

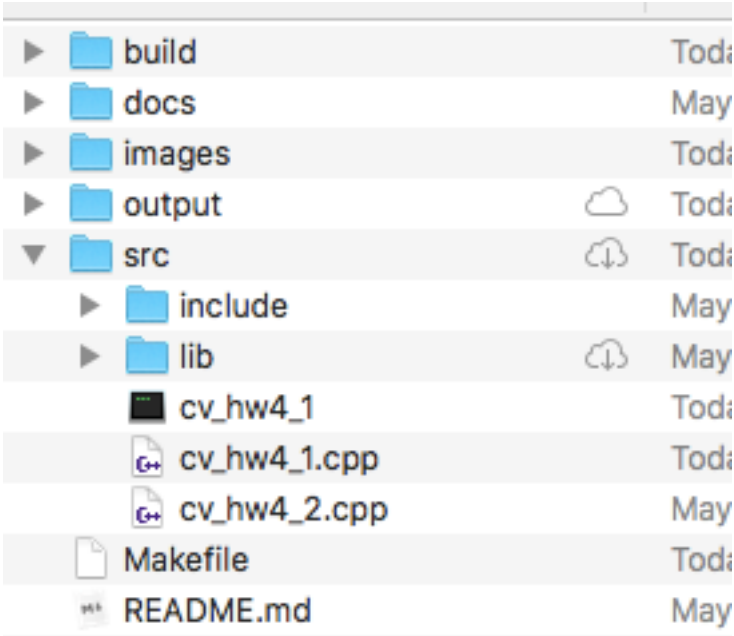
# 测试文档

课程	计算机视觉
姓名	詹宗沅
学号	15331386
时间	2018.05.02

## 一、项目代码说明

### 1.1 项目结构

- 1. build：项目构建产生中间的链接文件（如.o文件）
- 2. docs：项目相关文档（测试文档，实验要求）
- 3. images：项目图片数据集
- 4. output：程序运行结果截图
- 5. src：项目代码
  - 5.1 include：实现项目功能的主要原代码，包括lineDetector、cannyIit等
  - 5.2 lib：项目调用的第三方库：CImg，mac链接库X11（上传的代码中由于过大将其删除）
  - 5.3 cv\_hw4\_1.cpp：项目测试入口文件
- 6. Makefile：构建c++项目的makefile文件



### 1.2 命令行构建运行

- 1. 项目构建：在bash终端(Mac OSX)中进入项目当前文件，输入：

make

- 2. 运行可执行文件：在bash终端(Mac OSX)进入src文件，输入命令：

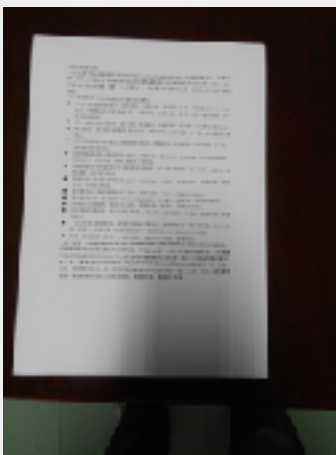

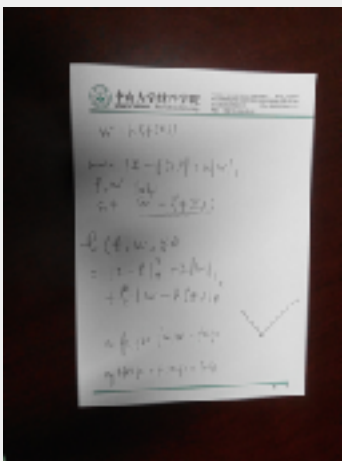
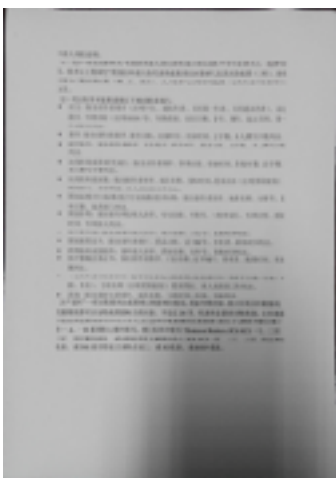
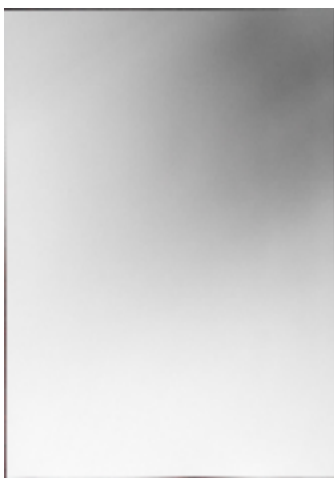
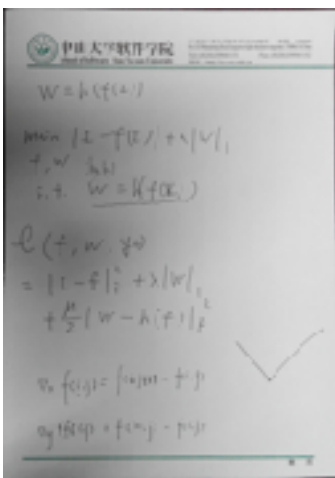
./cv\_hw4\_1


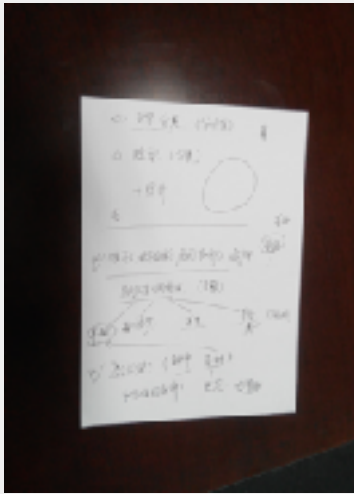
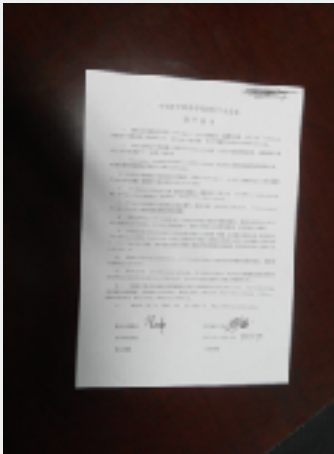
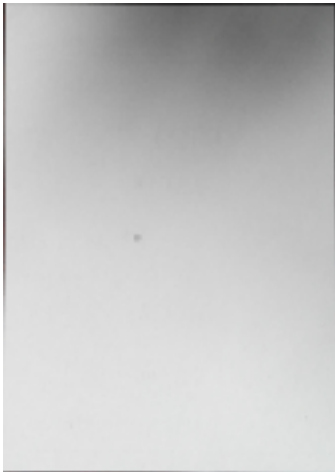
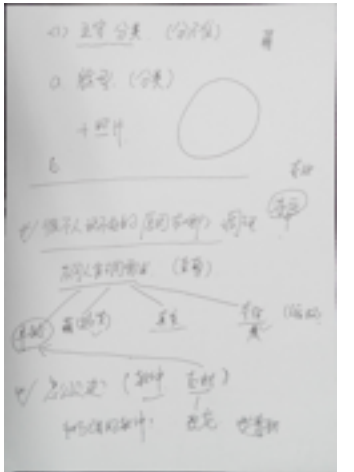
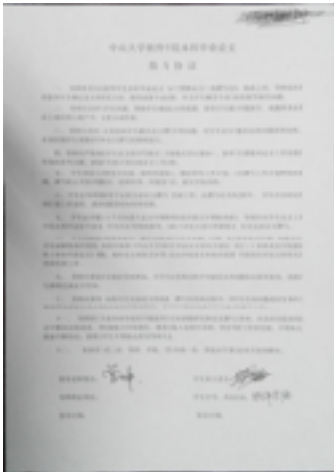
## 二、测试环境要求

测试系统	macOS High Sierra
编译器	g++

## 三、测试数据及结果

### 3.1 直线检测测试

	Dataset1/bmp/1.bmp	Dataset1/bmp/2.bmp	Dataset1/bmp/3.bmp
原图			
矫正结果			
	Dataset1/bmp/4.bmp	Dataset1/bmp/5.bmp	Dataset1/bmp/6.bmp

原图			
矫正结果			

# 四、测试分析及实验感想

## 1. 实验结果评价：

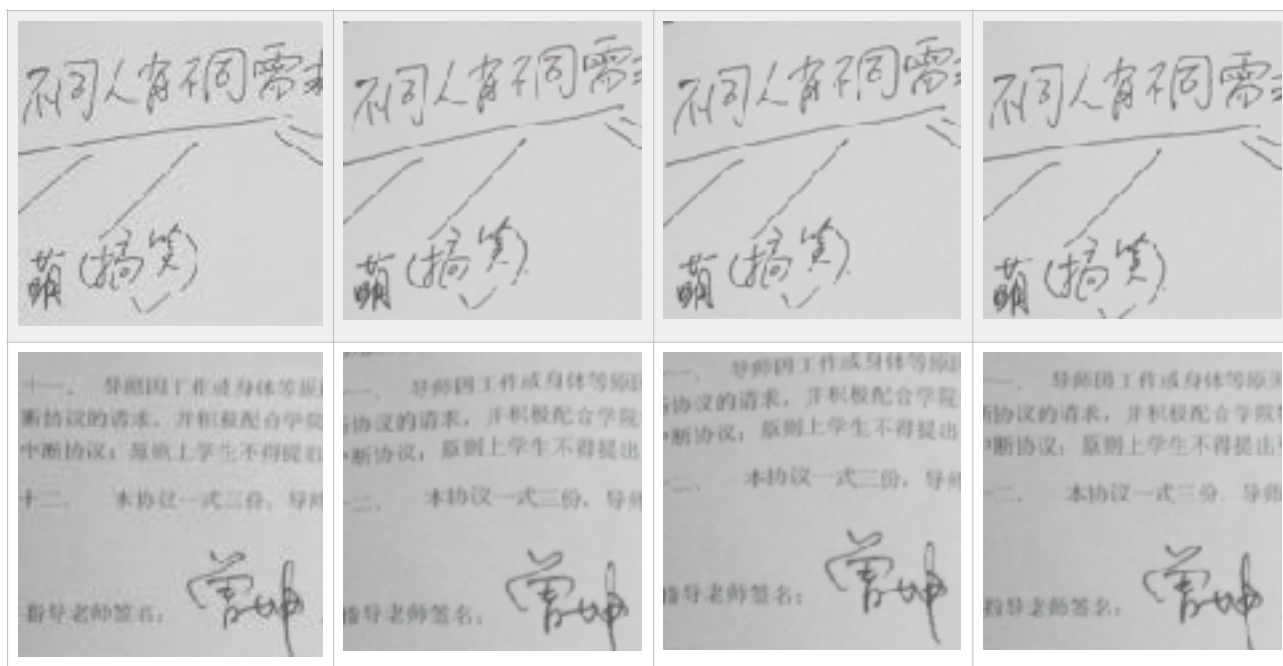
测试我使用了三种变形的方式：

Shear：Shear变换的实现是通过将4边形分成两个三角形分别求Shear变换矩阵，再反向做矫正  
Affine：Affine变换的实现与Shear变换类似，也是分成两个三角形，不同在于变换矩阵的形式多了两个自由度

Projective：Projective变换矩阵有八个自由度，所以这里我们需要四个点（刚好四个角点）列线性方程组，求解变换矩阵。

### a. 清晰度和矫正效果分析（500x500截取）

原图	Shear	Affine	Projective
			
			
			
			



Shear和Affine矫正结果导致平行的打印文章段落出现弯曲，特别由于将四边形分成两部分，导致三角形连接处的扭曲更加严重。

Projective矫正效果明显很好，恢复纸张上面的内容和前两种方法相比明显好很多，纸张上的打印段落矫正到平行，没有扭曲。

三种方法显示效果基本没有牺牲清晰度，也出现锯齿现象

## b. 时间复杂度分析

-O0	-O1
<pre> # Time complexity analysis for -O0 # ... (code) ...  [0] Line detecting costs 0.18748 sec. Shear Mapping costs 0.18411 sec. Affine Mapping costs 1.26948 sec. Projective Mapping costs 1.27045 sec.  [1] Line detecting costs 0.14679 sec. Shear Mapping costs 0.13911 sec. Affine Mapping costs 0.17666 sec. Projective Mapping costs 0.16661 sec.  [2] Line detecting costs 0.18880 sec. Shear Mapping costs 0.18704 sec. Affine Mapping costs 1.25444 sec. Projective Mapping costs 1.25286 sec.  [3] Line detecting costs 0.18831 sec. Shear Mapping costs 0.18672 sec. Affine Mapping costs 1.25659 sec. Projective Mapping costs 1.25799 sec.  [4] Line detecting costs 0.12811 sec. Shear Mapping costs 0.12866 sec. Affine Mapping costs 1.16119 sec. Projective Mapping costs 1.15896 sec.  [5] Line detecting costs 0.15647 sec. Shear Mapping costs 0.15670 sec. Affine Mapping costs 1.18020 sec. Projective Mapping costs 1.17893 sec. </pre>	<pre> # Time complexity analysis for -O1 # ... (code) ...  [0] Line detecting costs 0.18748 sec. Shear Mapping costs 0.18411 sec. Affine Mapping costs 1.26948 sec. Projective Mapping costs 1.27045 sec.  [1] Line detecting costs 0.14679 sec. Shear Mapping costs 0.13911 sec. Affine Mapping costs 0.17666 sec. Projective Mapping costs 0.16661 sec.  [2] Line detecting costs 0.18880 sec. Shear Mapping costs 0.18704 sec. Affine Mapping costs 1.25444 sec. Projective Mapping costs 1.25286 sec.  [3] Line detecting costs 0.18831 sec. Shear Mapping costs 0.18672 sec. Affine Mapping costs 1.25659 sec. Projective Mapping costs 1.25799 sec.  [4] Line detecting costs 0.12811 sec. Shear Mapping costs 0.12866 sec. Affine Mapping costs 1.16119 sec. Projective Mapping costs 1.15896 sec.  [5] Line detecting costs 0.15647 sec. Shear Mapping costs 0.15670 sec. Affine Mapping costs 1.18020 sec. Projective Mapping costs 1.17893 sec. </pre>

优化前，对于直线检测，基本可以达到0.2秒，而warp操作需要的时间在一秒左右，且与图像的大小有关

优化后，对于直线检测，基本可以达到0.06s作业，warp操作则需要0.3s左右，明显达到了实时矫正的要求

三种算法的时间复杂度基本相当

## 2. 感想:

1. Shear和Affine明显没有Projective效果那么好，Projective基本可以实现还原原来的A4纸。
2. CImg里面有很多好用的工具，用于解线性方程组，可以好好挖掘。
3. 对整数和浮点数的使用，严格区分清楚，部分接口如果要求浮点数，输入字面量1时一定要用1.0不然可能会发生不可预知的错误，或者0也是需要用0.0。
4. clang++和g++等c++编译器提供了很多优化可选项，要好好利用，可以显著提高运行时的速度，但可能牺牲一定的编译速度。