Instructions for Image Edge Detection

**Zian Gu**

**04/20/2020**

In project 1, I use the OpenCV to detect the edge of image with C++. The procedures and instructions are as follows:

1. Use imread() function to import an image and convert it to a matrix. The image is named as “image”. The second argument “0”, means converting the image to gray scale, in order to simplify the procedure of detection. I use the OpenCV function because the image needs to be decoded, or it cannot be used directly. Realizing the decode function may be difficult for me at this stage. The function can decode the image and converted it to a matrix, then it can do matrix operations, such as convolution operation.
2. Resize the size of the image to make sure the image can be displayed properly. I saved the resized image as “imageS”.
3. Use GaussianBlur() or medianBlur() function to reduce the noise of the image. Image after burred is saved as “imageS\_Blur”. Through my experiment, the kernel is larger, the noise on final image is less. However, it can’t be too large, or the edge will be blurry, and we might loss some tiny edges. I use a 5x5 kernel here.
4. Define the kernel of Sobel operation: ,, or the kernel of Scharr operation: ,. Notice that I use “Mat Gx = (Mat\_<int>(3, 3)-1,0,1,-2,0,2,-1,0,1); to define one of the kernels. The type of the matrix must be Mat\_<int>, then it’s matching to the type of image.

Gx and Gy represents to horizontal and vertical direction.

1. Use the filter2D() function to realize the convolution operation. The image which are convolved with Gx gets the horizontal edge, while Gy gets the vertical edge. The two images are saved as “imageX” and “imageY”. Notice that the argument “CV\_64F” means expanding the depth of the image. Thus, the image can be stored with more bytes, then, if the values after the convolution operation is negative, it can be hold, or it will be converted to 0, and we will loss the details.
2. Use the convertScaleAbs() function to convert the negative numbers in the “imageX” and “imageY” to positive. If not, the display of negative numbers will still be processed as 0, then we cannot get the full edge. The converted images are saved as “imageXAbs” and “imageYAbs”.
3. Use the addWeight() function to merge “imageXAbs” and “imageYAbs”. Then we can get the complete edge of the image.

**Figures:**

****

**Figure1:original image[3000x4000]**

**Figure2:gray scale image**

**(resized down to[480x640])**

****

**Figure3:edge of image[480x640]**

**Processed by a Sobel kernel and Gaussian blur with 5x5 kernel**

**Figure4:edge of image[480x640]**

**Processed by a Sobel kernel and Median blur with 5x5 kernel**

****

**Figure6:edge of image[480x640]**

**Processed by Scharr kernel and Median blur with 5x5 kernel**

**Figure5:edge of image[480x640]**

**Processed by Scharr kernel and Gaussian blur with 5x5 kernel**

****

**Figure8:edge of image[480x640]**

**Processed by Canny operator(use OpenCV directly)**

**Figure7:edge of image[480x640]**

**Processed by Scharr kernel and Gaussian blur with 11x11 kernel**

**Summary:**

Comparing the figures above, we can find that convolving with Scharr kernel can get the edge with thicker lines, thus we can get more lines of edge and more details. However, it may contain many noises. So, I use the Gaussian Blur to reduce the noise of the original image. But if the Gaussian kernel is too large, the noises are reduced perfectly while the lines of edge will be very blurry. Based on the method of Canny operator, we should use non-maximum suppression and double-threshold to exclude the interference that cannot be eliminated by blur. I haven’t learnt the non-maximum suppression and double-threshold according to the learning materials that I am learning. So, if I learn them in the future, I can realize the image edge detection more perfectly.

**Source code:**

/\*

edge detecion.cpp

This demo is used to detect the edge of image.

The algorithm is based on Sobel operator.

I just used the package of Opencv to decode the image(read the image and convert it to matrix) and do some morphological processing.The principle of Sobel is coded by myself.

by Zian Gu

04/17/2020

\*/

#include <opencv2/opencv.hpp>

#include<iostream>

using namespace cv;

using namespace std;

void imgShow(const String& imgName, InputArray img);

int main(int argc, char\*\* argv) {

Mat image = imread("C:\\Users\\gu573\\Documents\\GitHub\\Computer Vision\\img\\Jani2.jpg", 0);//读取图像，注意\\表示\的转义

if (image.empty())

{

printf("Could not load image...\n");

return -1;

}

Mat imageS;//缩小后的图像

Mat imageConverted;

Mat imageS\_Blur;//通过中值滤波后的图像

Mat imageX, imageY;//x,y方向卷积所得的图像

Mat imageXAbs, imageYAbs;//x,y方向卷积所得的图像并取绝对值

Mat imageSobel;//最终通过Sobel算子检测的边缘图像

//Mat Gx = (Mat\_<int>(3, 3) <<// Scharr算子横（x）向处理的卷积核

// -3, 0, 3,

// -10, 0, 10,

// -3, 0, 3);

//Mat Gy = (Mat\_<int>(3, 3) <<// Scharr算子纵（y）向处理的卷积核

// -3, -10, -3,

// 0, 0, 0,

// 3, 10, 3);

Mat Gx = (Mat\_<int>(3, 3) <<// Sobel算子横（x）向处理的卷积核

-1, 0, 1,

-2, 0, 2,

-1, 0, 1);

Mat Gy = (Mat\_<int>(3, 3) <<// Sobel算子纵（y）向处理的卷积核

-1, -2, -1,

0, 0, 0,

1, 2, 1);

//cout << Gx << endl

// << Gy << endl;

resize(image, imageS, imageS.size(), 0.16, 0.16, 1);//改变图像大小，以完整显示

//medianBlur(imageS, imageS\_Blur, 5);//对图像进行中值滤波，否则噪声太大效果不好(也可以用高斯滤波等等尝试一下哪个效果更好，在这里暂时使用中值滤波)。经实验卷积核越大，效果越好，取值为正奇数。

GaussianBlur(imageS,imageS\_Blur,Size(5,5),0);//对图像进行高斯滤波。原理同上。

filter2D(imageS\_Blur,imageX,CV\_64F,Gx);//x方向卷积

convertScaleAbs(imageX, imageXAbs);//需要取绝对值，否则卷积所得负值在显示的时候默认为0，将损失边界

filter2D(imageS\_Blur, imageY, CV\_64F, Gy);//y方向卷积

convertScaleAbs(imageY, imageYAbs);//同理

addWeighted(imageXAbs, 0.5, imageYAbs, 0.5, 0, imageSobel);//按权重将两个方向卷积所得图像融合

//cout << imageS.size();

imgShow("Jani", imageSobel);//输出最终结果

return 0;

}

//显示图像

void imgShow(const String& imgName, InputArray img)

{

imshow(imgName, img);

waitKey(0);

destroyAllWindows();

}