数字媒体技术基础作业 1

15331416 赵寒旭

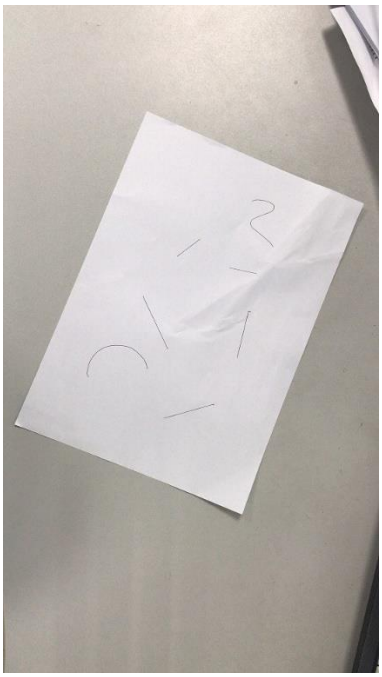
1. 实验环境

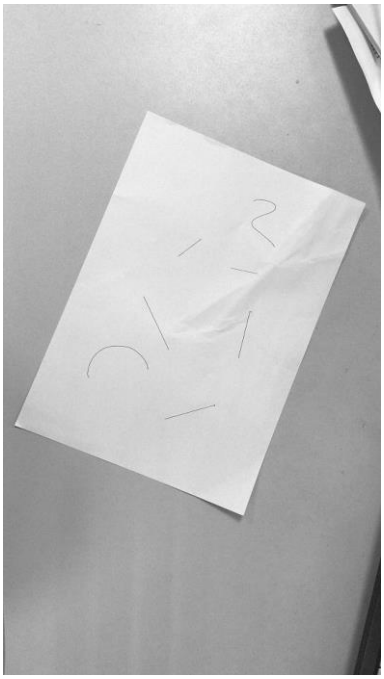
软件：Matlab R2015b

运行系统：windows10

2. 实验步骤（结合代码说明）

1) 将图像转成灰度图



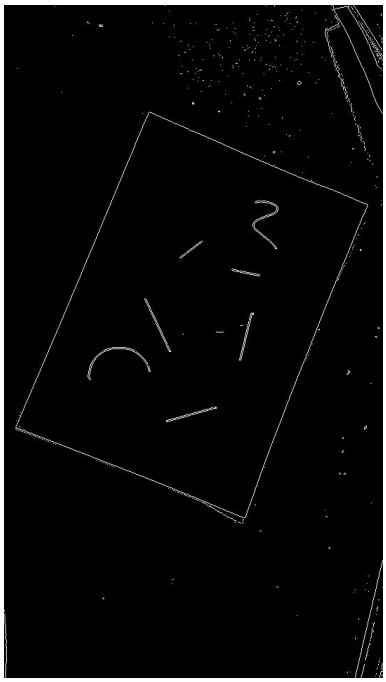


左图为原图，右图为转换后灰度图像。

方法：调用 matlab 中的 rgb2gray 函数，把输入的彩色图像从 rgb 模型转到灰度图。

算法：`img_gray = rgb2gray(img);`

2) 边缘检测（检测出图像的边缘信息）

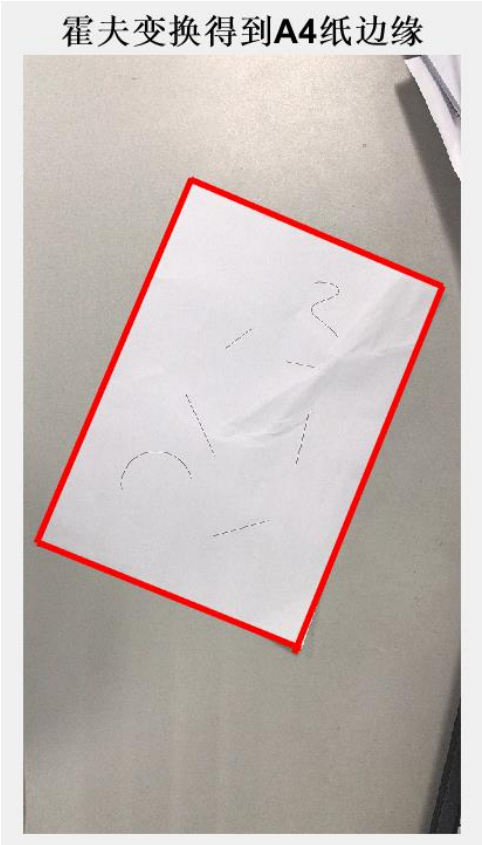


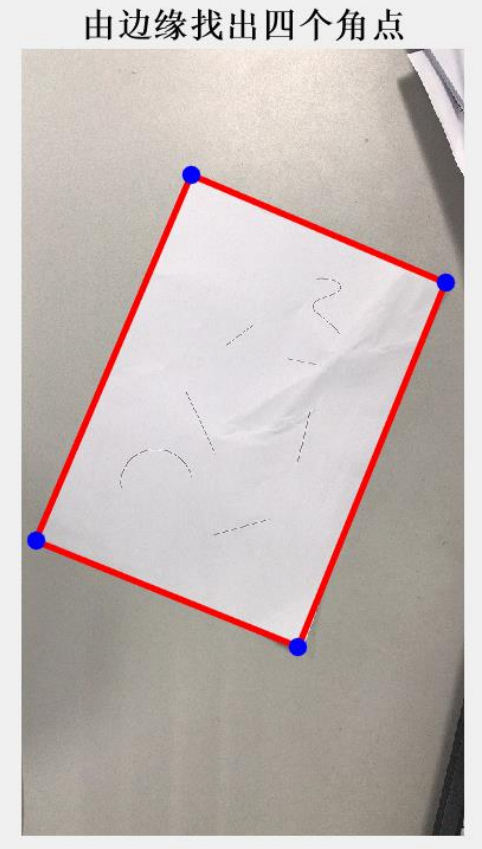
左图为 sobel 算子下的边缘提取结果。

方法：matlab 下 edge 根据选择的算子（sobel）对将图像转为二值图并提取边缘。

算法：`edged_result = edge(img_gray,'sobel');`

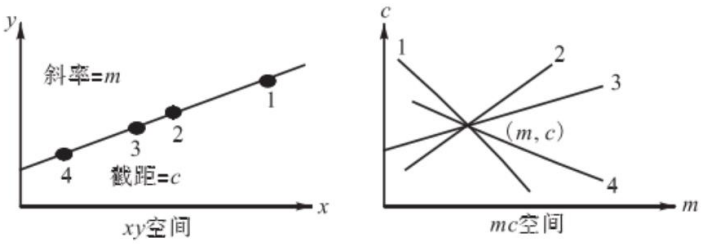
edge函数功能是采用img_gray作为它的输入，并返回一个与img_gray相同大小的二值化图像edged_result，在函数检测到边缘的地方为 1，其他地方为 0。此处未设置相关参数值，自动选择阈值用sobel算子进行边缘检测,由图可见，提取了主要边缘，效果较好，因此并未修改参数。





3) 通过霍夫变换得到 A4 纸边缘（可以看到 A4 纸外有一些线，可以通过霍夫变换去除）

方法：霍夫变换本质上是坐标变换，对图片中的直线来说，将直线中的每个点都变换到变换后空间，找出多条直线相交的点就可以找出变换前空间的直线。



左半部分表示直线的xy空间，直线方程为y = mx + c，其中斜率为m，截距为c。右半部分表示将直线从xy空间变换到mc空间，取直线在xy空间上的四点 1,2,3,4，在mc空间就是不同斜率和截距的四条直线。在mc空间中四条直线的交点处的m值和c值就对应xy空间中直线的m和c。

实际应用中是将xy空间变换到极坐标系，方程为ρ = xcos(θ) + ysin(θ)。

算法：

```
% 3. 霍夫变换得到A4纸边缘
[H,Theta,Rho] = hough(edged_result);
P  = houghpeaks(H,7,'threshold',0.4*max(H(:)));
x = Theta(P(:,2));
y = Rho(P(:,1));
lines = houghlines(edged_result,Theta,Rho,P,'FillGap',200,'MinLength',180);
```

函数中选择的参数是根据图像变换后的结果反复实验调整得到的，可能并不精确，但较好地

实现了提取边缘的要求。

hough：实现霍夫变换，得到霍夫变换矩阵。

$[H, theta, rho] = hough(BW)$

houghpeaks：在霍夫变换矩阵里找极值点

peaks = houghpeaks(H, numpeaks)

peaks = houghpeaks(..., param1, val1,param2, val2)

houghlines：从霍夫变换矩阵中提取线段

lines = houghlines(BW, theta, rho,peaks)

lines = houghlines(..., param1, val1,param2, val2)

同一条线段会由于某些原因（比如光照、噪音等）变成了不连续的两条较短的线段，所以要进项合并，Fillgap 参数决定将哪些长度的线段合并成同一条直线，Minlength 参数决定选择线段的最小值，小于此值的就舍去，滤去了噪声。

在图中描出边缘线段（用红色突出显示）：

```
for k = 1:length(lines)
 xy = [lines(k).point1; lines(k).point2];
 plot(xy(:,1),xy(:,2),'LineWidth',2,'Color','r');
 hold on
end
```

4) 通过 A4 纸的边计算 A4 纸的 4 个角点

方法：通过每条线段已知两端点，求出四条线段所在直线对应的全部四个交点。

我们考虑二维空间中两条直线L_1和L_2的交点。

L_1由相异的两点(x_1, y_1)和(x_2, y_2)确定，L_2由相异的两点(x_3, y_3)和(x_4, y_4)确定。

两直线的交点 P 可以由行列式确定：

$$P_x = \frac{\begin{vmatrix} \begin{vmatrix} x_1 & y_1 \\ x_2 & y_2 \end{vmatrix} & \begin{vmatrix} x_1 & 1 \\ x_2 & 1 \end{vmatrix} \\ \begin{vmatrix} x_3 & y_3 \\ x_4 & y_4 \end{vmatrix} & \begin{vmatrix} x_3 & 1 \\ x_4 & 1 \end{vmatrix} \end{vmatrix}}{\begin{vmatrix} \begin{vmatrix} x_1 & 1 \\ x_2 & 1 \end{vmatrix} & \begin{vmatrix} y_1 & 1 \\ y_2 & 1 \end{vmatrix} \\ \begin{vmatrix} x_3 & 1 \\ x_4 & 1 \end{vmatrix} & \begin{vmatrix} y_3 & 1 \\ y_4 & 1 \end{vmatrix} \end{vmatrix}} \qquad P_y = \frac{\begin{vmatrix} \begin{vmatrix} x_1 & y_1 \\ x_2 & y_2 \end{vmatrix} & \begin{vmatrix} y_1 & 1 \\ y_2 & 1 \end{vmatrix} \\ \begin{vmatrix} x_3 & y_3 \\ x_4 & y_4 \end{vmatrix} & \begin{vmatrix} y_3 & 1 \\ y_4 & 1 \end{vmatrix} \end{vmatrix}}{\begin{vmatrix} \begin{vmatrix} x_1 & 1 \\ x_2 & 1 \end{vmatrix} & \begin{vmatrix} y_1 & 1 \\ y_2 & 1 \end{vmatrix} \\ \begin{vmatrix} x_3 & 1 \\ x_4 & 1 \end{vmatrix} & \begin{vmatrix} y_3 & 1 \\ y_4 & 1 \end{vmatrix} \end{vmatrix}}$$

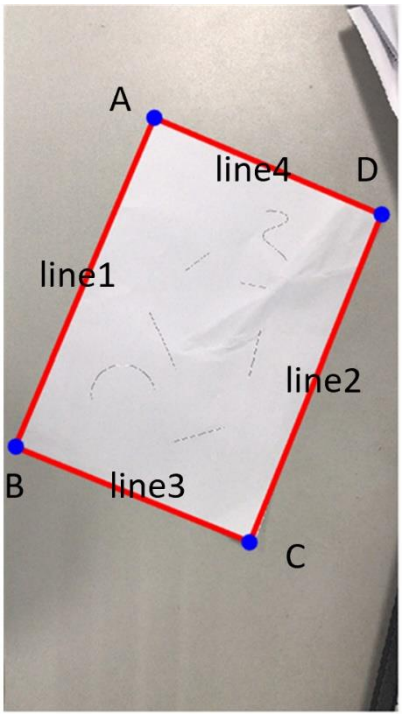
$$(P_x, P_y) = \left(\frac{(x_1y_2 - y_1x_2)(x_3 - x_4) - (x_1 - x_2)(x_3y_4 - y_3x_4)}{(x_1 - x_2)(y_3 - y_4) - (y_1 - y_2)(x_3 - x_4)}, \frac{(x_1y_2 - y_1x_2)(y_3 - y_4) - (y_1 - y_2)(x_3y_4 - y_3x_4)}{(x_1 - x_2)(y_3 - y_4) - (y_1 - y_2)(x_3 - x_4)} \right)$$

特殊地，当两直线平行时，有分母对应值为 0，无交点：

$(x_1 - x_2)(y_3 - y_4) - (y_1 - y_2)(x_3 - x_4) = 0$

算法：

lines 包含四项，每项中含有对应线段的起点和终点坐标。




```
Command Window
>> lines(1)

ans =

    point1: [208 153]
    point2: [17 602]
     theta: 23
       rho: 250
```

可以确定每个索引对应的实际线段

调用自定义 intersection 函数，由两个线段的两端点共四个点的坐标组成的矩阵

$$[x_1, y_1; x_2, y_2; x_3, y_3; x_4, y_4]$$

返回两直线交点。

```
% 4. 通过A4纸的边计算A4纸的四个角点
line_14 = [ lines(1).point1;lines(1).point2;lines(4).point1;lines(4).point2];
A = intersection(line_14);
line_13 = [ lines(1).point1;lines(1).point2;lines(3).point1;lines(3).point2];
B = intersection(line_13);
line_32 = [ lines(3).point1;lines(3).point2;lines(2).point1;lines(2).point2];
C = intersection(line_32);
line_24 = [ lines(2).point1;lines(2).point2;lines(4).point1;lines(4).point2];
D = intersection(line_24);
```

```
A = round(A);
B = round(B);
C = round(C);
D = round(D);
```

用 round 函数将计算得到的小数坐标转成对应像素的实际坐标。

5) 根据四个角点做透视变换

方法：透视变换

算法：

定义原图中找到的四个角点的矩阵和对应的目标位置矩阵。

利用 cp2tform 函数直接以这四对坐标为参考进行透视变换，参数'projective'

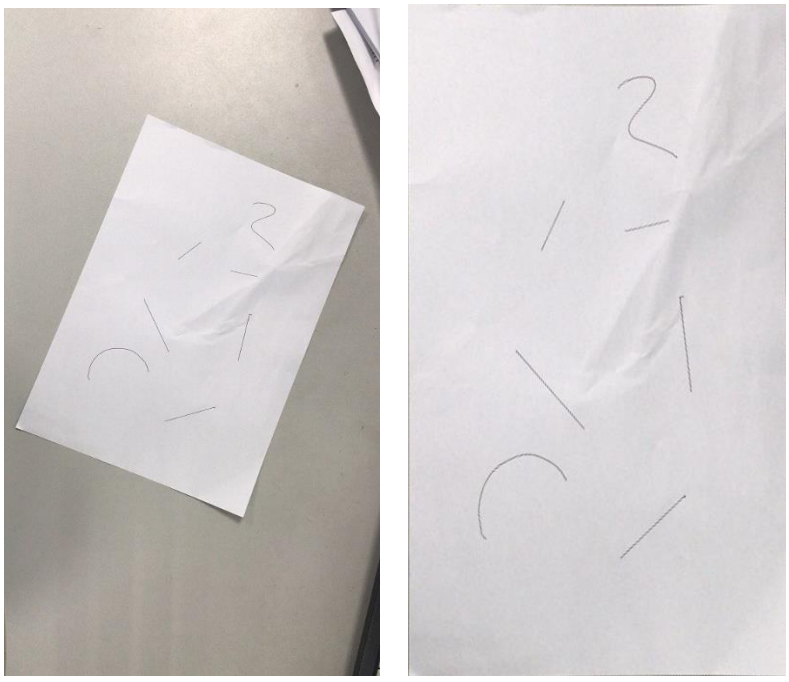
输出图像由 imtransform 对透视变换矩阵操作得到。

注意 imtransform 要截取目标图像大小的部分，否则会使图像显示时扭曲缩小。

('XData',[1 540], 'YData',[1 960])

```
% 透视变换后图像
original = [A;B;C;D];
new = [1 1;1 960;540 960;540 1];
TForm = cp2tform(original,new,'projective');
figure(2);
subplot(1,2,2);
output_img = imtransform(img,TForm,'XData',[1 540], 'YData',[1 960]);
```

3. 实验结果



左图为 input.jpg，右图为输出的校正后图像。
可以看到纸片上内容均均匀投射到输出图像上，校正了纸片的倾斜状况。