openCV 编程题

Collected and sorted out by Linst <root@sitao.org>

34. 请使用 OpenCV 编写一个简单的程序，用于从当前目录读入并显示一幅彩色图像（例如当前目录中的 lena.jpg）。

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

int main()
{
    Mat im = imread("lena.jpg", 1);
    if (im.empty()) return -1; // 载入失败
    //namedWindow("lena"); // 创建窗口
    imshow("lena", im); // 显示图像

    waitKey(); // 等待按键

}
```

35. 请使用 OpenCV 编写一个简单的程序，用于从当前目录读入并显示一幅灰度图像（例如当前目录中的 lena.jpg）。

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

int main()
{
    Mat im = imread("lena.jpg", 0);
    if (im.empty()) return -1; // 载入失败
   // namedWindow("lena"); // 创建窗口
    imshow("lena", im); // 显示图像

    waitKey(); // 等待按键

}
```

36. 请使用 OpenCV 编写一个简单的程序，该程序首先读入一幅彩色图像（例如当前目录中的 lena.jpg），然后将这幅彩色图像的 3 个通道分离出来，得到 3 幅灰度图像，最后显示这 3 幅灰度图像。

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

```
int main()
{
    Mat im = imread("lena.jpg", 1);
    if (im.empty()) return -1; // 载入失败
    vector<Mat> mv;
    split(im, mv);

    imshow("蓝色图像", mv[0]);
    imshow("绿色图像", mv[1]);
    imshow("红色图像", mv[2]);
    waitKey();
}
```

37. 请使用 OpenCV 编写一个简单的程序，该程序从 1 幅彩色图像（使用当前目录中的 lena.jpg）中分离出蓝色通道，得到 1 幅灰度图像。要求显示源图像和结果图像。

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


int main()
{
    Mat X = imread("lena.jpg");
    if(X.empty()) return -1;
    imshow("源", X);

    Mat B;
    extractChannel(X, B, 0); //提取蓝色通道
    imshow("结果", B);

    waitKey();
}
```

38. 请使用 OpenCV 编写一个简单的程序，该程序首先从一幅大小至少是 300*300 的真彩色图像（使用当前目录中的 lena.jpg）中选取一个矩形子集，并用蓝色填充该矩形子集，然后显示图像。其中矩形子集的起始位置为 (64, 96)，大小为 (96, 48)。

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

int main()
{
    Mat src = imread("lena.jpg", 1);
    if (src.empty()) return -1; // 载入失败
```

```
    // 显示原始图像
    imshow("src", src);

    // 定义矩形区域：起始位置 (64, 96)，大小 (96, 48)
    Rect rect(64, 96, 96, 48);

    // 获取图像中对应矩形区域的子集
    Mat roi = src(rect);

    // 用蓝色填充矩形子集（BGR顺序中蓝色是 (255, 0, 0)）
    roi = Scalar(255, 0, 0);

    // 显示修改后的图像（显示整个图像，包括被修改的矩形区域）
    imshow("dst", src);

    waitKey();
}
```

39. 使用OpenCV装入一幅大小至少为300*300的真彩色图像，并显示该图像（使用当前目录中的lena.jpg）。然后在源图像中指定一个矩形区域（左上顶点和宽高值分别为(64, 128)和(128, 64)的矩形），并在结果图像窗口中显示源图像中被选取的部分。

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

int main()
{
    Mat src = imread("lena.jpg", 1);
    if (src.empty()) return -1; // 载入失败
    imshow("src", src);
    Rect rect(64, 128, 128, 64);
    Mat dst = src(rect);
    imshow("dst", dst);
    waitKey();
}
```

40. 使用OpenCV编写一个程序，该程序将一幅灰度图像（使用当前目录中的lena.jpg）的灰度值线性地变换到范围[0, 255]。要求分别显示源图像和结果图像。

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

int main()
{
```

```
    Mat X = imread("lena.jpg", 0);
    if (X.empty()) return -1; // 载入失败

    Mat Y;
    normalize(X, Y, 255, 0, NORM_MINMAX);

    imshow("源图像", X);
    imshow("结果图像", Y);
    waitKey();
}
```

41. 随机生成一幅浮点数灰度图像（大小和亮度都是随机的，大小值位于区间[128, 639]），然后将该图像变换成亮度是0～1的浮点数图像，最后变换成字节图像并显示该图像。

```
#include <opencv2/opencv.hpp>
#include <iostream>

using namespace cv;

int main() {
    // 创建RNG对象
    RNG rng;

    // 随机生成图像大小 [128, 639]
    int rows = 128 + rng.uniform(0, 512); // 512 = 639 - 128 + 1
    int cols = 128 + rng.uniform(0, 512);

    // 随机生成浮点数灰度图像（亮度随机范围）
    Mat randomImage(rows, cols, CV_32FC1);
    rng.fill(randomImage, RNG::UNIFORM, 0.0, 255.0); // 填充随机浮点数灰度值

    // 将图像亮度变换到 [0, 1]
    normalize(randomImage, randomImage, 0, 1, NORM_MINMAX);

    // 将浮点数图像变换为字节图像（范围 [0, 255]）
    Mat byteImage;
    randomImage.convertTo(byteImage, CV_8UC1, 255);

    // 显示图像
    imshow("Random Float Image", randomImage); // 显示浮点数图像
    imshow("Byte Image", byteImage);           // 显示字节图像

    waitKey(0); // 等待按键

}
```

42. 首先使用OpenCV装入一幅灰度图像（例如当前目录中的lena.jpg），然后使用函数min()过滤掉源图像中亮度大于指定值（例如128）的像素，并显示源图像和结果图像以便对比。

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

int main(){
	Mat src = imread("lena.jpg", 0);
	if (src.empty()) return -1; // 载入失败
	Mat dst;
	min(src, 128, dst);
	imshow("源", src);
	imshow("结果", dst);

	waitKey(0);
}
```

46. 使用OpenCV编写一个演示傅立叶变换和逆变换的程序。该程序首先装入一幅灰度图像并显示该图像（例如当前目录中的lena.jpg），然后对该图像进行傅立叶正变换，对得到的结果进行傅立叶逆变换，显示得到的结果以便与原图像进行比对。

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

int main()
{
   Mat X = imread("lena.jpg", 0);
   if (X.empty()) return -1;

   Mat Y;
   X.convertTo(Y, CV_32F, (double)1 / 255);
   dft(Y, Y);
   idft(Y, Y, DFT_SCALE);
   imshow("src", X);
   imshow("dst", Y);

   waitKey();
}
```

49. 使用OpenCV编写一个程序，该程序对一幅灰度图像（使用当前目录中的lena-n.jpg）进行一次简单模糊，要求分别显示源图像和结果图像。其中内核大小为3×3。

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

```
int main()
{
    Mat X = imread("lena-n.jpg", 0);
    if (X.empty()) return -1;
    imshow("源图像", X);

    Mat Y;
    blur(X, Y, {3, 3});
    imshow("简单模糊", Y);
    waitKey();

}
```

50. 使用 OpenCV 编写一个程序，该程序对一幅彩色图像（例如当前目录中的 lena-n.jpg）进行一次中值模糊，要求分别显示源图像和模糊化以后的图像。其中内核大小为 5×5。

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

int main()
{
    Mat X = imread("lena-n.jpg", 1);
    if (X.empty()) return -1;
    imshow("源图像", X);

    Mat Y;
    medianBlur(X, Y, 5);
    imshow("中值模糊", Y);
    waitKey();

}
```

51. 使用 OpenCV 编写一个程序，该程序对一幅灰度图像（使用当前目录中的 lena-n.jpg）进行一次高斯模糊，要求分别显示源图像和结果图像。其中内核大小为 3×3，标准差由 OpenCV 自动计算。

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

int main()
{
    Mat X = imread("lena-n.jpg", 0);
    if (X.empty()) return -1;
```

```
    imshow("源图像", X);

    Mat Y;
    GaussianBlur(X, Y, {3, 3},0,0);
    imshow("高斯模糊", Y);
    waitKey();

}
```

52. 使用 OpenCV 编写一个程序，该程序对一幅灰度图像（例如当前目录中的 lena.jpg）进行 Sobel 锐化，要求显示锐化以后的图像。其中内核大小为 3×3，*x* 和 *y* 方向均使用 1 阶差分。

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

int main()
{
Mat X = imread("lena.jpg", 0);
if (X.empty()) return -1;
imshow("src", X);

Sobel(X, X, -1, 1, 1, 3);
imshow("dst", X);
waitKey();
}
```

53. 使用 OpenCV 编写一个程序，该程序对一幅灰度图像（使用当前目录中的 lena.jpg）进行 Sobel 锐化，要求显示源图像和结果图像。其中内核大小为 3×3，使用 1 阶 *x* 差分计算模版系数。

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

int main()
{
Mat X = imread("lena.jpg", 0);
if (X.empty()) return -1;
imshow("src", X);
Sobel(X, X, -1, 1, 0, 3);
imshow("dst", X);
waitKey();
}
```

54. 使用 OpenCV 编写一个程序，该程序对一幅灰度图像（例如当前目录中的 lena.jpg）进行 Laplace 锐化，要求显示锐化以后的图像。其中内核大小为 3×3。

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

int main()
{
Mat X = imread("lena.jpg", 0);
if (X.empty()) return -1;
imshow("src", X);
Laplacian(X, X, -1, 3);
imshow("dst", X);
waitKey();
}
```

55. 使用 OpenCV 编写一个程序，该程序使用大小为 3 的正方形模板（锚点位于模板中心）对一幅灰度图像（例如当前目录中的 image-j.bmp）进行腐蚀操作，要求显示源图像和腐蚀以后的图像。

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

int main()
{
	Mat X = imread("image-j.bmp", 0);
	if (X.empty()) return -1;
	imshow("源图像", X);
	Mat K = getStructuringElement(MORPH_RECT, {3,3}); //创建模版
	Mat Y;
	erode(X, Y, K);
	imshow("腐蚀", Y);

	waitKey();
}
```

56. 使用 OpenCV 编写一个程序，该程序使用大小为 3 的正方形模板（锚点位于模板中心）对一幅灰度图像（例如当前目录中的 image-j.bmp）进行开运算操作，要求显示源图像和开运算以后的图像。

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

int main()
{
	Mat X = imread("image-j.bmp", 0);
```

```
    if (X.empty()) return -1;
    imshow("源图像", X);
    Mat K = getStructuringElement(MORPH_RECT, {3,3}); //创建模版
    Mat Y;
    morphologyEx(X, Y, MORPH_OPEN, K);
    imshow("开运算", Y);

    waitKey();
}
```

58. 使用 OpenCV 编写一个程序，该程序对一幅灰度图像（例如当前目录中的 lena.jpg）进行二值化变换，要求分别显示源图像和二值化以后的图像。其中二值化阈值为 127，高亮度改为 255。

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

int main()
{
Mat X = imread("lena.jpg", 0);
imshow("Source image", X);
threshold(X, X, 127, 255, THRESH_BINARY);
imshow("Binary Image", X);
waitKey();
}
```

59. 使用 OpenCV 编写一个程序，该程序对一幅灰度图像（例如当前目录中的 lena.jpg）进行 Canny 边缘检测，要求分别显示源图像和检测到的边缘。其中小阈值为 50，大阈值为 150，内核大小为 3。

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

int main()
{
Mat X = imread("lena.jpg", 0);
imshow("Source image", X);
Canny(X, X, 50, 150, 3);
imshow("Canny Image", X);
waitKey();
}
```

60. 使用 OpenCV 编写一个程序，该程序对一幅彩色图像（例如当前目录中的 lena.jpg）使用指定的模板（例如当前目录中的 Template.jpg）进行模板匹配。要求使用差平方匹配算法进行模板匹配，源图像中与模板最匹配的区域用一个蓝色矩形标记。

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

int main ()
{
Mat I=imread ("lena.jpg")；//源图像
Mat T=imread ("Template.jpg")；// 模板图像
imshow ("模板", T)：//显示模板图像
Mat R; // 结果矩阵
matchTemplate (I, T, R, TM_SQDIFF)；//模板匹配 (差平方匹配)
Point minLoc; // 保存最小值位置，即矩形起点
minMaxLoc (R, 0,0，&minLoc)；//最小值位置
Rect rect {minLoc,T.size ()}; // 矩形 (起点，大小)
rectangle (I, rect, {255,0,0})；// 绘制矩形 (蓝色)
imshow ("匹配"，I);//显示匹配结果
waitKey ()；// 等待按键
}
```