C++小項目：OpenCV 車牌定位

新機器視覺 昨天

最近開始接觸C++ 了，就拿一個OpenCV 小項目來練練手。在車牌自動識別系統中，從汽車圖像的獲取到車牌字符處理是一個複雜的過程，本文就以一個簡單的方法來處理車牌定位。

我國的汽車牌照一般由七個字符和一個點組成，車牌字符的高度和寬度是固定的，分別為90mm和45mm，七個字符之間的距離也是固定的12mm，點分割符的直徑是10mm。

使用的圖片是從百度上隨便找的（侵刪），展示一下原圖和灰度圖：

```
#include <iostream>
#include <opencv2/highgui/highgui.hpp>
#include <opencv2/imgproc.hpp>
#include <opencv2/imgproc/types_c.h>

using namespace std;
```

```
using namespace cv;

int main() {
 // 读入原图
 Mat img = imread("license.jpg");
 Mat gray_img;
 // 生成灰度图像
 cvtColor(img, gray_img, CV_BGR2GRAY);
 // 在窗口中显示游戏原画
 imshow("原图", img);
 imshow("灰度图", gray_img);
 waitKey(0);
 return 0;
}
```


WEY
VV6
沪N·A7538
https://blog.csdn.net/weixin_44613063

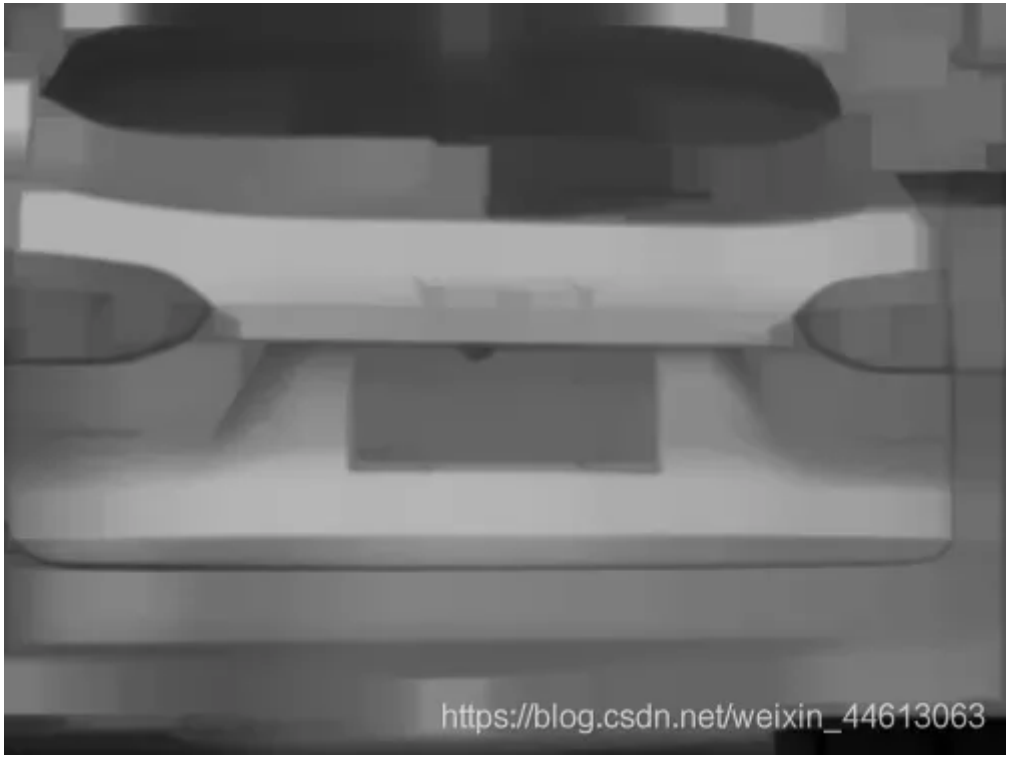
灰度圖像的每一個像素都是由一個數字量化的，而彩色圖像的每一個像素都是由三個數字組成的向量量化的，使用灰度圖像會更方便後續的處理。

圖像降噪

每一副圖像都包含某種程度的噪聲，在大多數情況下，需要平滑技術（也常稱為濾波或者降噪技術）進行抑製或者去除，這些技術包括基於二維離散卷積的高斯平滑、均值平滑、基於統計學方法的中值平滑等。這裡採用基於二維離散卷積的高斯平滑對灰度圖像進行降噪處理，處理後的圖像效果如下：

形態學處理

完成了高斯去噪以後，為了後面更加準確的提取車牌的輪廓，我們需要對圖像進行形態學處理，在這裡，我們對它進行開運算，處理後如下所示：



開運算呢就是先進行erode 再進行dilate 的過程就是開運算，它具有消除亮度較高的細小區域、在纖細點處分離物體，對於較大物體，可以在不明顯改變其面積的情況下平滑其邊界等作用。

erode 操作也就是腐蚀操作，类似于卷积，也是一种邻域运算，但计算的不是加权求和，而是对邻域中的像素点按灰度值进行排序，然后选择该组的最小值作为输出的灰度值。

dilate 操作就是膨胀操作，与腐蚀操作类似，膨胀是取每一个位置邻域内的最大值。既然是取邻域内的最大值，那么显然膨胀后的输出图像的总体亮度的平均值比起原图会有所上升，而图像中较亮物体的尺寸会变大；相反，较暗物体的尺寸会减小，甚至消失。

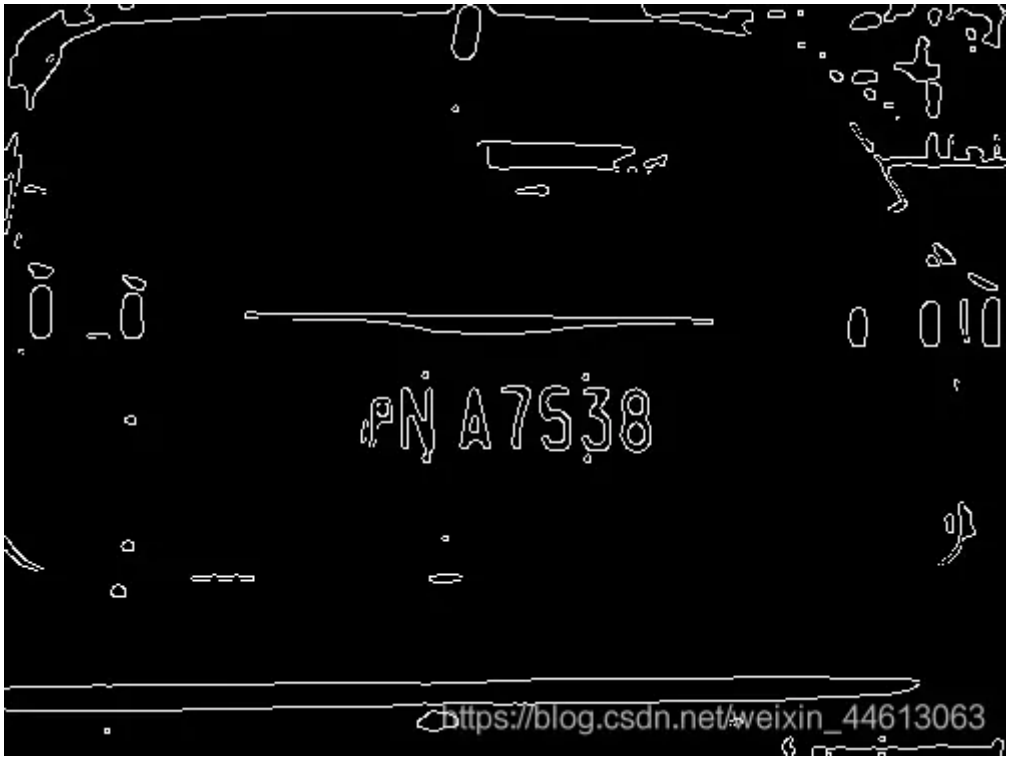
阈值分割

完成初步的形态学处理以后，我们需要对图像进行阈值分割，我们在这里采用了 Otsu 阈值处理，处理后的效果如下所示：

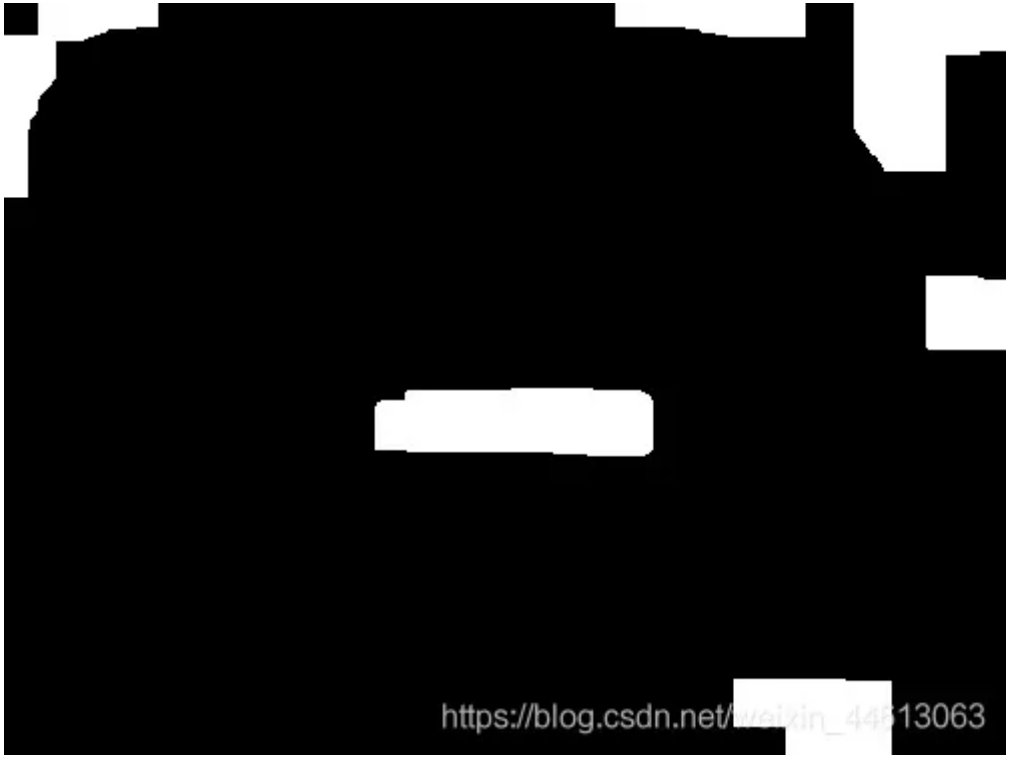
对图像进行数字处理时，我们需要把图像分成若干个特定的、具有独特性质的区域，每一个区域代表一个像素的集合，每一个集合又代表一个物体，而完成该过程的技术通常称为图像分割，它是从图像处理到图像分析的关键步骤。其实这个过程不难理解，就好比我们人类看景物一样，我们所看到的世界是由许许多多的物体组合而成的，就像教室是由人、桌子、书本、黑板等等组成。我们通过阈值处理，就是希望能够从背景中分离出我们的研究对象。

边缘检测

经过Otsu阈值分割以后，我们要对图像进行边缘检测，我们这里采用的是Canny边缘检测，处理后的结果如下：



接下来再进行一次闭运算和开运算，填充白色物体内细小黑色空洞的区域并平滑其边界，处理后的效果如下：



这个时候，车牌的轮廓已经初步被选出来了，只是还有一些白色块在干扰。

上述过程的代码：

```
// 得出轮廓
bool contour(Mat image, vector<vector<Point>> &contours, vector<Vec4i> &hierarchy) {
 Mat img_gau, img_open, img_seg, img_edge;
 // 高斯模糊
 GaussianBlur(image, img_gau, Size(7, 7), 0, 0);
 // 开运算
 Mat element = getStructuringElement(MORPH_RECT, Size(23, 23));
```

```
    morphologyEx(img_gau, img_open, MORPH_OPEN, element);
    addWeighted(img_gau, 1, img_open, -1, 0, img_open);
    // 阈值分割
    threshold(img_open, img_seg, 0, 255, THRESH_BINARY + THRESH_OTSU);
    // 边缘检测
    Canny(img_seg, img_edge, 200, 100);
    element = getStructuringElement(MORPH_RECT, Size(22, 22));
    morphologyEx(img_edge, img_edge, MORPH_CLOSE, element);
    morphologyEx(img_edge, img_edge, MORPH_OPEN, element);
    findContours(img_edge, contours, hierarchy, RETR_TREE, CHAIN_APPROX_SIMPLE, Point());
    return true;
}
```

选取轮廓

现在我们已经有了轮廓，我们需要筛选出车牌所在的那个轮廓，由于车牌宽和高的比例是固定的，依据这个几何特征，我们进行筛选，效果如图：

代码如下：

```
// 车牌轮廓点
Point2f(*choose_contour(vector<vector<Point>> contours))[2] {
 int size = (int)contours.size();
 int i_init = 0;
 Point2f (*contours_result)[2] = new Point2f[size][2];
 for (int i = 0; i < size; i++){
```

```
  // 获取边框数据
  RotatedRect number_rect = minAreaRect(contours[i]);
  Point2f rect_point[4];
  number_rect.points(rect_point);
  float width = rect_point[0].x - rect_point[1].x;
  float height = rect_point[0].y - rect_point[3].y;
  // 用宽高比筛选
  if (width < height) {
   float temp = width;
   width= height;
   height = temp;
  }
  float ratio = width / height;
  if (2.5 < ratio && ratio < 5.5) {
   contours_result[i_init][0] = rect_point[0];
   contours_result[i_init][1] = rect_point[2];
   i_init++;
  }

 }
 return contours_result;
}

// 截取车牌区域
int license_gain(Point2f (*choose_license)[2], Mat img) {
 int size = (int)(_msize(choose_license) / sizeof(choose_license[0]));
 // 绘制方框
 for (int i = 0; i < size; i++) {
  if ((int)choose_license[i][0].x > 1 && (int)choose_license[i][0].y > 1) {
   int x = (int)choose_license[i][1].x;
```

```
    int y = (int)choose_license[i][1].y;
    int width = (int)(choose_license[i][0].x) - (int)(choose_license[i][1].x);
    int height = (int)(choose_license[i][0].y) - (int)(choose_license[i][1].y);
    Rect choose_rect(x, y, width, height);
    Mat number_img = img(choose_rect);
    rectangle(img, choose_license[i][0], choose_license[i][1], Scalar(0, 0, 255), 2, 1, 0);
    imshow("车牌单独显示" + to_string(i), number_img);
   }
  }
  imshow("绘制方框", img);
  return 0;
}
```

最后的 main 函数：

```
int main() {
 // 读入原图
 Mat img = imread("license.jpg");
 Mat gray_img;
 // 生成灰度图像
 cvtColor(img, gray_img, CV_BGR2GRAY);
 // 得出轮廓
 vector<vector<Point>> contours;
 vector<Vec4i> hierarchy;
 contour(gray_img, contours, hierarchy);
 // 截取车牌
 Point2f (*choose_license)[2] = choose_contour(contours);
 license_gain(choose_license, img);
```

```
    delete [] choose_license;
    waitKey(0);
    return 0;
}
```

转自：Steven·简谈
https://blog.csdn.net/weixin_44613063/article/details/109409564

End

喜歡此內容的人還喜歡

關於圖像語義分割的總結和感悟

新機器視覺

深度學習圖像分類任務中那些不得不看的技巧總結

新機器視覺

【OpenCV入門】手把手教你圖片預處理

新機器視覺