PRAKTIKUM 10

Hough Transform

Materi

- Contoh Implementasi image segmentation (hough transform) di OpenCV

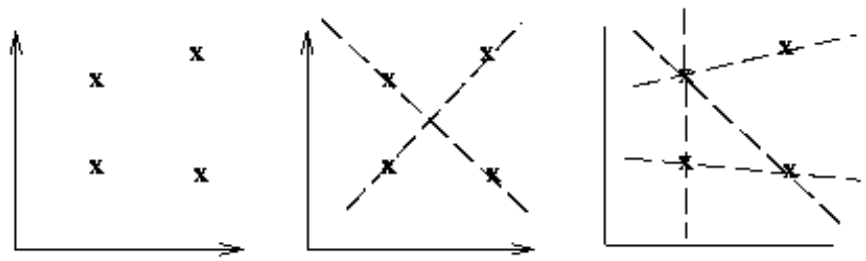
Tujuan Praktikum

- Mahasiswa dapat mengimplementasikan hough transform di OpenCV.

A. Penyajian

Hough Transform

Konsep dasar dari *Hough transform* adalah bahwa dalam gambar terdapat garis dan kurva dengan berbagai ukuran dan orientasi yang melalui titik mana saja. Tujuan transformasi adalah untuk menemukan persamaan yang paling banyak melalui titik.



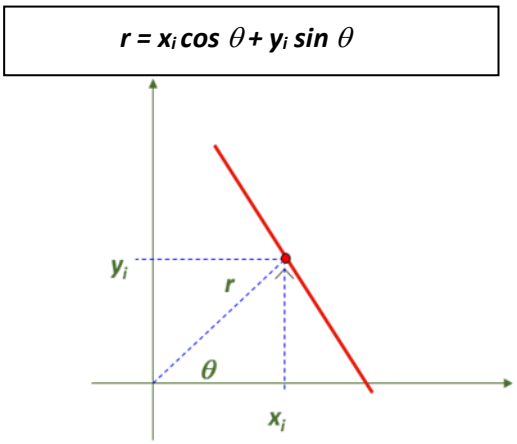
Gambar 10.1. Hough transform

Ada beberapa jenis Hough Tranform, yaitu :

1. Hough Transform Line
2. Hough Transform Circle

Hough Transform Line (r, θ)

- Dikembangkan oleh Richard Duda dan Peter Hart, 1972
- ***Standard Hough Transform* (*SHT*)**

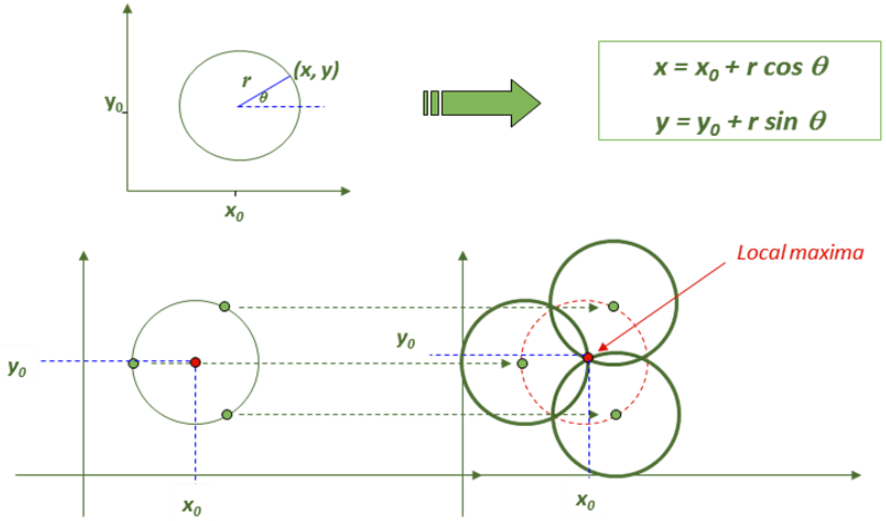


Gambar 10.2 Hough transform garis

Fungsi Hough Transform Line pada OpenCV:
HoughLines(InputArray **image**, OutputArray **lines**, double **rho**, double **theta**, int **threshold**, double **srn**=0, double **stn**=0)

Parameternya
image – 8-bit, single-channel binary source image. The image may be modified by the function.
lines – Output vector of lines. Each line is represented by a two-element vector (ρ, θ). ρ is the distance from the coordinate origin $(0, 0)$ (top-left corner of the image). θ is the line rotation angle in radians ($0 \sim \text{vertical line}, \pi/2 \sim \text{horizontal line}$).
rho – Distance resolution of the accumulator in pixels.
theta – Angle resolution of the accumulator in radians.
threshold – Accumulator threshold parameter. Only those lines are returned that get enough votes ($> \texttt{threshold}$).
srn – For the multi-scale Hough transform, it is a divisor for the distance resolution rho . The coarse accumulator distance resolution is rho and the accurate accumulator resolution is rho/srn . If both srn=0 and stn=0 , the classical Hough transform is used. Otherwise, both these parameters should be positive.
stn – For the multi-scale Hough transform, it is a divisor for the distance resolution theta.

***Hough Transform Circle* (x_0, y_0, r):**



Gambar 10.3 Hough transform lingkaran

Fungsi Hough Transform Circle di openCV:

HoughCircles(InputArray **image**, OutputArray **circles**, int **method**, double **dp**, double **minDist**, double **param1**=100, double **param2**=100, int **minRadius**=0, int **maxRadius**=0)

Keterangan

- **image** – 8-bit, single-channel, grayscale input image.
- **circles** – Output vector of found circles. Each vector is encoded as a 3-element floating-point vector $(x, y, radius)$.
- **circle_storage** – In C function this is a memory storage that will contain the output sequence of found circles.
- **method** – Detection method to use. Currently, the only implemented method is CV_HOUGH_GRADIENT , which is basically *21HT* , described in.
- **dp** – Inverse ratio of the accumulator resolution to the image resolution. For example, if dp=1 , the accumulator has the same resolution as the input image. If dp=2 , the accumulator has half as big width and height.
- **minDist** – Minimum distance between the centers of the detected circles. If the parameter is too small, multiple neighbor circles may be falsely detected in addition to a true one. If it is too large, some circles may be missed.
- **param1** – First method-specific parameter. In case of CV_HOUGH_GRADIENT , it is the higher threshold of the two passed to the Canny() edge detector (the lower one is twice smaller).
- **param2** – Second method-specific parameter. In case of CV_HOUGH_GRADIENT , it is the accumulator threshold for the circle centers at the detection stage. The smaller it is, the more false circles may be detected. Circles, corresponding to the larger accumulator values, will be returned first.

- **minRadius** – Minimum circle radius.
- **maxRadius** – Maximum circle radius.

A LATIHAN

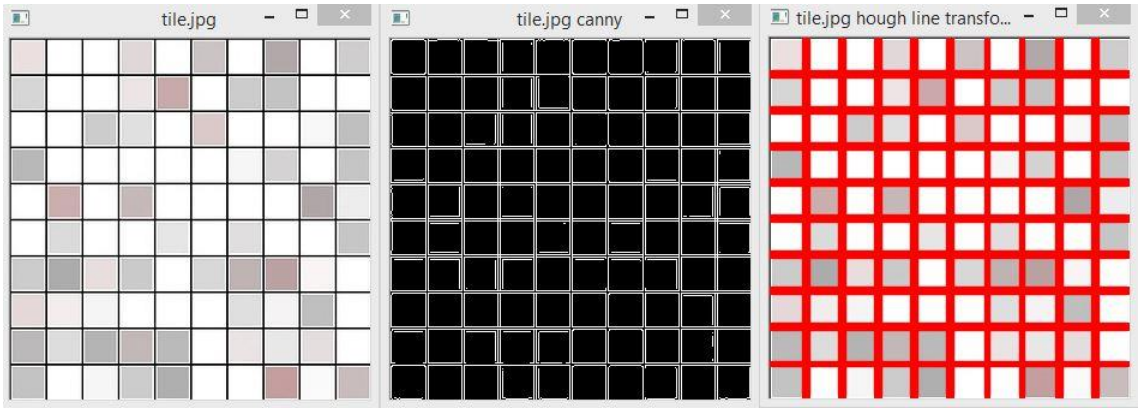
Hough line transform

Berikut ini merupakan latihan membuat program yang dapat mendeteksi garis pada citra menggunakan *hough line transform*. Citra tile.jpg tersedia di LMS pada LKP 11.

```
#include <cv.h>
#include <highgui.h>
#include <iostream>

using namespace cv;
using namespace std;

void detectLine(Mat &src, Mat &dst, string filename)
{
    Mat cannyDst;
    Mat graySrc;

    if( src.channels() > 1) cvtColor(src, graySrc, CV_BGR2GRAY);
    else graySrc = src.clone();
    //apply canny to edge detection
    Canny(graySrc, cannyDst, 50, 100, 3);
    //detect line
    vector<Vec2f> lines;
    HoughLines(cannyDst, lines, 1, CV_PI/180, 255, 0, 0 );
    //draw line to src image
    cout << filename  << " line detected = " << lines.size() << endl;
    for( size_t i = 0; i < lines.size(); i++ )
    {
        float rho = lines[i][0], theta = lines[i][1];
        Point pt1, pt2;
        double a = cos(theta), b = sin(theta);
        double x0 = a*rho, y0 = b*rho;
        pt1.x = cvRound(x0 + 1000*(-b));
        pt1.y = cvRound(y0 + 1000*(a));
        pt2.x = cvRound(x0 - 1000*(-b));
        pt2.y = cvRound(y0 - 1000*(a));
        line( dst, pt1, pt2, Scalar(0,0,255), 3, CV_AA);
    }
    imshow(filename + " canny", cannyDst);
}

int main()
{
    /* original source
   //tile.jpg source: https://pixabay.com/en/banner-header-tiles-grey-
1763839/
   */
```

```
    string filename = "tile.jpg";
    Mat img = imread(filename);
    Mat dst = img.clone();
    detectLine(img, dst, filename);
    imshow(filename, img);
    imshow(filename+" hough line transformation", dst);

    waitKey(0);
    return 0;
}
```

Berikut ini merupakan hasil *hough line transform* setelah dijalankan. Hasil *hough line transform.*



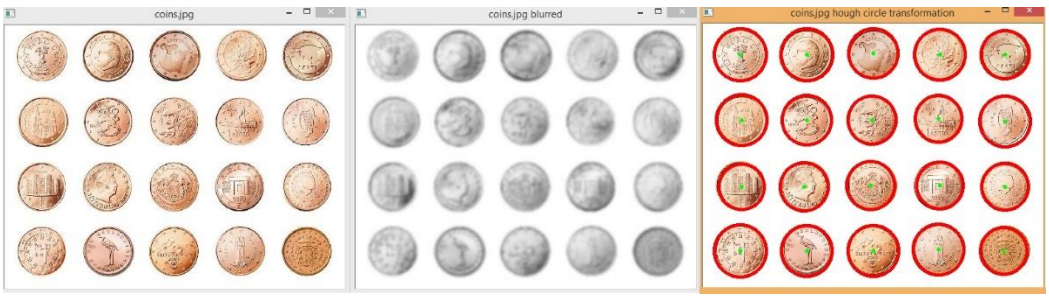
Hough circle transform

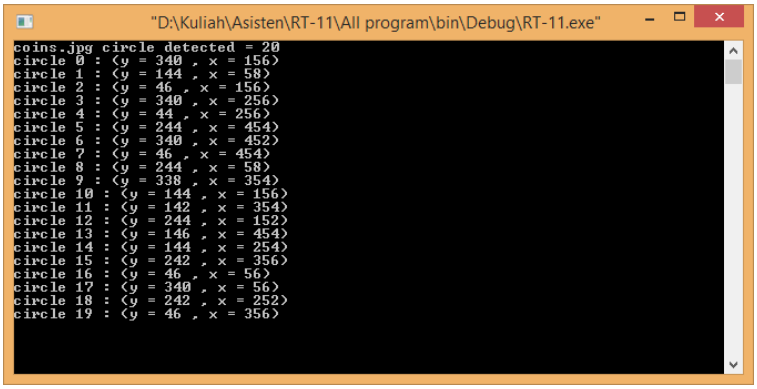
Berikut ini merupakan latihan membuat program yang dapat mendeteksi dan menghitung jumlah lingkaran menggunakan *hough circle transform*. Citra coins.jpg tersedia di LMS pada LKP 11.

```
#include <cv.h>
#include <highgui.h>
#include <iostream>

using namespace cv;
using namespace std;

void detectCircle(Mat &src, Mat &dst, string filename)
{
    Mat graySrc;
    dst = src.clone();

    if( src.channels() > 1) cvtColor(src, graySrc, CV_BGR2GRAY);
    else graySrc = src.clone();

    //Reduce the noise so we avoid false circle detection
    GaussianBlur( graySrc, graySrc, Size(9, 9), 2, 2 );

    vector<Vec3f> circles;
    // Apply the Hough Transform to find the circles
    HoughCircles( graySrc, circles, CV_HOUGH_GRADIENT, 1, graySrc.rows/8,
    125, 55, 0, 0 );
```

```
    // Draw the circles detected
    cout << filename  << " circle detected = " << circles.size() << endl;
    for( size_t i = 0; i < circles.size(); i++ )
    {
        Point center(cvRound(circles[i][0]), cvRound(circles[i][1]));
        int radius = cvRound(circles[i][2]);
        //print circle location
        cout << "circle " << i << " : " << "(y = " <<
          cvRound(circles[i][1]) << " , x = " << cvRound(circles[i][0])
          << ")" << endl;
        // circle center
        circle( dst, center, 3, Scalar(0,255,0), -1, 8, 0 );
        // circle outline
        circle( dst, center, radius, Scalar(0,0,255), 3, 8, 0 );
    }
    imshow(filename+" blurred", graySrc);
}

int main()
{
    /* original source
   coins,jpg source: https://pixabay.com/en/cent-1-coin-currency-europe-
money-400275/
    */
    string filename = "coins.jpg";
    Mat img = imread(filename);
    Mat dst;
    detectCircle(img, dst, filename);
    imshow(filename, img);
    imshow(filename+" hough circle transformation", dst);

    waitKey(0);
    return 0;
}
```

Berikut ini merupakan hasil *hough circle transform* setelah dijalankan. Hasil *hough circle transform.*




"D:\Kuliah\Asisten\RT-11\All program\bin\Debug\RT-11.exe"
coins.jpg circle detected = 20
circle 0 : (y = 340 , x = 156)
circle 1 : (y = 144 , x = 58)
circle 2 : (y = 46 , x = 156)
circle 3 : (y = 340 , x = 256)
circle 4 : (y = 44 , x = 256)
circle 5 : (y = 244 , x = 454)
circle 6 : (y = 340 , x = 452)
circle 7 : (y = 46 , x = 454)
circle 8 : (y = 244 , x = 58)
circle 9 : (y = 338 , x = 354)
circle 10 : (y = 144 , x = 156)
circle 11 : (y = 142 , x = 354)
circle 12 : (y = 244 , x = 152)
circle 13 : (y = 146 , x = 454)
circle 14 : (y = 144 , x = 254)
circle 15 : (y = 242 , x = 356)
circle 16 : (y = 46 , x = 56)
circle 17 : (y = 340 , x = 56)
circle 18 : (y = 242 , x = 252)
circle 19 : (y = 46 , x = 356)

Nama :

NRP :
Nama Dosen :
Nama Asisten :

B TUGAS

Download citra cube.jpg dan circles.jpg pada LMS. Lakukan *hough line transform* terhadap citra cube.jpg dan lakukan *hough circle transform* terhadap citra circles.jpg. Tuliskan hasil analisis terhadap fungsi *hough line transform* dan *hough circle transform. File* disimpan dengan format LKP11_NIM_Kelas dalam file .pdf.

Hough line transform

Berikut ini merupakan fungsi *hough line transform* yang terdapat pada OpenCV

```
HoughLines(dst, lines, rho, theta, threshold, srn, stn );
```

Jelaskanlah pengaruh *parameter* dari fungsi HoughLines

1 Pengaruh rho
2 Pengaruh theta
3 Pengaruh threshold
4 Pengaruh srn
5 Pengaruh stn

Hough circle transform

Berikut ini merupakan fungsi *hough circle transform* yang terdapat pada OpenCV

```
HoughCircles
(
   src_gray, circles, CV_HOUGH_GRADIENT,
   dp, min_dist, param_1, param_2,
   min_radius, max_radius
);
```

Jelaskanlah pengaruh *parameter* dari fungsi HoughCircles

1 Pengaruh dp
2 Pengaruh min_dist
3 Pengaruh param_1
4 Pengaruh param_2
5 Pengaruh min_radius
6 Pengaruh max_radius