电 子 科 技 大 学 实 验 报 告

课程名称： 数学实验

实验地点： 基础实验大楼227

指导教师：

评 分：

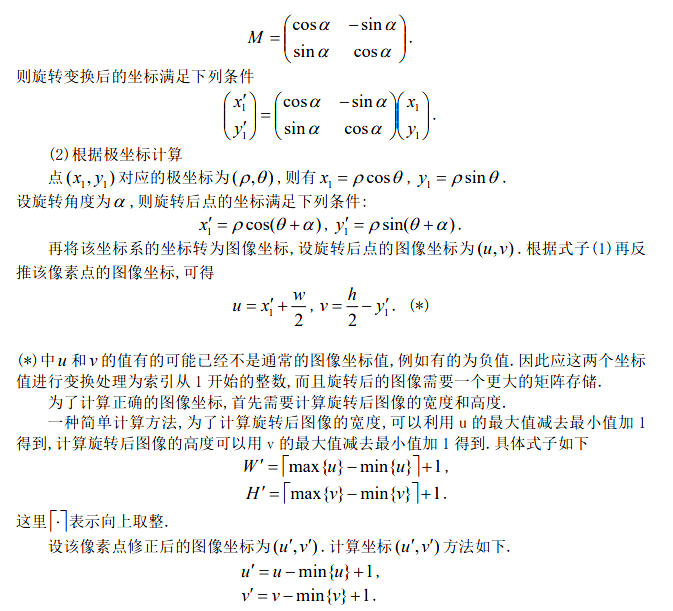
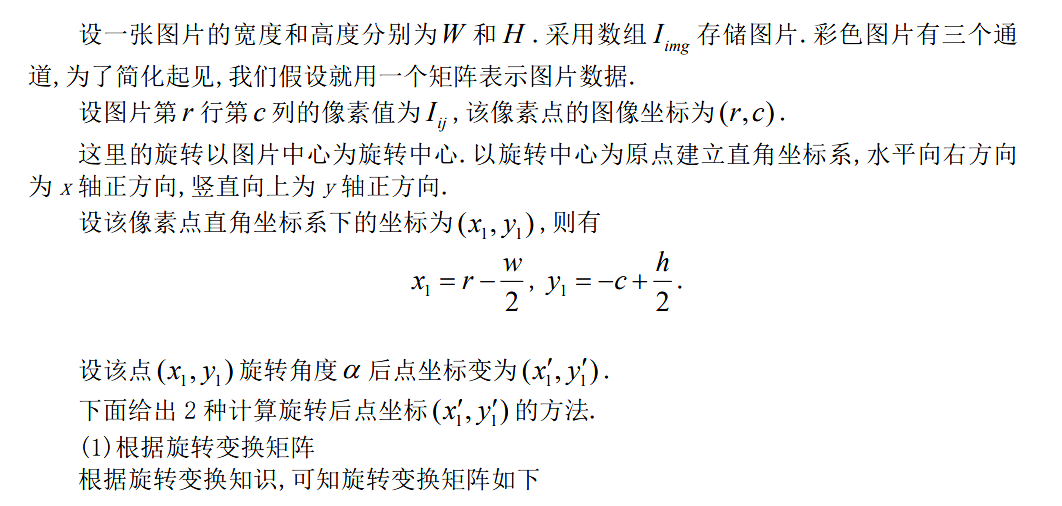
完成实验学生信息：

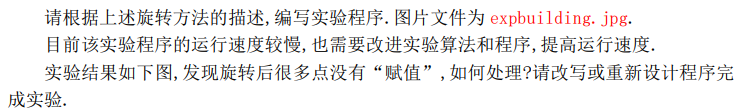
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **姓名** | **学号** | **贡献百分比**  **(%)** | **备注**  **（主要工作**） |
| 1 |  |  |  |  |
| 2 |  |  |  |  |
| 3 |  |  |  |  |

1. 学生人数按照任课教师要求限定；
2. 对于“评价、改进、总结和体会”都要认真填写，和其他内容是评价实验成绩的重要参考。

**实验题目名称: 图片旋转的实验**

1. **实验内容**





1. **实验目的**

* 掌握图片旋转坐标计算方法
* 熟悉矩阵变换相关算法
* 程序设计与分析的应用

1. **实验过程**

具体实验过程见正文部分。

**目 录**

[1. 实验：图片旋转的实验 4](#_Toc91781245)

[1.1. 问题分析 4](#_Toc91781246)

[1.2. 模型假设 5](#_Toc91781247)

[1.3. 变量与符号说明 5](#_Toc91781248)

[1.4. 模型建立与求解(实验过程) 5](#_Toc91781249)

[1.4.1. 正向映射法的建模与求解 5](#_Toc91781250)

[1.4.2. 反向映射法的建模与求解 6](#_Toc91781251)

[1.5. 实验结果分析 8](#_Toc91781252)

[1.6. 优缺点及改进方向 8](#_Toc91781253)

[1.7. 心得体会与总结 8](#_Toc91781254)

[2. 对本次实验的设计提出改进意见 9](#_Toc91781255)

[附件 9](#_Toc91781256)

[附件1.两种方法的MATLAB程序 9](#_Toc91781257)

# 实验：图片旋转的实验

## 问题分析

通过查找相关资料我们了解到，在当代使用的相关计算机系统中，图像是通过二维矩阵存放的，其形象的来说是由一个个像素点拼接而成。因此，在计算机中图像旋转的本质即为向量的旋转：我们可以将图象的一个像素点作为旋转轴，然后将其他的像素点看作是从这个旋转轴出发的向量，因此图像的旋转即为一个个像素点对应向量的旋转，下图可以更加形象地说明。



**旋转轴点**

图1 旋转图片的原理解释

假设现有一张图片的宽度和高度分别为W和H。根据上文的分析，我们就可以采用数组Iimg来存储这张图片。经过查阅资料可知彩色图片有三个通道,为了简化起见,我们可以假设只用一个矩阵表示图片数据。

设图片第r行第c列的像素值为I ij，那么该像素点的图像坐标为 (r, c) 。这里我们选取旋转的轴点为图片中心（便于坐标系的建立以及后文的相关计算）。以旋转中心为原点建立直角坐标系，水平向右方向 为 x 轴正方向，竖直向上为 y 轴正方向，设该像素点直角坐标系下的坐标为(x1,y1)，则：



设该点(x1,y1)旋转角度α后点坐标变为(x1’,y1’)。

题目中给出了两种旋转的计算方案，一种为根据极坐标计算，一种为根据旋转变换矩阵计算。由于使用变换矩阵旋转变换的方法更具有可行性，因此下文的分析计算及程序编写全部基于使用变换矩阵旋转变换的方法。

## 模型假设

* 假设我们所需要旋转的图片为jpg格式，即为多个像素点组成。
* 假设彩色图片也能使用一个矩阵来完全表示其图片数据。
* 假设旋转后图像的长度和宽度均用(W\*sin(a)+H\*cos(a))+1的向下取整值来代替。

## 变量与符号说明

|  |  |
| --- | --- |
| **变量名** | **符号** |
| 原始图像的长度 | W |
| 原始图像的宽度 | H |
| 旋转变换矩阵 | M |
| 旋转角度 | α |
| 相关点的坐标 | x1，y1，x1’，y1’ |

## 模型建立与求解(实验过程)

根据线性代数的相关知识，我们可以求得变换的旋转矩阵如下：



那么变换后的坐标满足下列条件



根据上述原理，我们可以设计出两种旋转方法：一种为正向映射法，一种为改进后的反向映射法，下面我们将更加细致的解释这两种旋转方法，并最后将其与MATLAB自带的旋转图像功能进行旋转后的图象分析比较。

### 正向映射法的建模与求解

正向映射法是最为简单，直观的。它的原理是即根据原图的像素点坐标推导出旋转图像对应点坐标，然后直接将原图坐标赋给x1,y1，根据相关的旋转矩阵将数组表示的源图片转化为新的数组，并由此得到新的图片。

根据这样的思路以及题目所给的参考代码，我们写出了相关代码（具体见附件），并运行得到了旋转过后的图片。图片如下：



图2 正向映射法旋转图片的结果

然而我们能看到这种方法的瑕疵：图像中似乎会出现很多“噪声”、“杂点”，我们可以将这些点解释为没有“赋值”：从原图旋转后的像素位置在原图可能找不到，而对于他们又没有弥补的方法，就会出现图中所出现的噪声（这些噪声点的分布似乎是有规律的也是因为这个原因）。

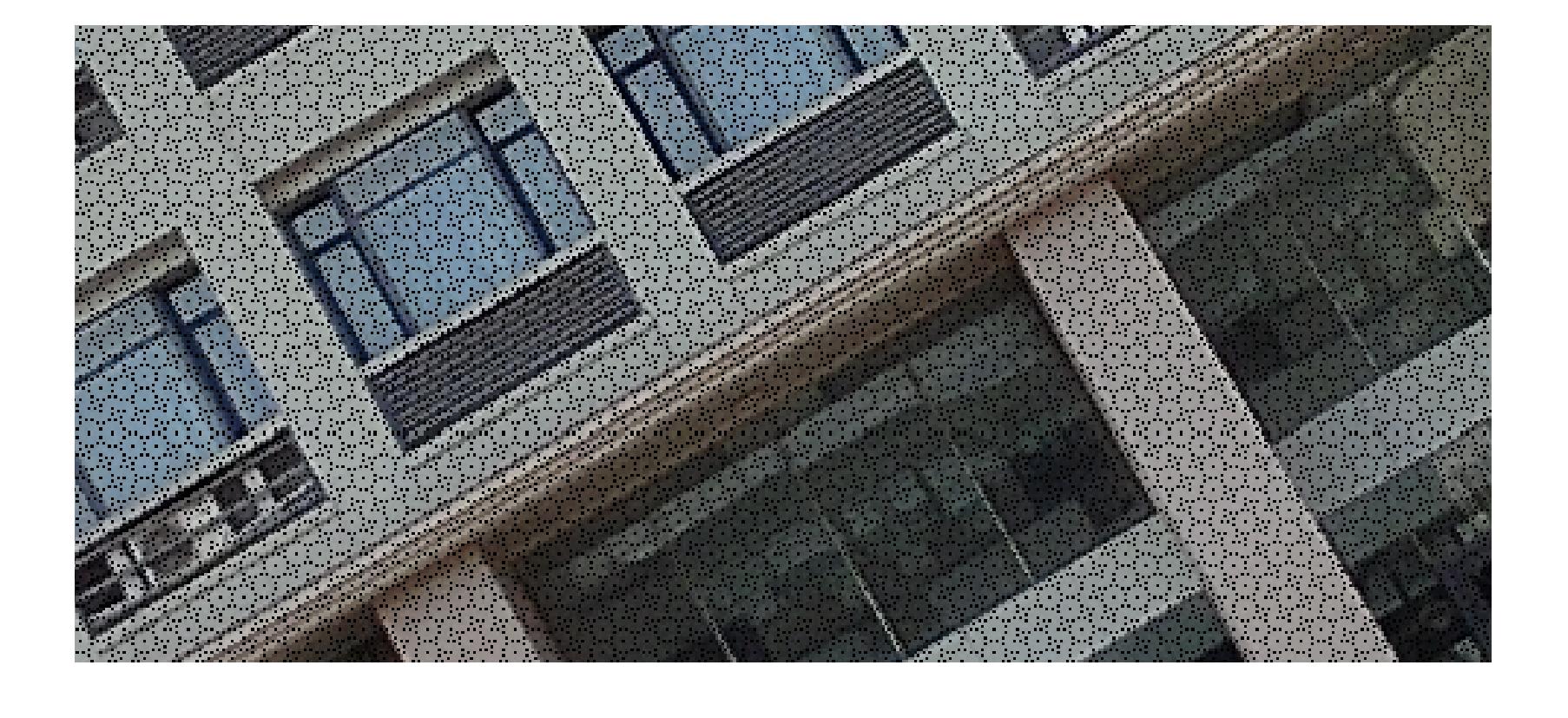


图3 正向映射法旋转图片的缺陷（局部放大图）

### 反向映射法的建模与求解

由于上述的正向映射法是有缺陷的，因此我们考虑更加优化的算法。

经过思考和查阅资料，我们采用逆向思维来设计算法：正向映射法的方法缺陷在于旋转后的图片的一些像素位置在原图中找不到，因此会呈现出黑色的“空白”，那么如果我们逆向进行上述过程，对于旋转后找不到位置的像素坐标，我们取原图上距离其最近的像素对应的值来赋给找不到位置的像素坐标，这样尽管图片的清晰性可能有所下降，但从原理上来说应当不会再出现正向映射法中的噪点现象。



图3 反向映射法旋转图片的结果

通过观察旋转后图片的局部放大图，我们能看到使用这种方法不再会出现正向映射法的噪点，同时图片的清晰度也相应的下降了一些，这和我们的分析完全一致（相关代码见附件）。

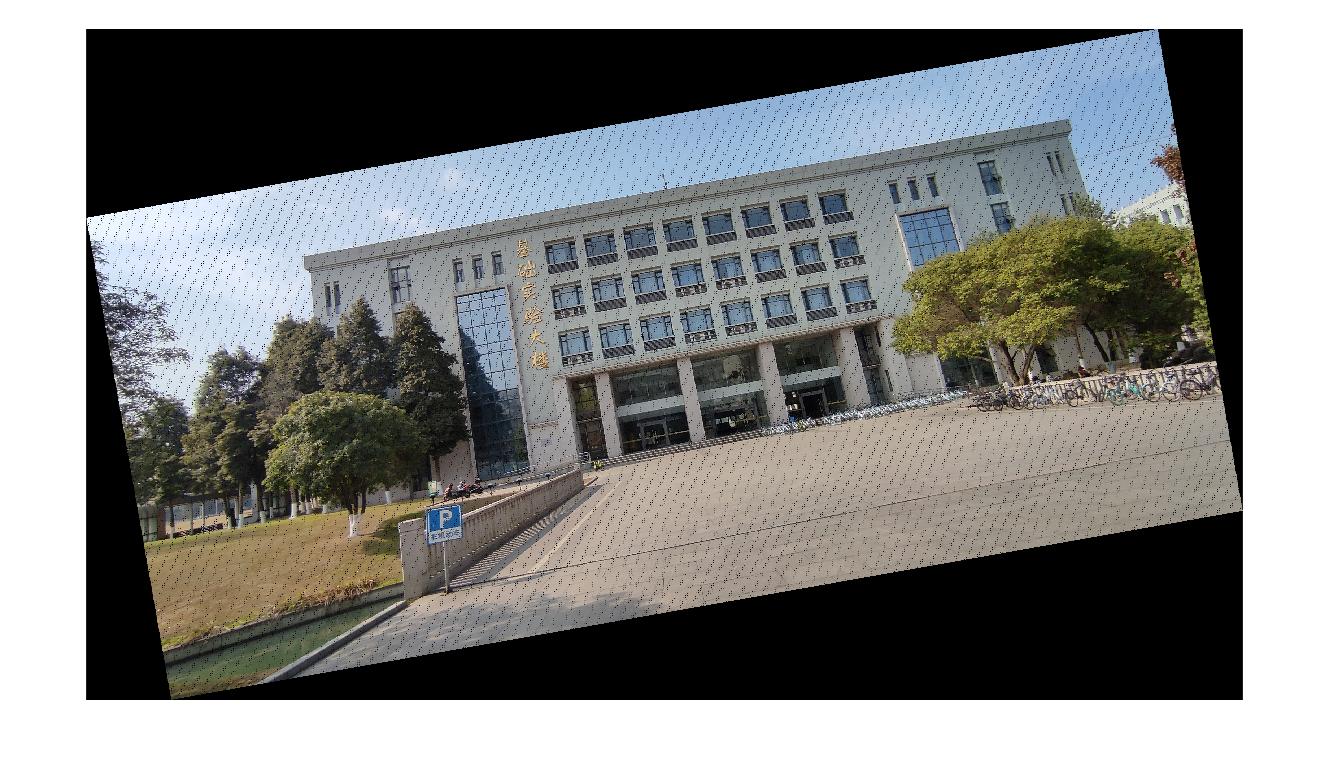
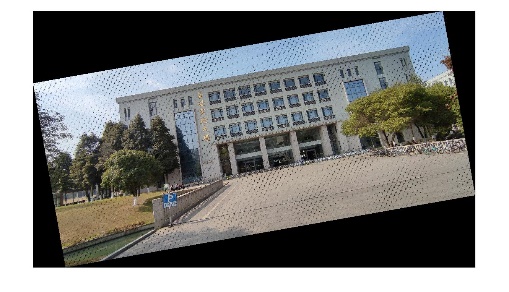


图4 反向映射法旋转图片的局部放大图

此外，我们在参考代码的基础上进行了相关改进：对于一些循环的写法进行了一些优化，具体的相关改进可以见附件中的代码部分。最终，通过tic、toc测试，参考代码的运行时间为15.821938 秒，经过改进的实验代码的运行时间为6.521506 秒，完成了对运行速度的提高。

## 实验结果分析

最后我们将正向映射法，反向映射法和MATLAB自带的旋转图片进行列举比较。



上述三个图从左到右依次为：正向映射法，反向映射法和MATLAB自带的旋转图片方法得到的图片。

## 优缺点及改进方向

* 优点：较为完美实现图片旋转，像素点损失较少，极大程度保留了图片原有像素；相较于参考代码提高了运行速度；代码较简洁。
* 缺点：使用直接法计算旋转后的坐标，时间复杂度较高，程序运行较慢；存在极少量像素缺失，采用附近像素点填充导致旋转后图片与原图片不完全一致。

改进方向：提高坐标精度，如使用双浮点数代替单浮点数，减少像素损失；采用新的方法计算坐标变换，降低计算的时间复杂度，如采用累加法替代直接法。

## 心得体会与总结

本次综合性实验是对图片旋转进行操作，使我们熟悉了MATLAB自带旋转函数的使用方法，同时掌握了利用线性代数矩阵变化进行图片旋转的技巧，学会了求解类似问题的新的思路。本次实验我们通过大量查阅相关资料如百度、csdn以及老师提供的资料，尝试对算法进行优化。本实验具有一定的趣味性和难度，加深了我们对线性代数矩阵变换相关知识的理解，并能加深我们对其综合运用的能力，使以前学到的理论与实践相结合。  
在实验最初，我们尝试利用老师提供的正向旋转的思路，即把原图进行向量的旋转，找到旋转后的向量的位置，然后将原图的像素值赋值过去；但该方法会使图像中出现很多分布有规律的“噪声”、“杂点”，能找不到，而对于他们又没有弥补的方法，就会出现图中所出现的噪声。之后，为了减少出现的噪声，以及得到完整的旋转图像，我们采用了用逆向思维，从目标图片反向旋转到原图进行像素查找。

# 对本次实验的设计提出改进意见

1. 可以围绕两种旋转方法（矩阵运算、极坐标）提供不同的参考代码；
2. 增加实验任务，比如实现图片镜像翻转。

# 附件

## 附件1.两种方法的MATLAB程序

%正向映射法

function testmain\_imrotate

close all

filename = 'expbuilding.jpg';

srcimg = imread(filename);

angle = 10; %单位:度. 顺时针旋转 angle 为负值,逆时针为正值.

Mnew = func\_imrotate(srcimg,angle); %% 运行速度较慢.这个方面也需要改进.

imshow(Mnew)

function Mnew = func\_imrotate(img,angle)

a = angle\*pi/180;%旋转角度

T = [cos(a),-sin(a);sin(a),cos(a)]; % 构造变换矩阵

[H,W,~] = size(img); % H 为图片高度, W 为图片宽度

[c,r] = meshgrid(1:W,1:H); % 产生所有点的图像坐标. 此步骤很关键.

% 上一行确保 r,c 为遍历图片各行各列像素点的 图像坐标,同时也是矩阵的索引.

X1 = c-W/2; Y1 = -r+H/2; %转为(x,y)坐标.以图片中心为原点

P = [X1(:)';Y1(:)'];

Pnew = T\*P; %旋转变换. Pnew 为旋转后点的坐标

X1new = reshape(Pnew(1,:),H,W); % 重新把 x 坐标 变换为一个矩阵

Y1new = reshape(Pnew(2,:),H,W); % 重新把 y 坐标 变换为一个矩阵

Hnew = floor(W\*sin(a)+H\*cos(a))+1; % 计算旋转后图片的高度

Wnew = floor(W\*cos(a)+H\*sin(a))+1; % 计算旋转后图片的宽度

u = round(X1new);

v = round(-Y1new);

Mnew = uint8(zeros(Hnew,Wnew,3));

c2 = u - min(min(u)) + 1;

r2 = v - min(min(v)) + 1;

for i=1:size(r2,1)

for j=1:size(r2,2)

Mnew(r2(i,j),c2(i,j),1:3) = img(i,j,1:3);

end

end

%反向映射法

% 读入图片

im = imread('expbuilding.jpg');

% 求出旋转矩阵

a = 30 / 180 \* pi;

R = [cos(a), -sin(a); sin(a), cos(a)];

R = R'; % 求出旋转矩阵的逆矩阵进行逆向查找

% 计算原图大小

sz = size(im);

h = sz(1);

w = sz(2);

ch = sz(3);

c1 = [h; w] / 2;

% 计算显示完整图像需要的画布大小

hh = floor(w\*sin(a)+h\*cos(a))+1;

ww = floor(w\*cos(a)+h\*sin(a))+1;

c2 = [hh; ww] / 2;

% 初始化目标画布

im2 = uint8(ones(hh, ww, 3)\*128);

for k = 1:ch

for i = 1:hh

for j = 1:ww

p = [i; j];

pp = round(R\*(p-c2)+c1);

% 逆向进行像素查找

if (pp(1) >= 1 && pp(1) <= h && pp(2) >= 1 && pp(2) <= w)

im2(i, j, k) = im(pp(1), pp(2), k);

end

end

end

end

% 显示图像

figure;

imshow(im2);