

# 《计算机视觉（1）》实验报告

## 实验一 图像的基本操作

实验小组成员 (学号+班级+姓名)	分工及主要完成任务	成绩
201800830004 18 数科 叶家辉	全部	

山东大学

2021 年 3 月

此次作业是为了确保学生能够读取图像，操纵像素，并显示结果。此作业必须独立完成。  
编程语言可以选择 MATLAB, C, C++, Python 等，可用 OpenCV。

实验完成主要任务：

- 1) 完成对图像的读取、保存操作；
- 2) 完成对输入图像平移操作；
- 3) 完成对输入图像旋转操作；
- 4) 完成对输入图像缩放操作；
- 5) 完成鼠标响应，实现鼠标在图像上移动（或点击图像）时输出该点的 RGB 值。
- 6) 设计 GUI 界面，并在后续的实验逐步修正完善。

C++做界面可用 MFC、Qt；Python 做界面可用 TKinter，还有其他的 gui，感兴趣搜搜；  
MATLAB 就用它的 GUI 编程。

这个基础实验如果用 OpenCV 或 MATLAB, 希望大家至少有一个图像处理的功能是不调用已有的函数的，自己编写函数实现。

原始图像请自行选择。实验报告写在如下空白处，页数不限，此次实验比较基础，实验报告可以重点写编程遇到的问题和解决思路。

## 一、实验目的与原理

此实验聚焦于图像的最基本操作，包括读取与保存、平移、旋转、缩放、鼠标响应读取图像 RGB 信息等内容，可谓“麻雀虽小，五脏俱全”，并通过图形化操作界面实现用户与程序间的交互。本报告主要通过 Matlab 实现，同时借助 Matlab 自带的 GUI 设计工具实现 GUI 的设计。

图像是由像素构成的，像素是构成图像的不可分割的最小单元。在计算机中，一幅图像可以看作是由若干像素点组成的二维矩阵。因此，对图像进行的操作就可以看作是对这个二维矩阵的操作。以平移为例，对图像进行平移其实就是对某个像素的原坐标 $(x_0, y_0)$ 进行如下的计算得到该像素的新坐标 $(x_1, y_1)$ ：

$$(x_1, y_1, 1) = (x_0, y_0, 1) \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ T_x & T_y & 1 \end{pmatrix}$$

其中 $T_x$ 是  $x$  方向偏移量， $T_y$ 是  $y$  方向偏移量。一般取向为  $x$  轴正方向，向右为  $y$  轴正方向。

对于图像的旋转，我们也可以推导出矩阵变换的公式。直觉上我们认为图像的旋转应该绕着中心点，然而图像默认是以左上角为原点的，所以我们首先要将图像左上角的原点移到中心点，具体方法是进行如下变换：

$$(x_1, y_1, 1) = (x_0, y_0, 1) \begin{pmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ -0.5W & 0.5H & 1 \end{pmatrix}$$

其中  $W$  为图像的宽， $H$  为图像的高。原坐标为 $(x_0, y_0)$ 的点经上述变换后得到坐标为 $(x_1, y_1)$ 的点。假设图像旋转角度（逆时针）为  $\theta$ ，设旋转后得到的坐标为 $(x_2, y_2)$ ，则应有如下变换：

$$(x_2, y_2, 1) = (x_1, y_1, 1) \begin{pmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

假设旋转后的图像宽为 $W'$ ，高为 $H'$ ，那么将坐标原点变回左上角需要进行如下变换：

$$(x_3, y_3, 1) = (x_2, y_2, 1) \begin{pmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0.5W' & 0.5H' & 1 \end{pmatrix}$$

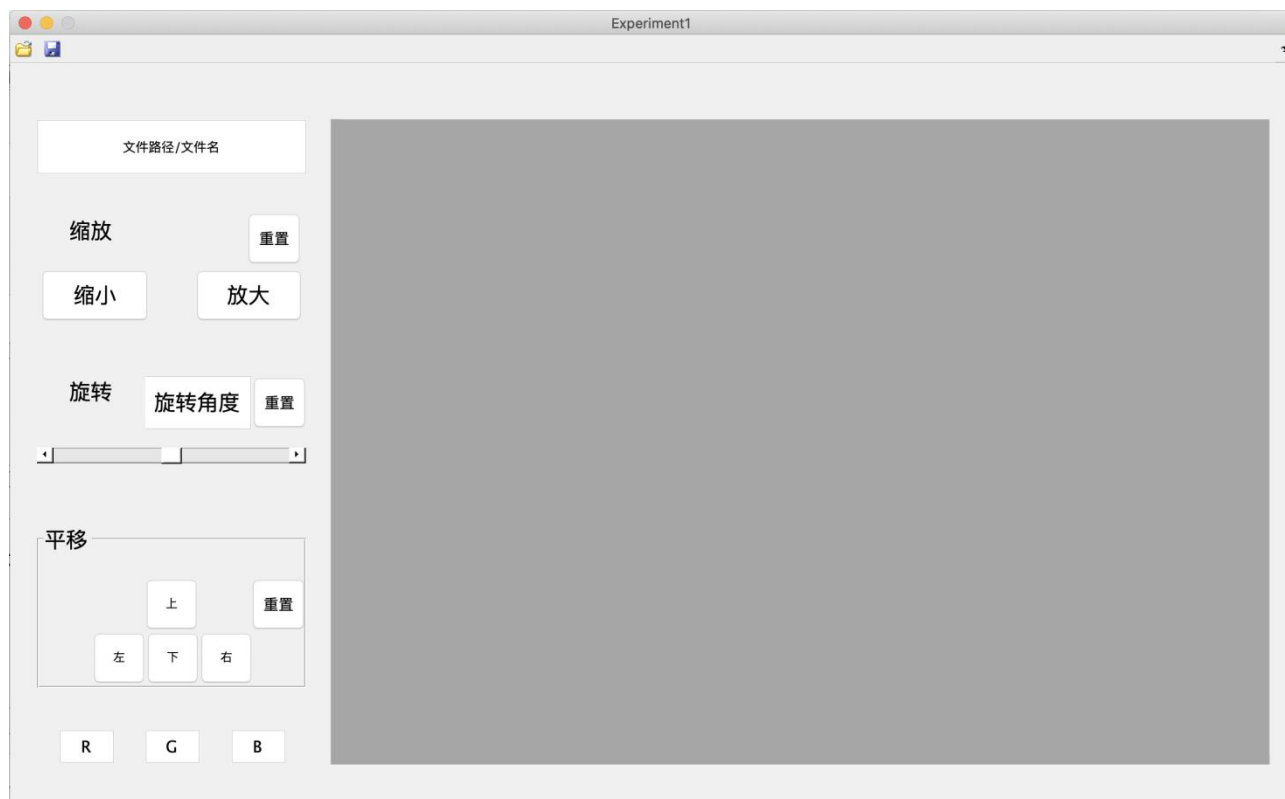
最后得到的 $(x_3, y_3)$ 即为图像 $(x_0, y_0)$ 的点旋转（逆时针） $\theta$ 后得到的坐标。

当然，现实世界的我们认为旋转是连续的，可是计算机对图像的处理却是离散的，这就导致旋转后新图像的某些像素值会有所缺失。这是我们会用到插值的方法对缺失的像素进行赋值。三种计算方法（包括最近邻插值、双线性插值和三次线性插值）及其原理上课的时候已经详细讨论过，在此不再赘述。

在图像的缩放过程中，我们可以很容易得到原图 $(x_0, y_0)$ 的点在新图中的坐标为 $(x_0 \times d, y_0 \times d)$ ，其中  $d$  为缩放倍率。但是和旋转的情况类似，新图中的像素点可能在原图中没有定义，因此我们也要进行插值操作对没有定义的点进行赋值。

## 二、实验内容与步骤

本次实验的主要内容分为三块，首先是图片的导入与保存，其次是对图像进行基于像素的操作，包括平移、旋转和缩放，最后是图像像素 RGB 信息的读取，并且这些功能需要编制于 GUI 界面上以响应用户的交互需求。程序的总体界面设计如下图所示。



### 1. 图片的导入与保存

为了符合大多数用户的操作习惯，我们将图片的导入与保存按钮放置在程序顶部导航栏。在 Matlab 的图形化界面窗口编辑器中点击工具栏编辑器，添加“打开”和“保存”两个按钮，并修改它们的回调函数以适应我们的需要。

```
function uipushtool5_ClickedCallback(hObject, eventdata, handles)
% hObject    handle to uipushtool5 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
[filename,pathname]=uigetfile({'*.bmp;*.jpg;*.png;*.jpeg;*.tif'}, '请选择图片');
str=[pathname filename];
OriginalPic = imread(str);
axes(handles.axes1);
imshow(OriginalPic);
set(handles.edit1, 'String', str);
```

其中“打开”按钮的回调函数修改如上。其表达的含义是：当用户点击顶部导航栏中的“打开”按钮时，程序会打开一个本地文件获取窗口，请用户选择本地的图片文件，然后获取用户所选文件的路径名和文件名（注意 `uigetfile()` 函数返回的文件名和路径名的顺序问题），用 `imread()` 获取具体的图片，并显示在坐标图区 `axes1`，并将左上角的可编辑文本框 `edit1` 内的文字设置为该图片的路径名和文件名。

```
function uipushtool6_ClickedCallback(hObject, eventdata, handles)
% hObject    handle to uipushtool6 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
h=getimage(handles.axes1);
imwrite(h,'output.bmp','bmp');
% cla(handles.axes1);
```

上图展示的则是“保存”按钮的回调函数。当用户按下“保存”按钮时，程序会读取当前坐标图区 `axes1` 内的图片，并将其写入当前路径下名为“output.bmp”的文件中。

## 2. 图片的平移

图片的平移主要通过用户对“上”“下”“左”“右”四个按键的点击实现，因此只需编写这四个按键的回调函数即可。用户每次点击按键，图像会在相应方向平移 10 个像素。下面我们以向上平移为例，其它方向均同理可得。

```
% --- Executes on button press in pushbutton3.
function pushbutton3_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% 向上平移10像素
OriginalPic=getimage(handles.axes1);
NewPic= imtranslate(OriginalPic,[0 -10]);
imshow(NewPic);
```

上图就是“上”按键的回调函数。首先通过 `getimage()` 函数得到坐标图区 `axes1` 内的图片信息，并命名为 `OriginalPic`，然后借助 Matlab 自带的 `imtranslate()` 函数进行平移得到新的图片。需要注意的是后面传入的数组所选取的坐标轴为以左上角为坐标原点，向右代表 X 轴正方向，向下代表 Y 轴正方向。因此向上平移应传入 `[0,-10]`，向下平移应传入 `[0,10]`，向左平移应传入 `[-10,0]`，向右平移应传入 `[10,0]`。

在平移控件组中还设置了“重置”按钮，帮助用户得到初始的图像。具体回调函数如下图所示。其基本原理就是获取左上角可编辑文本框 `edit1` 中的文件信息，进行重新读入与展示。



```

% --- Executes on button press in pushbutton9.
function pushbutton9_Callback(hObject, eventdata, handles)
% hObject      handle to pushbutton9 (see GCBO)
% eventdata    reserved - to be defined in a future version of MATLAB
% handles      structure with handles and user data (see GUIDATA)
str=get(handles.edit1,'String');
OriginalPic=imread(str);
imshow(OriginalPic);

```

### 3. 图片的旋转

图片旋转的部分可通过两种方式实现与用户的交互。用户可以通过拖动滑条定性地调整旋转角度，也可以直接在可编辑文本框 edit4 中定量输入旋转角度。（角度范围为-180 度到 180 度。）滑动条与可编辑文本框实现联动，即滑动条的读数可以在文本框中实时显示，反之文本框中输入的数值也可以实时定位到滑动条中的位置。

下图展示的是滑动条的回调函数。主要分为两个步骤，首先是读入滑动条 slider4 当前的数值 data，并将其转化为字符串后显示在文本 edit4 处，其次是将原图旋转角度 data 后进行显示。imrotate()函数中的‘bilinear’参数表明选用的内插方法为双线性插值。

```

% --- Executes on slider movement.
function slider4_Callback(hObject, eventdata, handles)
% hObject      handle to slider4 (see GCBO)
% eventdata    reserved - to be defined in a future version of MATLAB
% handles      structure with handles and user data (see GUIDATA)
data = get(handles.slider4,'Value');
set(handles.edit4,'string',num2str(data));
str=get(handles.edit1,'String');
OriginalPic=imread(str);
NewPic=imrotate(OriginalPic,data,'bilinear','crop');
imshow(NewPic);

```

下图展示的是可编辑文本框的回调函数。同样分为两个部分，首先是读取文本框中输入的字符串 data，将其转化为数值后设置为滑动条 slider4 的值，其次是将原图旋转角度 data（需转化为数值型）后进行显示。同样采用双线性插值的内插方法。

```

function edit4_Callback(hObject, eventdata, handles)
% hObject      handle to edit4 (see GCBO)
% eventdata    reserved - to be defined in a future version of MATLAB
% handles      structure with handles and user data (see GUIDATA)
data = get(handles.edit4,'string');
set(handles.slider4,'value',str2num(data));
str=get(handles.edit1,'String');
OriginalPic=imread(str);
NewPic=imrotate(OriginalPic,str2num(data),'bilinear','crop');
imshow(NewPic);

```

在旋转的部分也加入了“重置”按钮，具体的回调函数如下图所示。除了通过读取原文件的信息并将其重新显示在坐标图区以外，还需要将滑动条和文本框的显示内容进行归零。

```
% --- Executes on button press in pushbutton8.
function pushbutton8_Callback(hObject, eventdata, handles)
% hObject      handle to pushbutton8 (see GCBO)
% eventdata    reserved - to be defined in a future version of MATLAB
% handles      structure with handles and user data (see GUIDATA)
set(handles.slider4,'value',0);
set(handles.edit4,'string','0');
str=get(handles.edit1,'String');
OriginalPic=imread(str);
imshow(OriginalPic);
```

#### 4. 图片的缩放

为了实现图片的缩放功能，我们首先使用的是 `imresize()` 函数，可以方便地实现对图片的缩放功能，但是实际操作中却发现，图片还是呈现原大小，并没有得到缩放，这十分令人困惑。接下来的实验中发现了两件事：第一，当我们将放缩后的图片进行保存时，发现图片尺寸确实改变了；第二，当我们将图片进行放缩时，虽然图片显示的大小没有改变，但是分辨率改变了，“缩小”时变得分辨率降低，“放大”时分辨率提高。这都证明了一件事，图片确实得到了放缩。那就只有一种解释，坐标图区的大小限制了图片显示的大小。之所以只有分辨率的区别，是因为实际发生的过程相当于把不同尺寸的图片显示在相同大小的区域。因此在进行缩放时，除了要对图片本身进行尺寸上的缩放，还要对坐标图区的大小进行调整。下图为“缩小”按键的回调函数。我们假定每按一次“缩小”按键，就对图片的长宽各减少 0.02 个单位（放大操作亦然）。首先需要获取当前坐标图区 `axes1` 的位置并存入变量 `a`。`a` 是一个四维的向量，其中第三维为图像的宽度，因此图像应缩放到原来的  $\frac{a(3)-0.02}{a(3)}$  倍，其中内插操作选择的是三次线性插值。同时，图像显示区域 `axes1` 与 `figure` 上侧和左侧的距离各增加 0.01 个单位，`axes1` 的宽度和高度各减少 0.02 个单位。对于放大操作也是类似，只是将图像缩放到原来的  $\frac{a(3)+0.02}{a(3)}$  倍，同时将 `axes1` 与 `figure` 上侧和左侧的距离各减少 0.01 个单位，`axes1` 的宽度和高度各增加 0.02 个单位。

```
% --- Executes on button press in pushbutton7.
function pushbutton7_Callback(hObject, eventdata, handles)
% hObject      handle to pushbutton7 (see GCBO)
% eventdata    reserved - to be defined in a future version of MATLAB
% handles      structure with handles and user data (see GUIDATA)

a=get(handles.axes1,'position');
OriginalPic=getimage(handles.axes1);
NewPic=imresize(OriginalPic,(a(3)-0.02)/a(3),'bicubic');
imshow(NewPic);
set(handles.axes1,'units','normalized','position',[a(1)+0.01 a(2)+0.01 a(3)-0.02 a(4)-0.02]);
```

同样地，缩放模块也有“重置”按钮。它所做的事情很简单，就是把坐标图区的大小设置为原大小，并重新读入原图进行显示。

5. 像素 RGB 鼠标响应


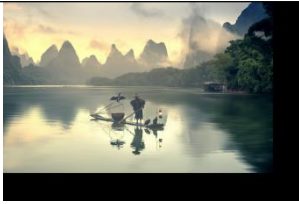


为了在用户点击鼠标时读取相应像素点的 RGB 信息，需要添加一个鼠标响应。下图为鼠标响应的回调函数。首先得到鼠标当前的坐标，并将双精度浮点型 double 转化为整型 int。这里一开始由于不太清楚图像的坐标范围和 Matlab 中不同 int 类型的表数范围，出现了溢出的情况，而在后来的编写中调整为 int64()问题就得到了解决。然后得到当前坐标图区图片的信息，要注意的是鼠标的坐标轴是[x,y]，而 Matlab 中图像存储时前两维是[y,x]，这一点一开始导致了索引超出范围，后来通过在命令行中读入图像观察其矩阵的维数发现了这一问题。然后通过 R，G，B 三个变量存储该坐标下三个通道内的值，并将其转化为字符串后写入到 edit5，edit6，edit7 三个可编辑文本框中。

```
% --- Executes during object creation, after setting all properties.
function figure1_WindowButtonDownFcn(hObject, eventdata, handles)
% hObject      handle to figure1 (see GCBO)
% eventdata    reserved - to be defined in a future version of MATLAB
% handles      empty - handles not created until after all CreateFcns called
[x,y]=ginput(1);
x=int64(x(1,1));
y=int64(y(1,1));
OriginalPic=getimage(handles.axes1);
R=OriginalPic(y,x,1);
G=OriginalPic(y,x,2);
B=OriginalPic(y,x,3);
% fprintf('R=%d,G=%d,B=%d\n',R,G,B);

set(handles.edit5,"string",num2str(R));
set(handles.edit6,"string",num2str(G));
set(handles.edit7,"string",num2str(B));
```

三、实验结果与呈现

以下呈现的效果图均为程序内处理后保存的图。

原图	平移	旋转	缩放
			

更多操作展示请见演示视频。