阿洲的程式教學
關於Qt、OpenCV、影像處理演算法

找邊緣(Sobel、Scharr)

Sobel是一種獲得影像一階梯度的手法，常見應用於邊緣檢測，有分成水平和垂直方向的模板，就像以下的Gx和Gy模板，Gx用來檢測垂直邊緣，Gy用來檢查水平邊緣，通常會分別對影像進行水平和垂直模板的運算，得到像素的梯度，梯度是一個有距離和方向的二維向量，距離表示變化的幅度，方向表示強度變化最大的方向。

$$G_x = \begin{bmatrix} -1 & 0 & +1 \\ -2 & 0 & +2 \\ -1 & 0 & +1 \end{bmatrix} \qquad G_y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ +1 & +2 & +1 \end{bmatrix}$$

在一般的數學計算上，通常使用歐拉距離(也稱為L2距離)，計算方式為平方的開根號。

$$G = \sqrt{G_x^2 + G_y^2}$$

在影像處理上，由於上式包括平方和根號，計算上較為費時，所以通常採用絕對值之和(L1距離)，OpenCV的Sobel()函式，也是採用絕對值之和。

$$G = |G_x| + |G_y|$$

OpenCV Sobel

void Sobel(InputArray src, OutputArray dst, int ddepth, int dx, int dy, int ksize=3, double scale=1, double delta=0, int borderType=BORDER_DEFAULT)

- src：輸入圖。
- dst：輸出圖，和輸入圖有相同的尺寸和通道數。
- ddepth：輸出圖的深度，假設輸入圖為CV_8U, 支援CV_8U、CV_16S、CV_32F、CV_64F，假設輸入圖為 CV_16U, 支援CV_16U、CV_32F、CV_64F。

- dx：x方向的微分階數。
- dy：y方向的微分階數。
- ksize：核心，必須為1、3、5或7。
- scale：縮放值。
- delta：偏移量。

進行Sobel運算時，要是輸出圖和輸入圖深度相同，很有可能會發生saturate，以8位元強度0到255的影像來說，Sobel運算結果可能大於255或小於0，進而得到不合理的結果，所以假使輸入圖的深度為CV_8U，通常輸出圖深度使用CV_16S。

OpenCV 轉換位元

計算輸入圖各像素，並將結果轉成8位元圖

void convertScaleAbs(InputArray src, OutputArray dst, double alpha=1, double beta=0)

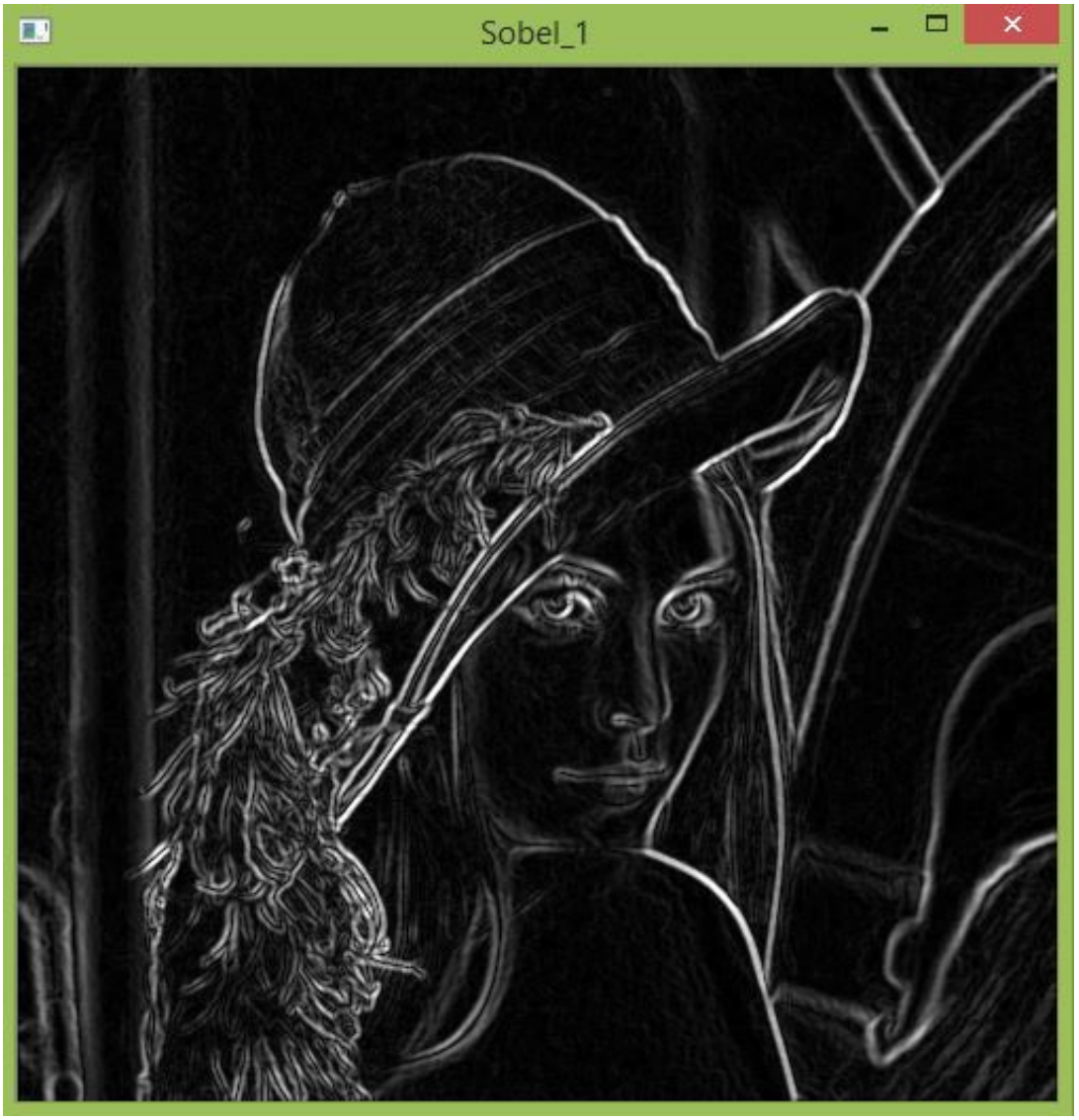
- src：輸入圖。
- dst：輸出圖。
- alpha：選擇性的乘法因子。
- beta：選擇性的加法因子。
- 此函式主要進行3步驟；1.計算 2.取絕對值 3.轉成無正負號8位元圖

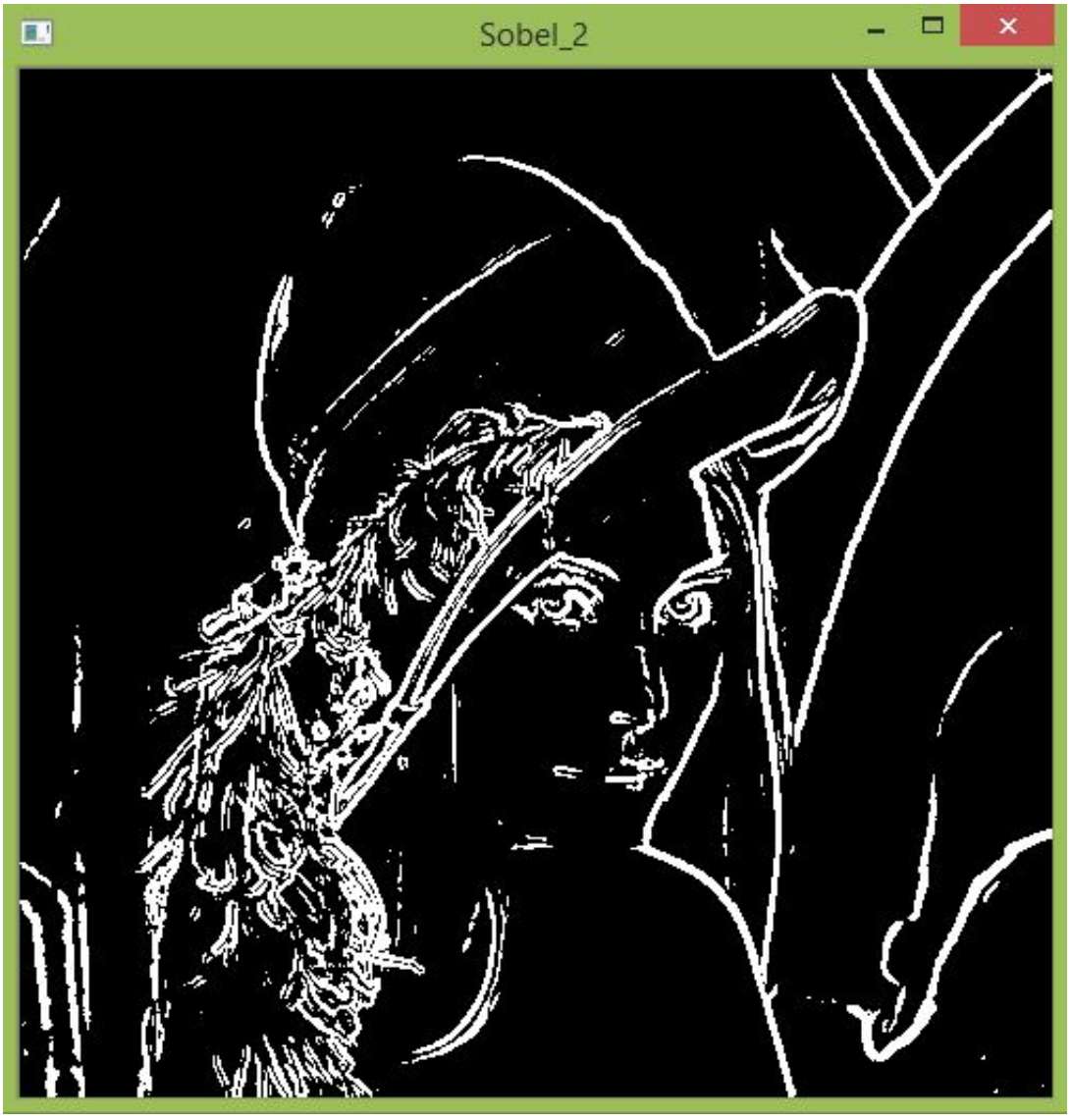
$$\texttt{dst}(I) = \texttt{saturate\_cast<uchar>}(|\texttt{src}(I) * \texttt{alpha} + \texttt{beta}|)$$

以下程式碼示範Sobel()的使用：

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

int main(){
    Mat src = imread("lena.jpg", CV_LOAD_IMAGE_GRAYSCALE);
    GaussianBlur(src, src, Size(3,3), 0, 0);

    Mat grad_x, grad_y;
    Mat abs_grad_x, abs_grad_y;
    Sobel(src, grad_x, CV_16S, 1, 0, 3, 1, 0, BORDER_DEFAULT);
    convertScaleAbs(grad_x, abs_grad_x);  //轉成CV_8U
    Sobel(src, grad_y, CV_16S, 0, 1, 3, 1, 0, BORDER_DEFAULT );
```

```
    convertScaleAbs(grad_y, abs_grad_y);

    Mat dst1, dst2;
    addWeighted( abs_grad_x, 0.5, abs_grad_y, 0.5, 0, dst1);
    threshold(dst1, dst2, 80, 255, THRESH_BINARY|THRESH_OTSU);
    imshow("origin", src);
    imshow("Sobel_1", dst1);
    imshow("Sobel_2", dst2);
    waitKey(0);

    return 0;
}
```


Sobel_1



當我們使用3×3的Sobel核心大小時，可以替換成Scharr運算，Scharr運算通常梯度方向較精確，兩者濾波係數不同，以下分別為Scharr算子的水平和垂直方向模板，Scharr模板只有3×3大小。

$$G_x = \begin{bmatrix} -3 & 0 & +3 \\ -10 & 0 & +10 \\ -3 & 0 & +3 \end{bmatrix} \quad G_y = \begin{bmatrix} -3 & -10 & -3 \\ 0 & 0 & 0 \\ +3 & +10 & +3 \end{bmatrix}$$

OpenCV Scharr

void Scharr(InputArray src, OutputArray dst, int ddepth, int dx, int dy, double scale=1, double delta=0, int borderType=BORDER_DEFAULT)

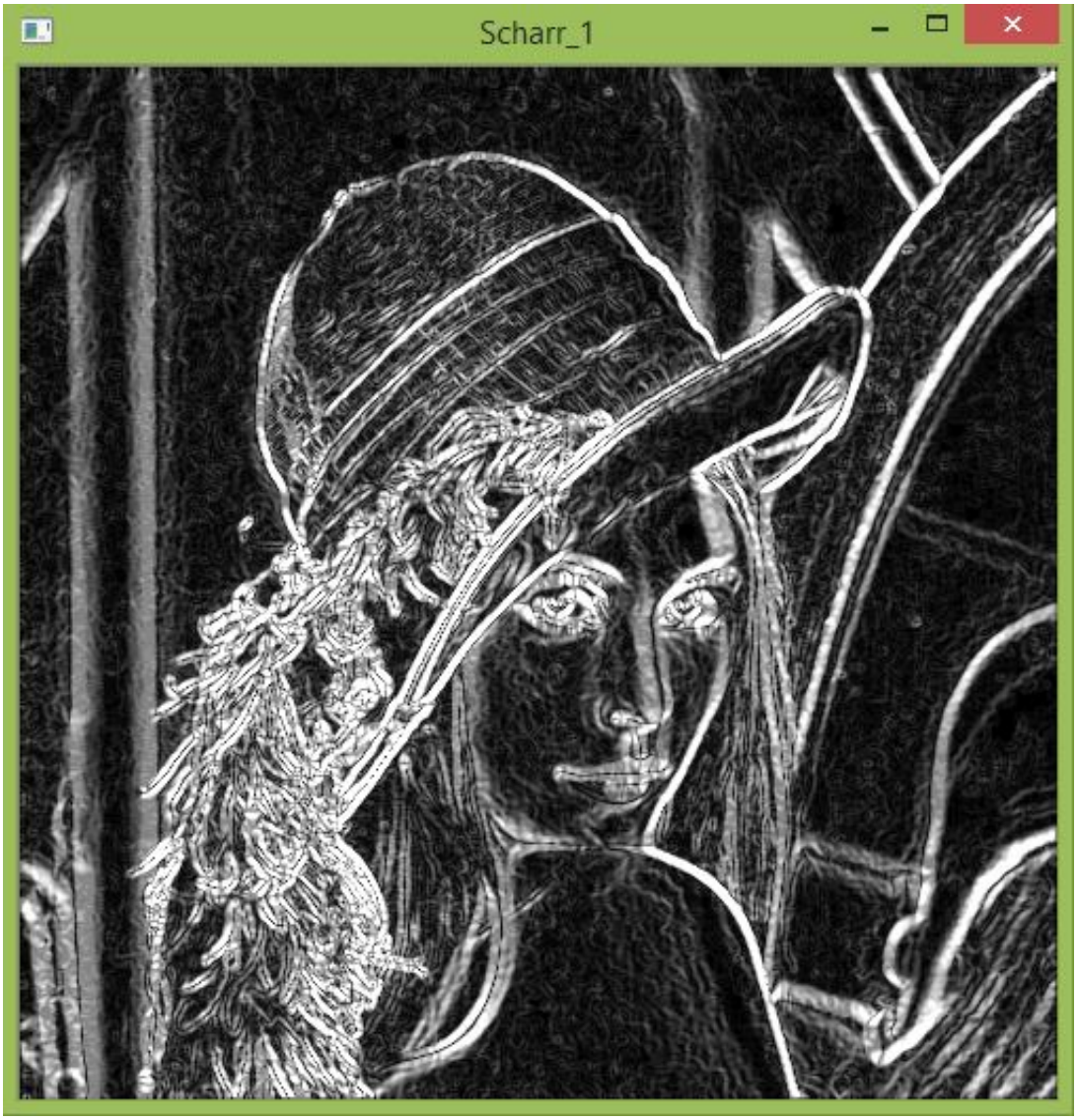
或是Sobel(src, dst, ddepth, dx, dy, CV_SCHARR)，兩者效果相同。

- src：輸入圖。
- dst：輸出圖，和輸入圖有相同的尺寸和通道數。
- ddepth：輸出圖的深度，使用方式和Sobel相同。
- dx：x方向的微分階數。
- dy：y方向的微分階數。
- scale：縮放值
- delta：偏移量。

以下程式碼示範Scharr()的使用：

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

int main(){
    Mat src = imread("lena.jpg", CV_LOAD_IMAGE_GRAYSCALE);
    GaussianBlur(src, src, Size(3,3), 0, 0);

    Mat grad_x, grad_y;
    Mat abs_grad_x, abs_grad_y;
    Scharr(src, grad_x, CV_16S, 1, 0, 1, 0, BORDER_DEFAULT);
    convertScaleAbs(grad_x, abs_grad_x);  //轉成CV_8U
    Scharr(src, grad_y, CV_16S, 0, 1, 1, 0, BORDER_DEFAULT);
    convertScaleAbs(grad_y, abs_grad_y);

    Mat dst1, dst2;
    addWeighted( abs_grad_x, 0.5, abs_grad_y, 0.5, 0, dst1);
    threshold(dst1, dst2, 80, 255, THRESH_BINARY|THRESH_OTSU);
    imshow("origin", src);
    imshow("Sobel_1", dst1);
    imshow("Sobel_2", dst2);
    waitKey(0);

    return 0;
}
```


origin


Scharr_1

回到首頁

回到OpenCV教學

參考資料：

OpenCV 教程

2015-11-30 阿宅 OpenCV, 邊緣 Scharr, Sobel, 邊緣

自豪的採用 WordPress