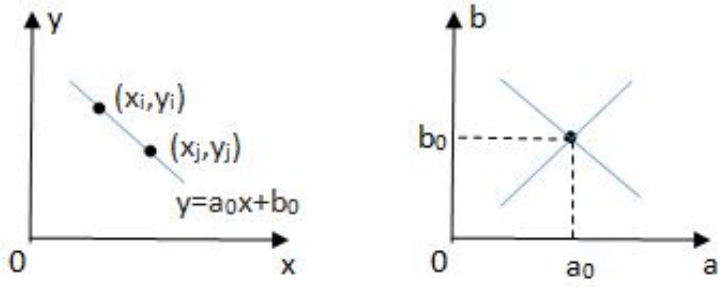
阿洲的程式教學
關於Qt、OpenCV、影像處理演算法

霍夫找線(HoughLines、HoughLinesP)

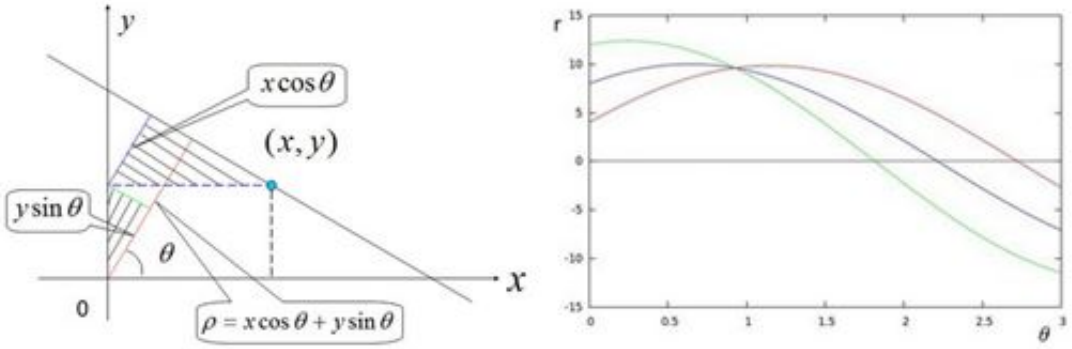
計算機視覺中經常需要識別或者定位某些幾何圖形，像直線、圓、橢圓等，檢測直線的霍夫變換提供在圖像中尋找直線的一種算法，後來這概念發展到能檢測圓、橢圓等，不僅能夠識別出圖像中想要的圖形，而且能夠得到位置、角度等資訊，這邊解釋霍夫直線偵測的原理。

核心思想是把圖像中某個點集映射到另一空間的一個點集上，這個點記錄了點集合的數目，通過搜索峰值來決定線。像我們常用$y_i=ax_i+b$表達空間中通過點(x_i,y_i)的一條線，a和b決定了通過此點的線，假設空間有另一點(x_j,y_j)，我們以$y_j = a'x_j+b'$得到a'和b'，a'和b'決定了通過此點的線。

具體算法先設定一個二維陣列，代表所有可能的a和b，陣列大小依圖像尺寸和需求的解析度而定，陣列值為相對的a和b能通過點的數目。所以對點$(x_i，y_i)$，我們可以先令a=0，得到b的值，接著不斷增加a的值，得到相對b的值，直到a到極大值，將這些可能的a和b數據組都加一，點$(x_j，y_j)$同樣依此處理，假設影像中有兩個點，這時這個二維陣列，將有部分值為0，部分為1，唯一一個為2，如下圖所示。我們對空間中所有點都依此處理，最後從a和b的二維陣列，得知空間中的某一條線，有經過幾個點，然後我們自己下個閾值，定義要通過幾點以上才稱作線。



但是這種一般式，可能會遇到斜率為0或無窮大的情況，造成計算上的麻煩，所以習慣上都轉換成極座標，用極座標來表達空間中的一條線，轉換的公式為$r=x\cos\theta+y\sin\theta$，由於轉換的方式，轉換空間會從上述的直線變成正弦曲線，但同樣可得到通過此點的所有r和θ，透過r和θ這個2維陣列，得知空間中的某一條線，總共通過幾個點。



線性偵測通常處理二值化後的輪廓圖，否則會因為太多的可能線段，造成很難找出正確的結果，OpenCV霍夫直線偵測有兩個式子，HoughLines()和HoughLinesP()，這兩個式子分別找出直線(無窮長)和線段。

OpenCV 直線偵測

void HoughLines(InputArray image, OutputArray lines, double rho, double theta, int threshold, double srn=0, double stn=0)

- image：輸入圖，8位元單通道二值化圖。
- lines：將所有線的資料存在vector< Vec2f >，Vec2f為每個線的資料，分別有ρ、θ這兩個參數，ρ表示和左上角(0,0)的距離，θ是線的旋轉角度，單位弧度，垂直線的θ為0，水平線的θ為π/2。
- rho：距離解析度，越小表示定位要求越準確，但也較易造成應該是同條線的點判為不同線。
- theta：角度解析度，越小表示角度要求越準確，但也較易造成應該是同條線的點判為不同線。
- threshold：累積個數閾值，超過此值的線才會存在lines這個容器內。
- srn：可有可無的距離除數。
- stn：可有可無的角度除數。

OpenCV 直線偵測

void HoughLinesP(InputArray image, OutputArray lines, double rho, double theta, int threshold, double minLineLength=0, double maxLineGap=0)

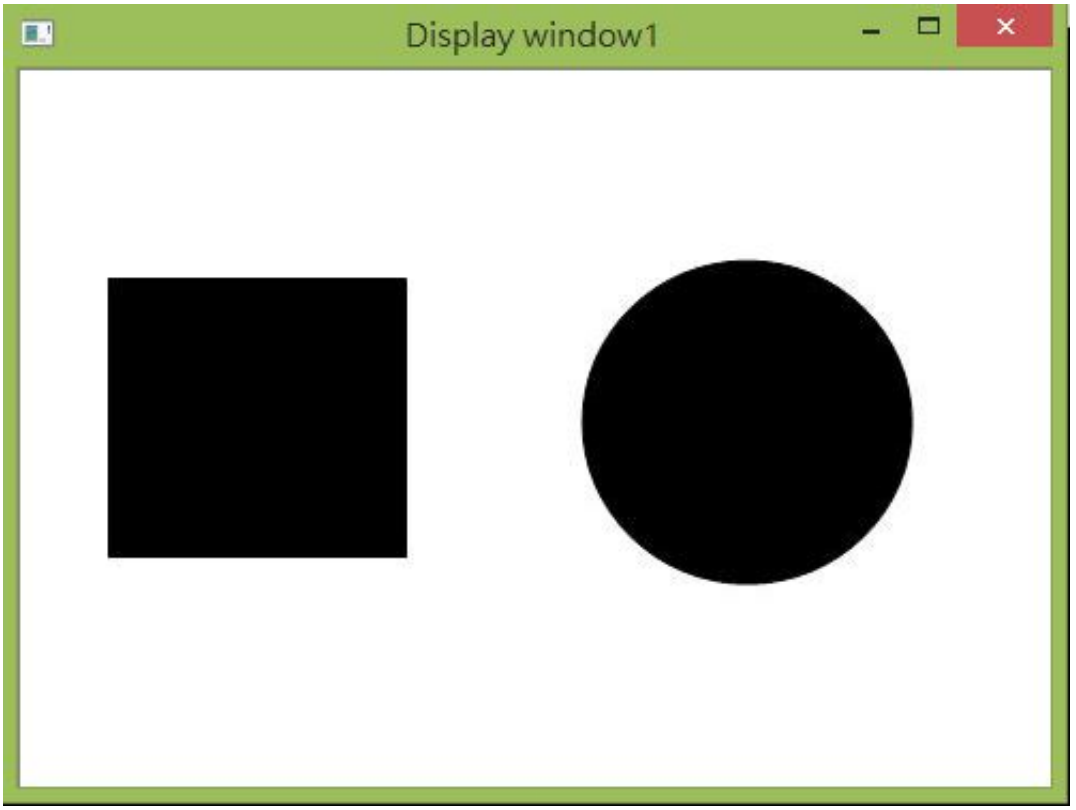
- image：輸入圖，8位元單通道二值化圖。
- lines：將所有線的資料存在vector< Vec4i >，Vec4i為每個線段的資料，分別有x_1、y_1、x_2、y_2這四個值，(x_1，y_1)和(x_2，y_2)分別表示線段的頭尾頂點。

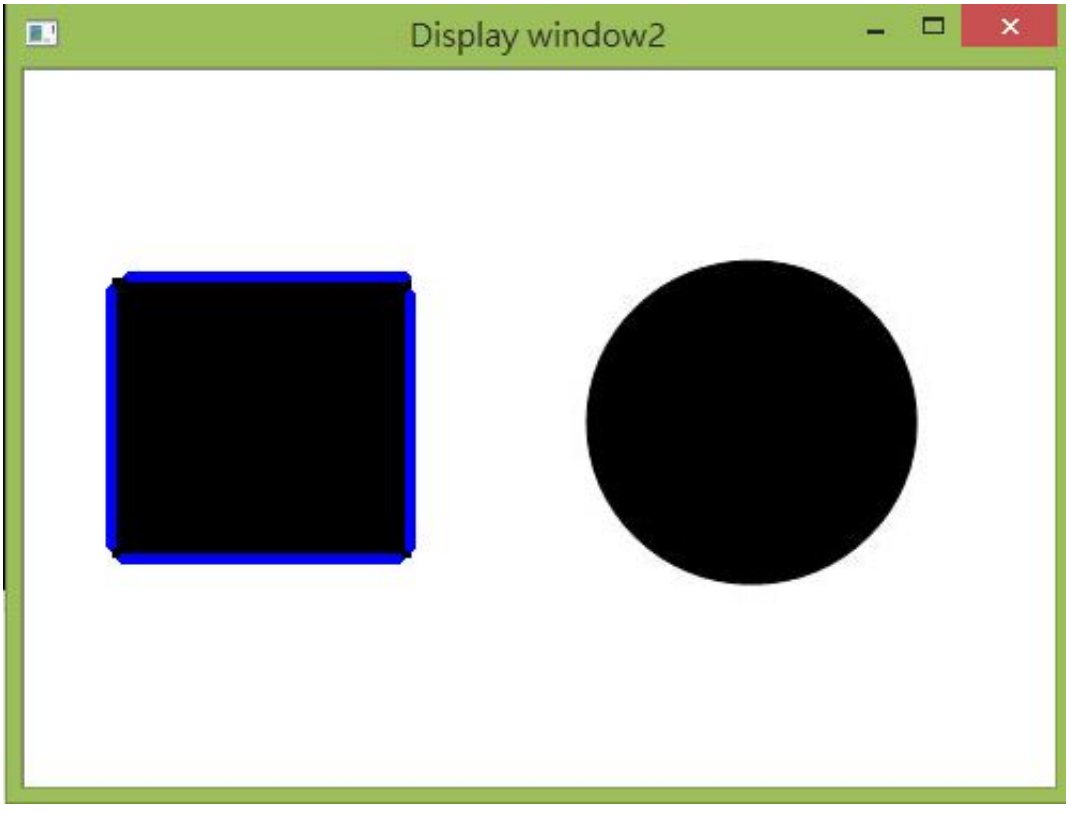
- rho：距離解析度，越小表示定位要求越準確，但也較易造成應該是同條線的點判為不同線。
- theta：角度解析度，越小表示角度要求越準確，但也較易造成應該是同條線的點判為不同線。
- threshold：累積個數閾值，超過此值的線才會存在lines這個容器內。
- minLineLength ：線段最短距離，超過此值的線才會存在lines這個容器內。
- maxLineGap：最大間隔。

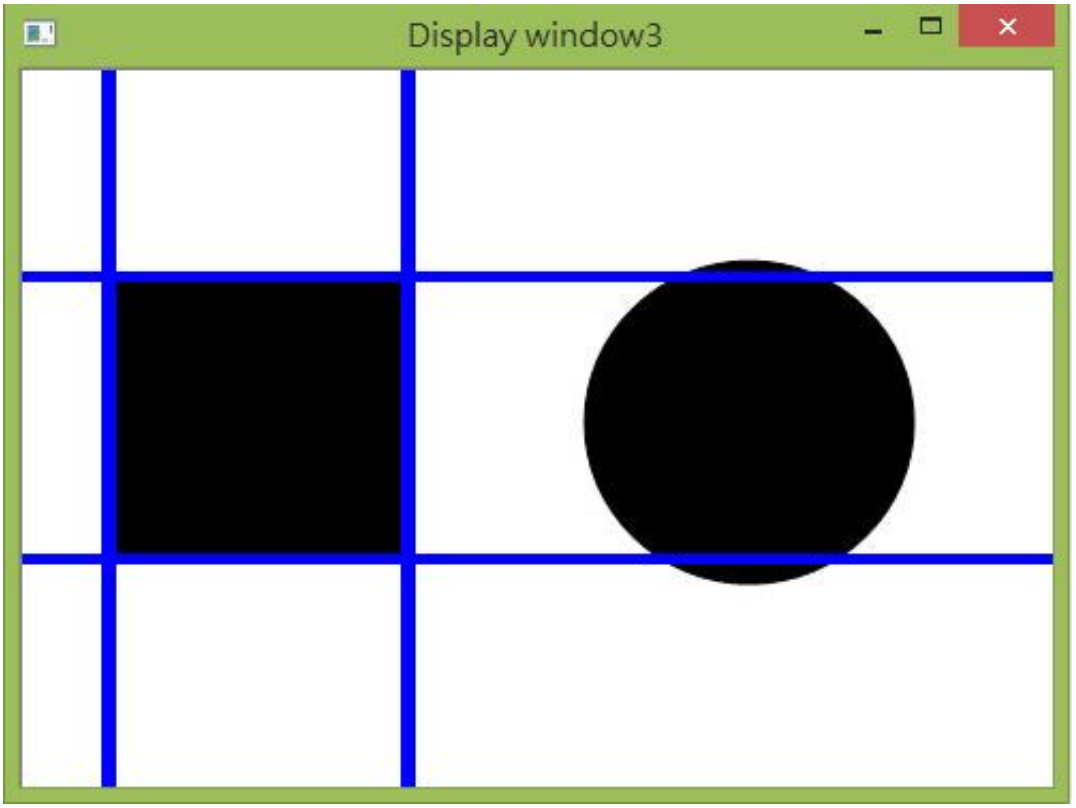
以下範例分別用HoughLines()和HoughLinesP()找出圖中的直線或線段，並用自行撰寫的drawLines()將找到的直線或線段畫出，在HoughLines()中，我們先判斷線是直的或橫的，直的線兩端點會在第一列和最後一列，橫的線兩端點會在第一欄和最後一欄。

```
#include <cstdio>
#include <opencv2/opencv.hpp>
using namespace cv;
#define PI 3.1416

void calcLinesP(const Mat &input, std::vector<Vec4i> &lines);
void drawLinesP(Mat &input, const std::vector<Vec4i> &lines);
void calcLines(const Mat &input, std::vector<Vec2f> &lines);
void drawLines(Mat &input, const std::vector<Vec2f> &lines);

int main(){
    Mat img = imread("test.jpg",CV_LOAD_IMAGE_GRAYSCALE);
    Mat result1 = imread("test.jpg",CV_LOAD_IMAGE_COLOR);
    Mat result2 = imread("test.jpg",CV_LOAD_IMAGE_COLOR);

    vector<Vec4i> linesP;
    calcLinesP(img,linesP);
    drawLinesP(result1, linesP);
    vector<Vec2f> lines;
    calcLines(img,lines);
    drawLines(result2, lines);

    namedWindow("Display window1", WINDOW_AUTOSIZE);
    namedWindow("Display window2", WINDOW_AUTOSIZE);
    namedWindow("Display window3", WINDOW_AUTOSIZE);
    imshow("Display window1", img);
    imshow("Display window2", result1);
    imshow("Display window3", result2);
    waitKey(0);

    return 0;
}

void calcLinesP(const Mat &input, std::vector<Vec4i> &lines){
    Mat contours;
```

```
    Canny(input, contours, 50, 150);
    lines.clear();
    HoughLinesP(contours, lines, 1, CV_PI/180, 50);
}

void calcLines(const Mat &input, std::vector<Vec2f> &lines){
    Mat contours;
    Canny(input,contours,50,150);
    lines.clear();
    HoughLines(contours, lines, 1, CV_PI/180, 50);
}

void drawLinesP(Mat &input, const std::vector<Vec4i> &lines){
    for(int i=0; i<lines.size(); i++){
        line(input, Point(lines[i][0], lines[i][3]), Point(lines[i][4],
lines[i][5]), Scalar(255,0,0), 3);
    }
}

void drawLines(Mat &input, const std::vector<Vec2f> &lines){
    for(int i=0; i<lines.size(); i++){
        float r = lines[i][0];
        float theta = lines[i][6];
        if(theta<PI/4.0 || theta>3*PI/4.0){
            Point pt1(r/cos(theta),0);
            Point pt2((r-input.rows*sin(theta))/cos(theta), input.rows);
            line(input, pt1, pt2, Scalar(255,0,0), 5);
        }
        else{
            Point pt1(0,r/sin(theta));
            Point pt2(input.cols, (r-input.cols*cos(theta))/sin(theta));
            line(input, pt1, pt2, Scalar(255,0,0), 3);
        }
    }
}
```


Display window1


Display window2



2015-11-30 阿宅 OpenCV, 影像分割 HoughLines, HoughLinesP

自豪的採用 WordPress