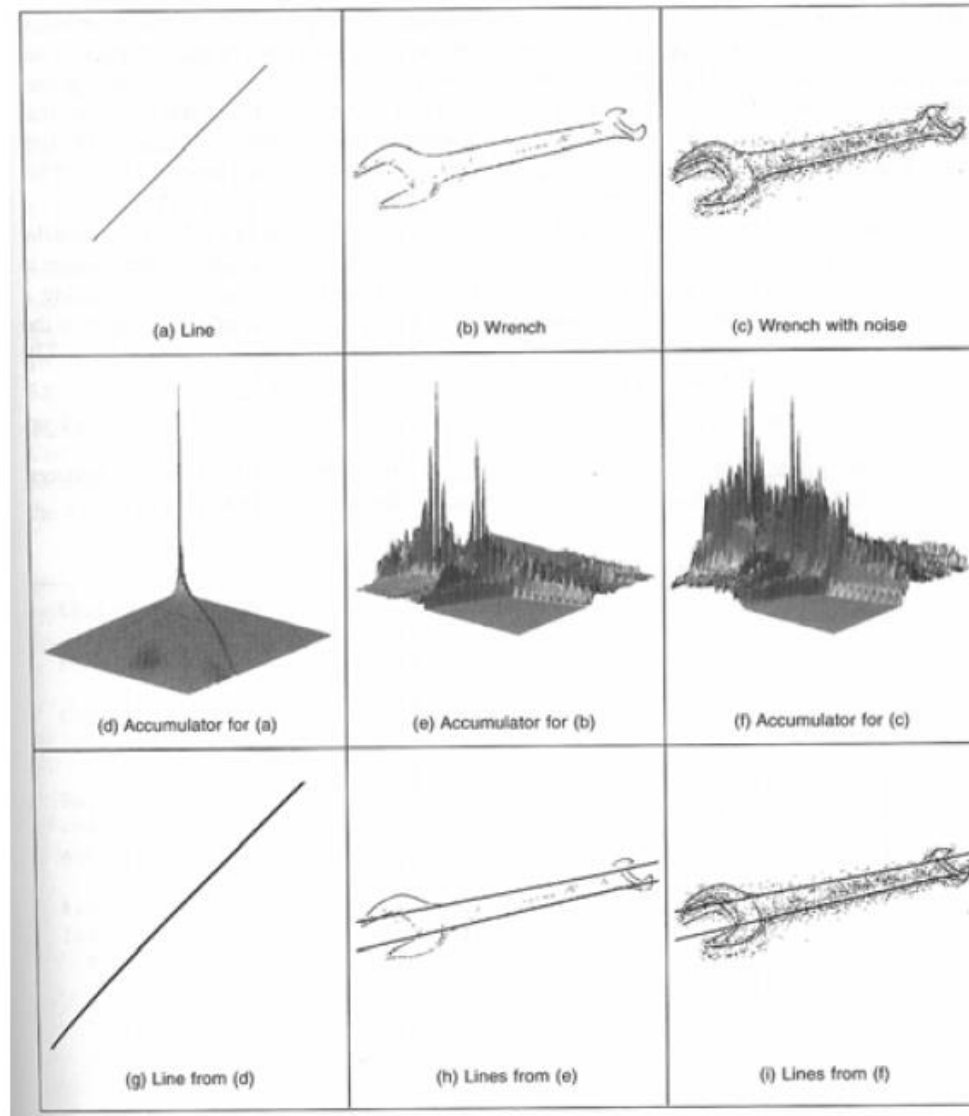
```

Transformasi Hough dapat mendeteksi garis meskipun citra mengandung derau sekalipun

Thresholded
edge images

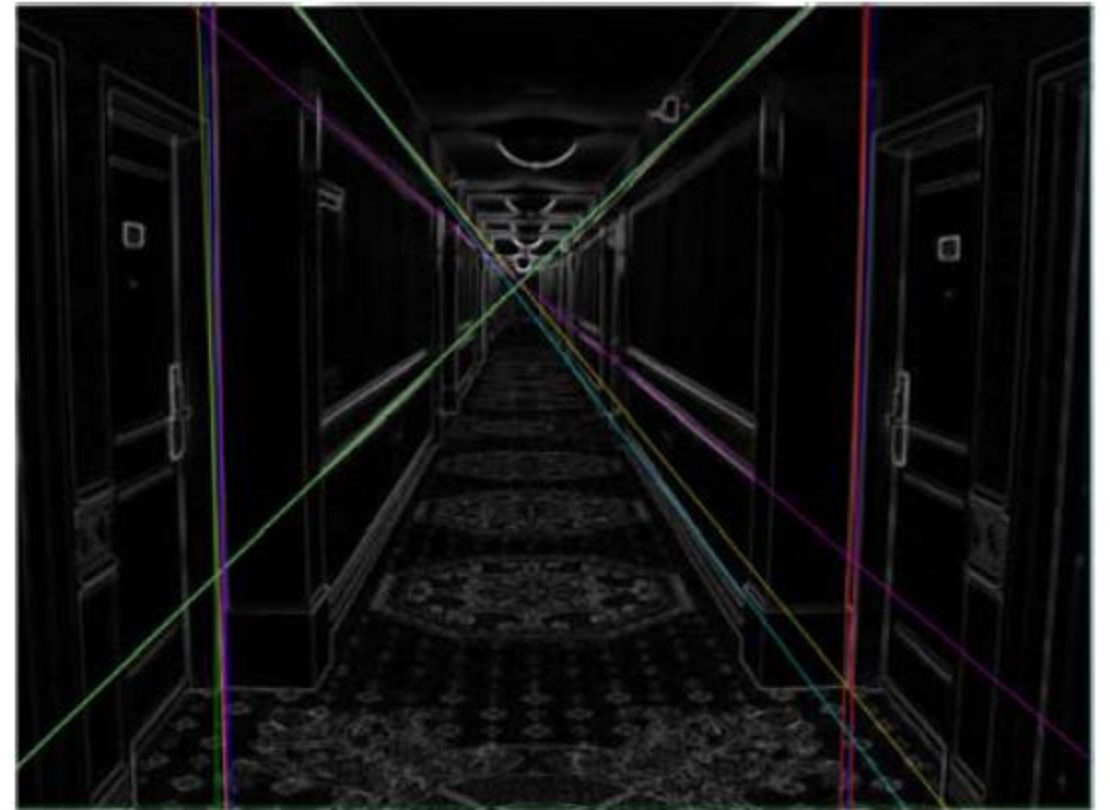
Visualizing the
accumulator space
The height of the
peak will be defined
by the number of
pixels in the line.

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.

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

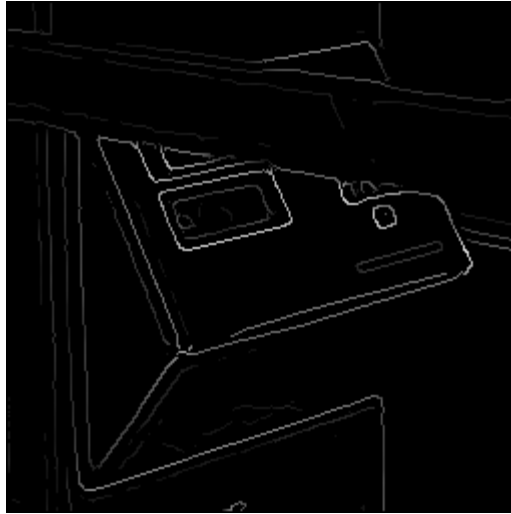


Sumber: INF 4300 – Hough transform, Anne Solberg

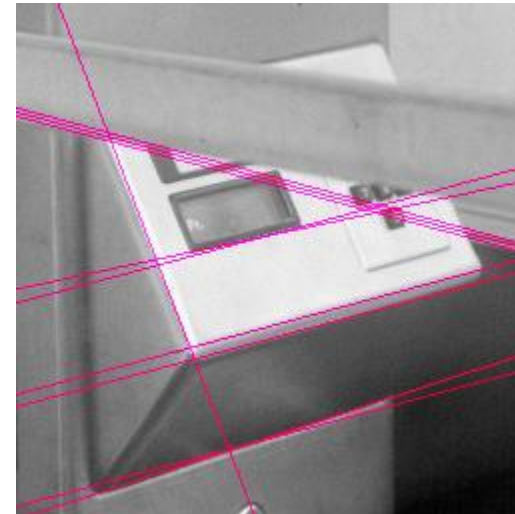
Real World Example



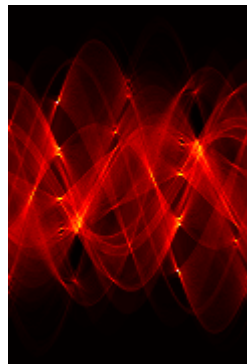
Original



Edge
Detection



Found Lines



Parameter Space

Sumber: *Computer Vision*, S. Narasimhan

Fungsi Hough di dalam Matlab

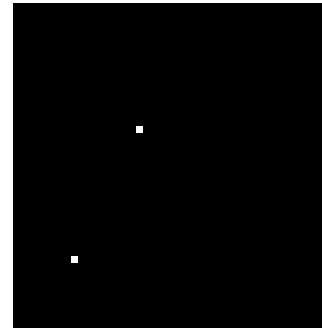
- Matlab menyediakan fungsi `hough()` untuk menghitung matriks hough.
- Fungsi ini menerima input citra tepi yang berupa citra biner. Fungsi `hough()` menghasilkan tiga keluaran yaitu matriks transform (H), theta(T), dan rho(R).
- **Contoh 1:** Misalkan terdapat citra yang hanya berisi dua titik saja sebagai berikut



Output:

```
close all
clear
clc
% 1. Image with two dots
a = zeros(50);
a(20,20) = 1; a(40,10) = 1;
figure;
subplot(2,2,1)
imshow(a)
title('Image with 2 dots')
```

Image with 2 dots



Sumber: <https://www.section.io/engineering-education/line-detection-using-hough-transform-in-matlab/>

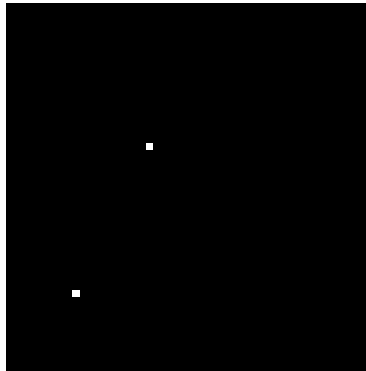
- Lakukan tranformasi Hough

```
[H, T, R] = hough(a);  
subplot(2, 2, 2)  
houghMatViz(H, T, R)           % Visualisasikan output dengan fungsi houghMatViz
```

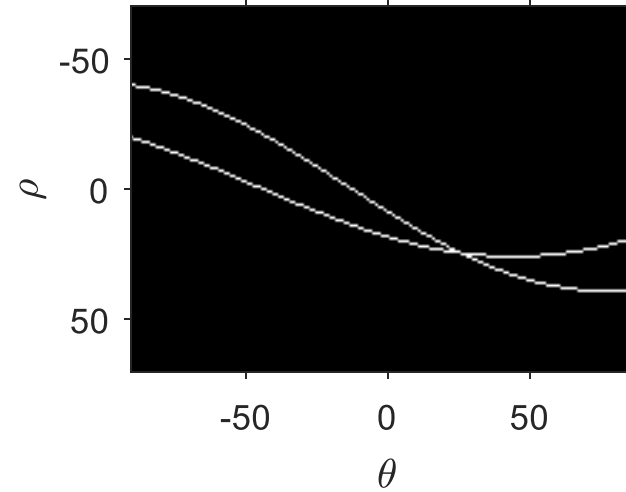
```
function houghMatViz(H,T,R)  
Hgray = mat2gray(H); % Converting hough matrix(H) into grayscale image.  
Hgray = imadjust(Hgray); % Enhancing the brightness  
  
imshow(Hgray, 'XData',T, 'YData',R);  
hold on  
  
axis on % turning on the axis.  
axis normal  
colormap(hot);title('Hough Transform'); % Giving colormap to the image.  
  
xlabel('\theta') % add x-y labels  
ylabel('\rho');
```

Output: dua sinusoida yang berkoresponden dengan dua titik pada citra

Image with 2 dots



Hough Transform



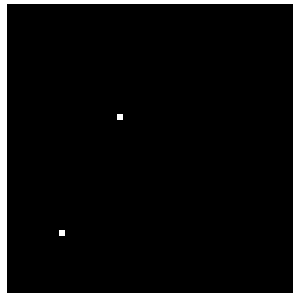
- Gunakan fungsi `houghpeaks()` untuk menemukan satu atau lebih hasil pemungutan suara terbanyak (default = 1).

```
hPeaks = houghpeaks(H);
```

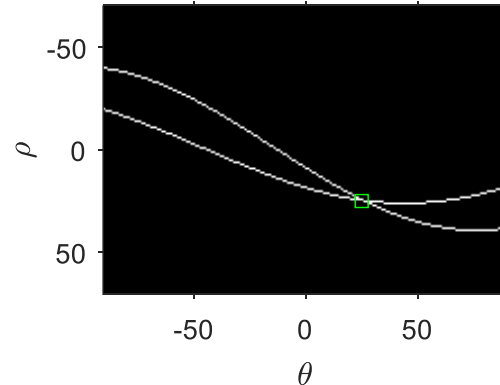
```
x = T(hPeaks(:,2)); %indexing the theta(T) array  
y = R(hPeaks(:,1)); %Indexing the rho(R) array
```

Output:

Image with 2 dots

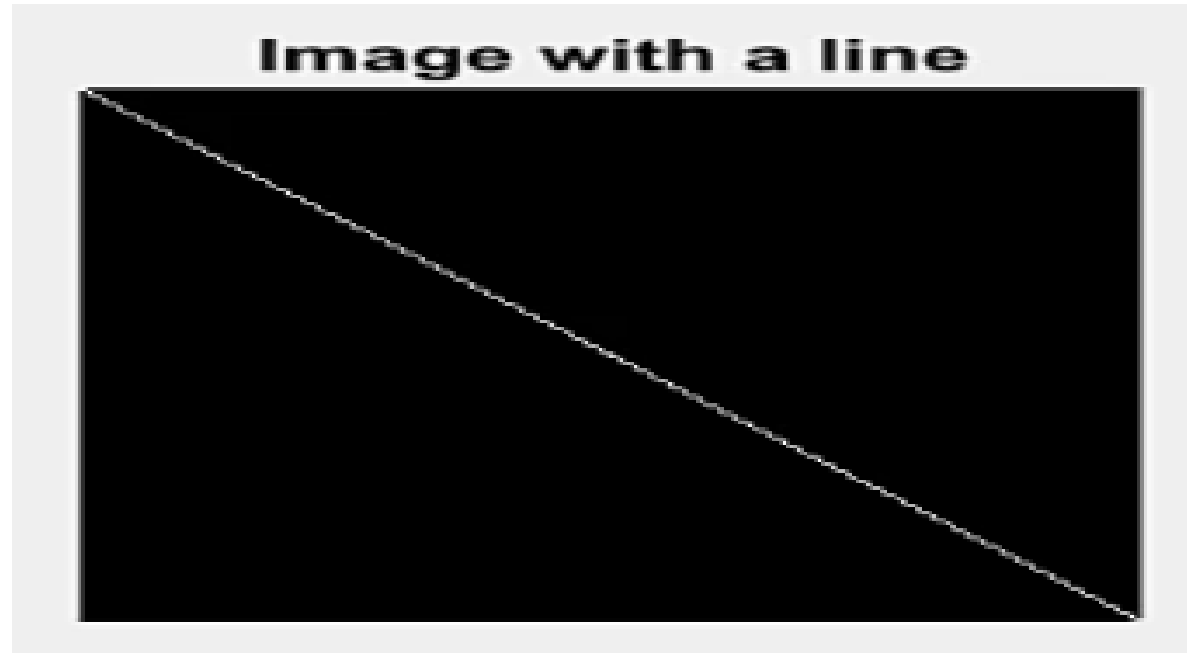


Hough Transform



Bujursangkar kecil berwarna hijau adalah lokasi penumpukan suara terbanyak (*peak location*). Lokasi suara terbanyak menyatakan parameter persamaan garis yang melalui dua titik di dalam citra tersebut

- **Contoh 2:** Misalkan terdapat citra yang berisi menampilkan sebuah garis lurus



Program Matlab untuk mendeteksi garis lurus di dalam citra di atas adalah pada halaman berikut ini

```

close all
clear
clc
% 2. Image with a line
b = eye(150);
subplot(2,2,1)
imshow(b)
title('Image with a line')

[H, T, R] = hough(b);
subplot(2, 2, 2)
houghMatViz(H, T, R)           % Visualisasikn output

hPeaks = houghpeaks(H);

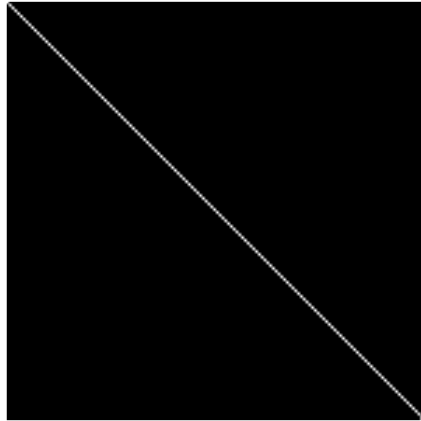
x = T(hPeaks(:,2)); %indexing the theta(T) array
y = R(hPeaks(:,1)); %Indexing the rho(R) array

%Temukan garis-garis dengan fungsi houghlines
hlines = houghlines(b, T, R, hPeaks);

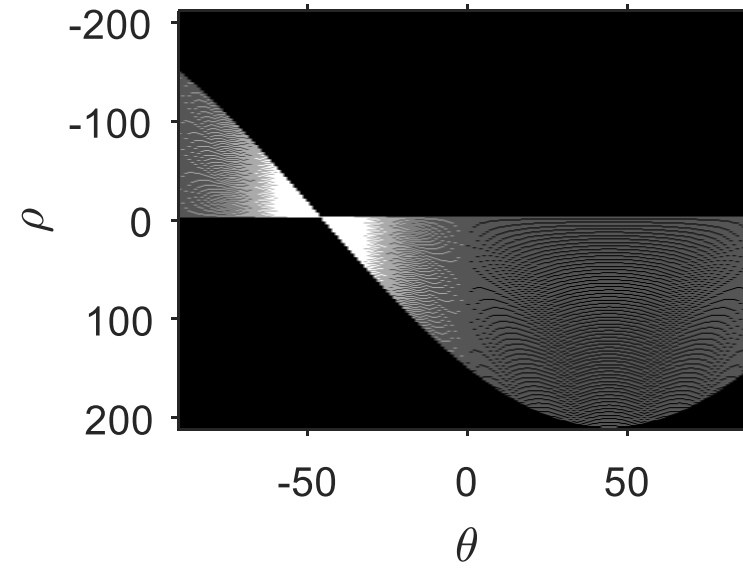
% Tampilkan citra semula dengan garis-garis hasil deteksi Hough
xy = [hlines.point1; hlines.point2];
subplot(2, 2, 3)
imshow(b)
hold on
plot(xy(:,1), xy(:,2), 'g--', 'Linewidth', 5)
title('Detected line')

```

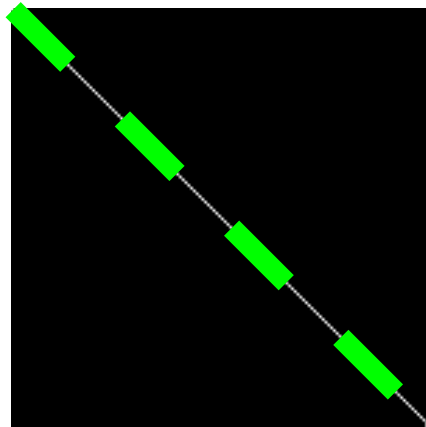
Image with a line



Hough Transform



Detected line

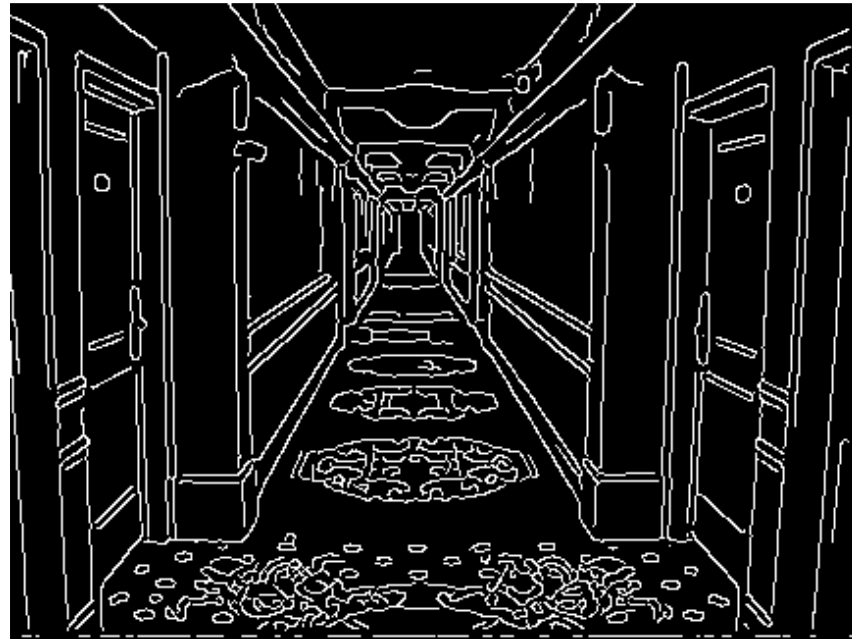


Program 3:

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



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

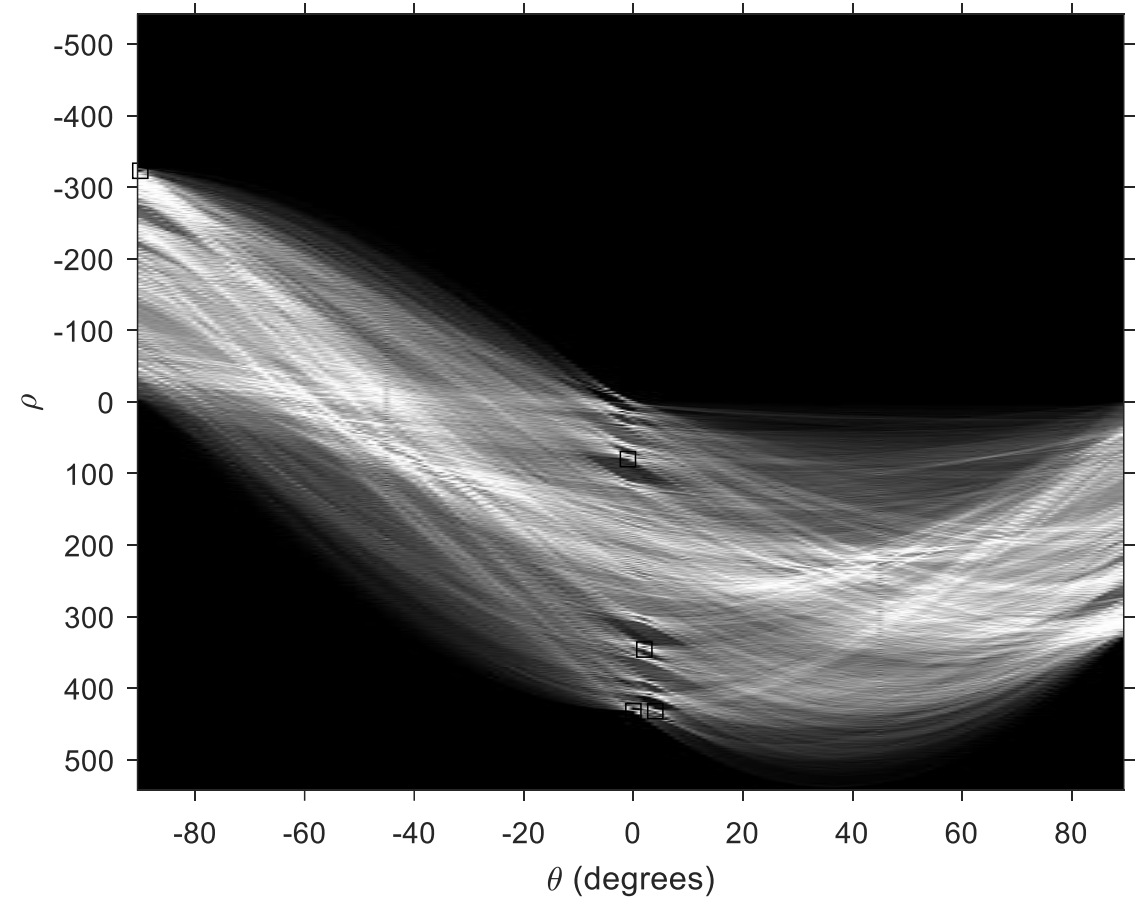


```

% 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');

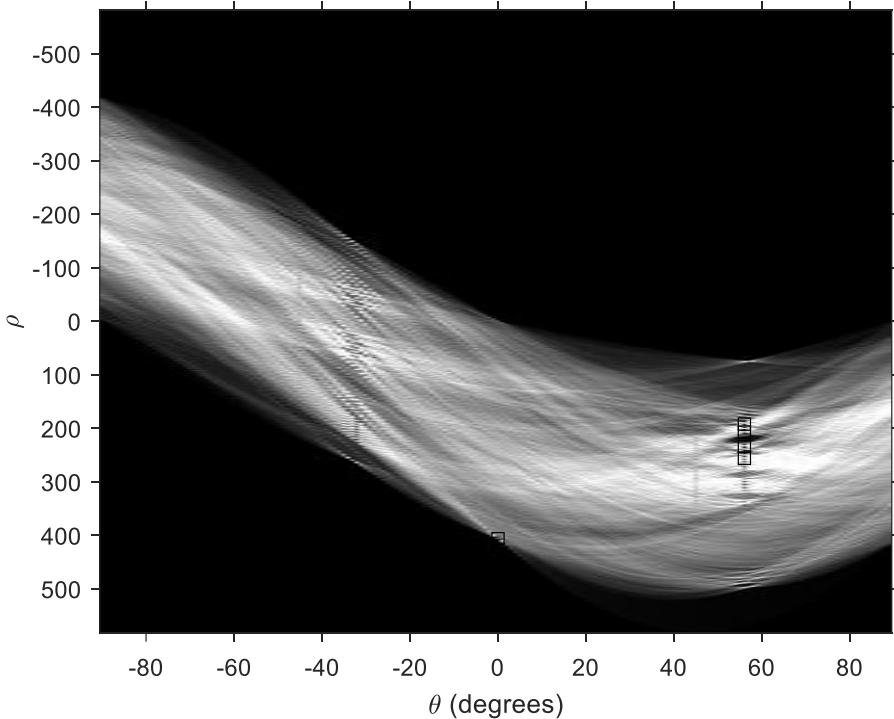
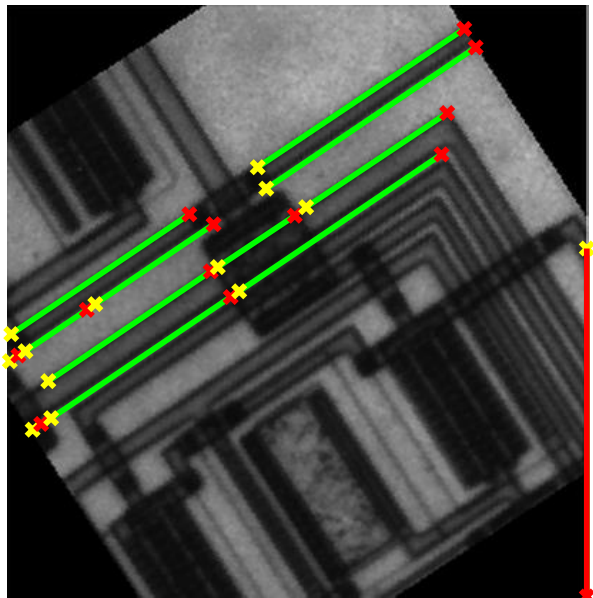
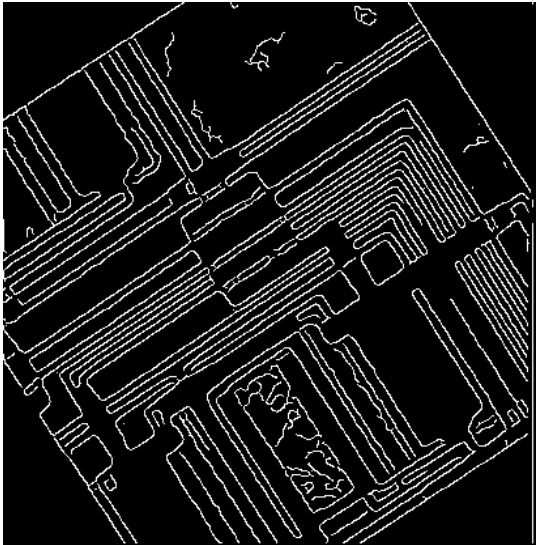
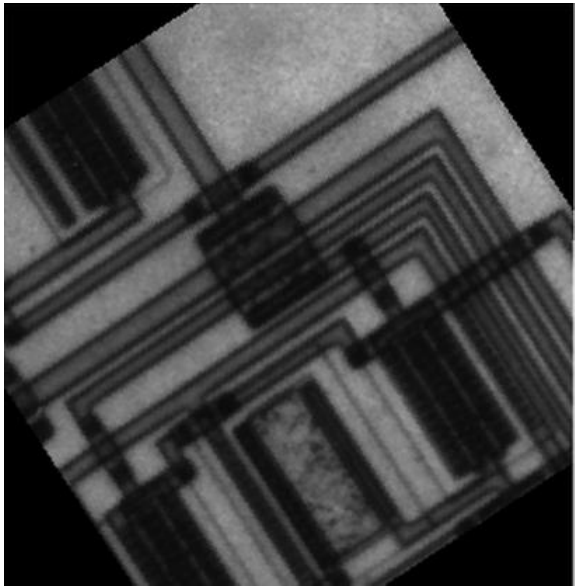
    % Plot beginnings and ends of lines
    plot(xy(1,1),xy(1,2), 'x', 'LineWidth',2, 'Color','yellow');
    plot(xy(2,1),xy(2,2), 'x', 'LineWidth',2, 'Color','red');

    % 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:



B. Mendeteksi lingkaran

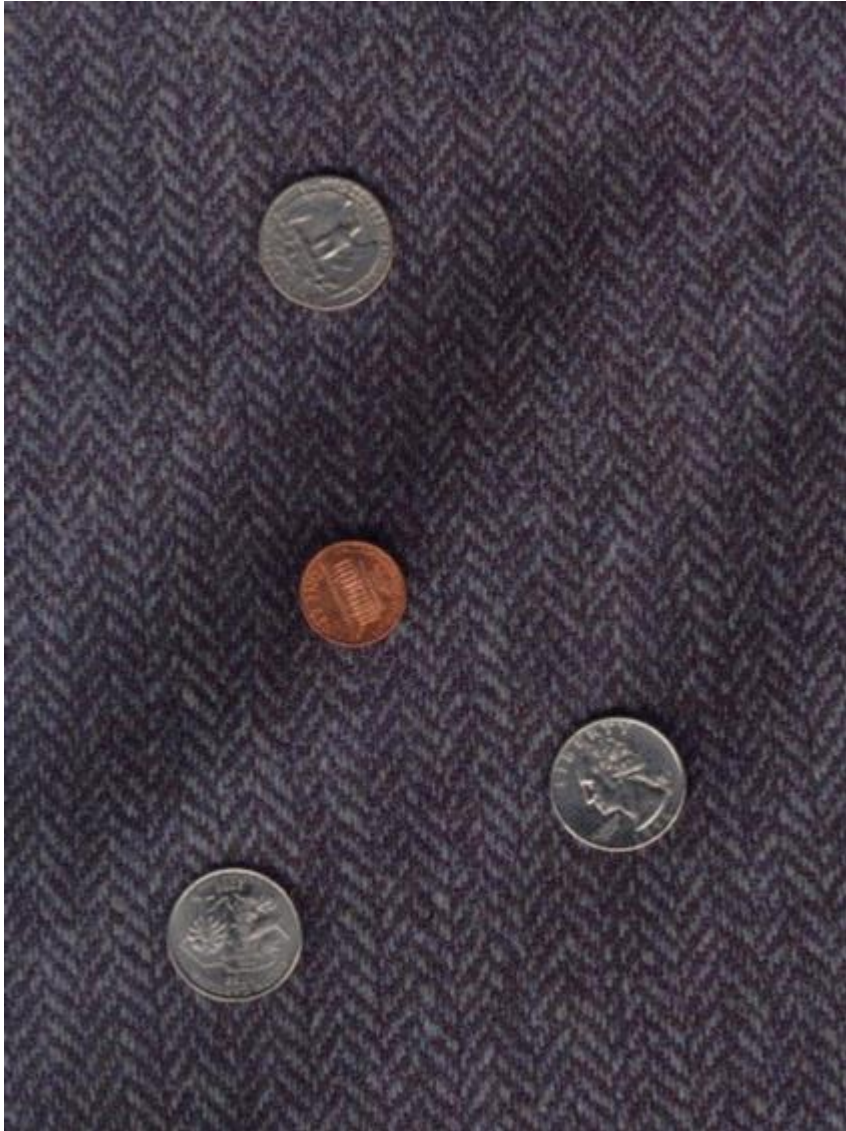
Contoh lingkaran dalam dunia nyata:



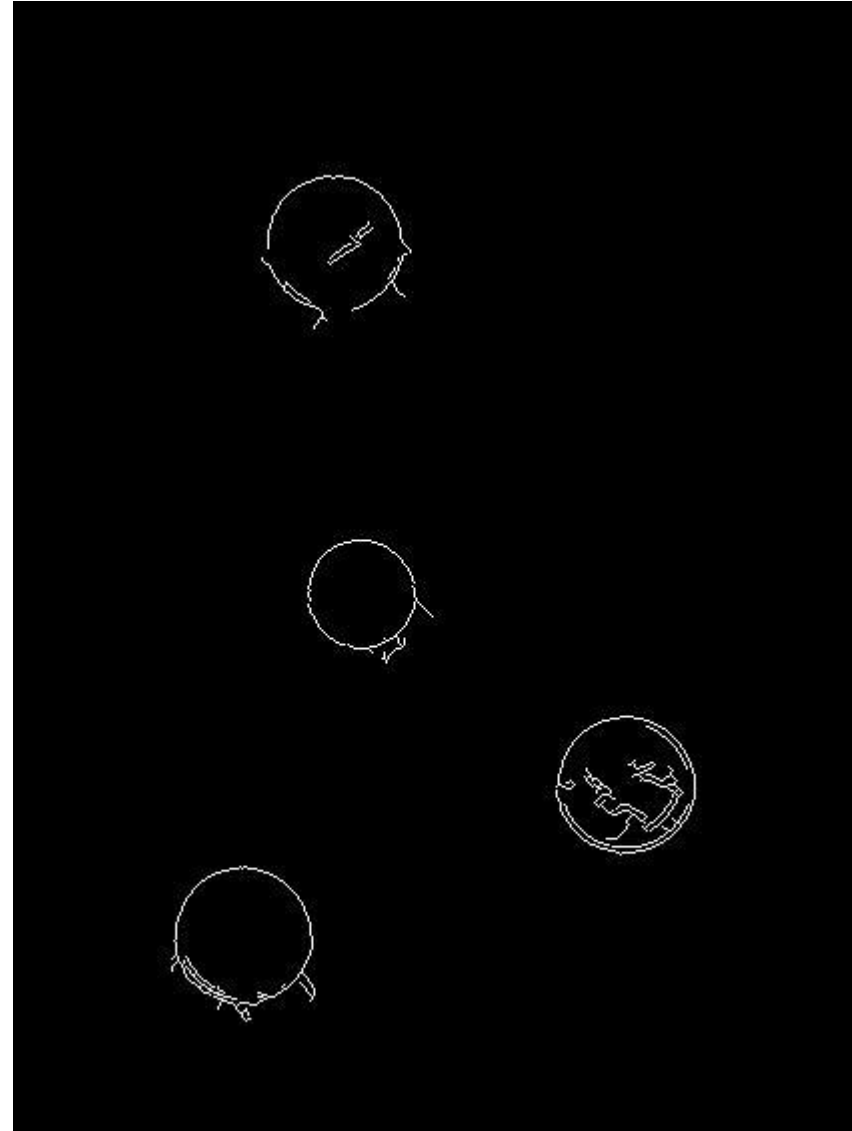
Finding Coins

Sumber: *Computer Vision*, S. Narasimhan

Original



Edges (note noise)



B. Transformasi Hough untuk 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

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

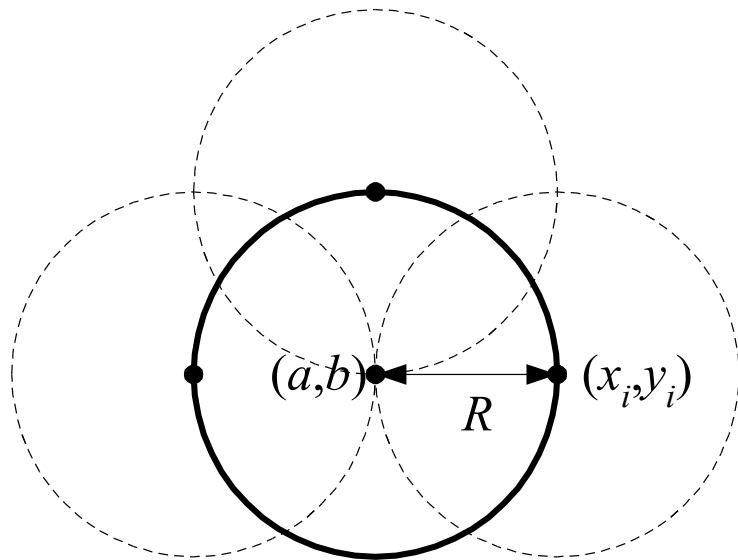
$$b = y - r \sin \theta \quad (7)$$

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

- Misalkan (x_i, y_i) 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 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.

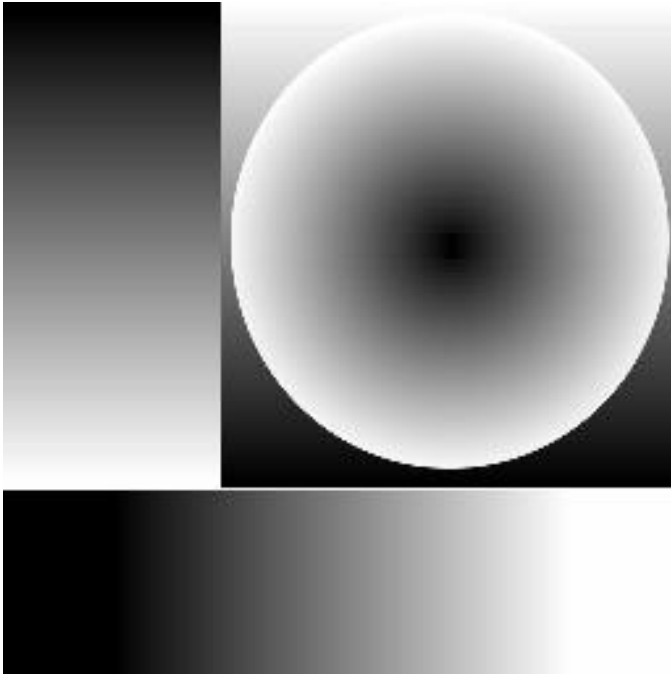
- Persamaan (8) dan (9) dapat dimanipulasi dengan mengeliminasi r dari kedua persamaan:

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

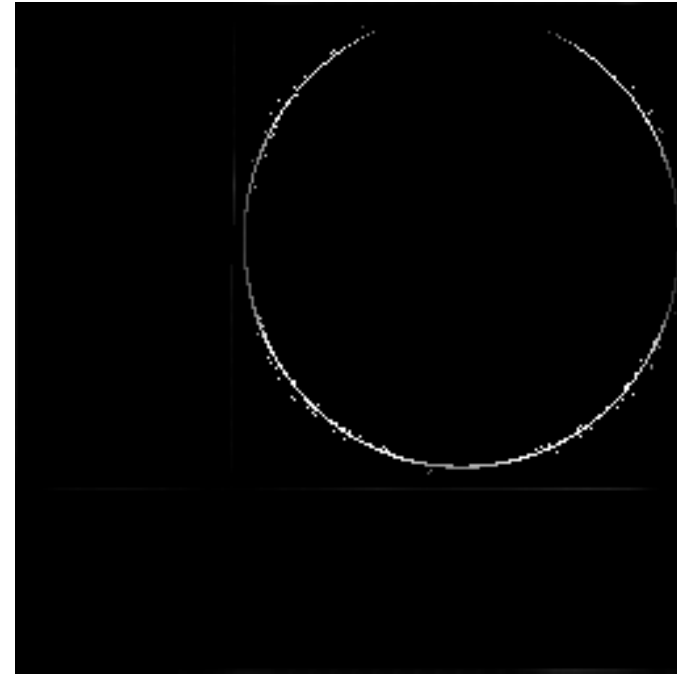
$$b = y - r \sin \theta \rightarrow b = y - \frac{(x - a)}{\cos \theta} \sin \theta = y - (x - a) \tan \theta$$

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

- Dengan demikian, maka ruang parametrik trimatra, $P(r,a,b)$, dapat direduksi menjadi ruang dwimatra, $P(a,b)$.
- 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



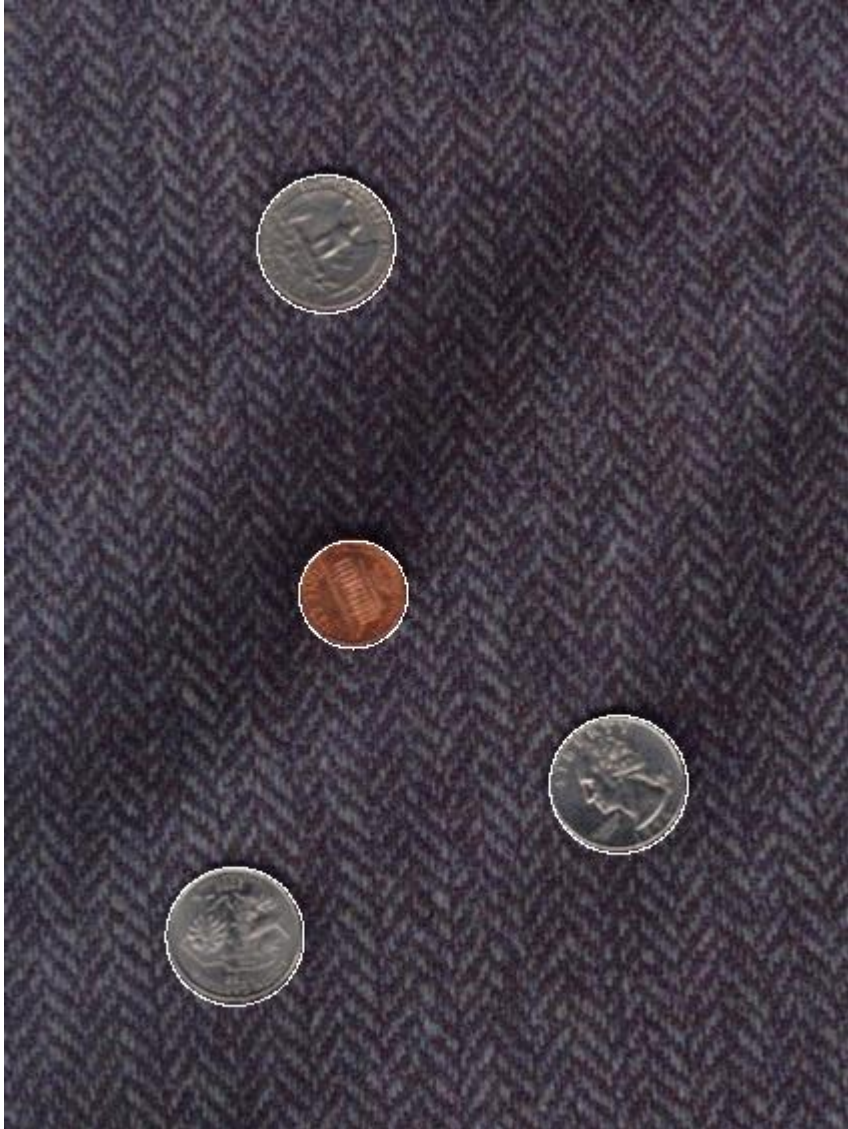
Citra slope



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 $\mathbf{x}$ adalah vektor peubah dan $\mathbf{a}$ 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

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

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

