Lab1 Report

Zepeng Chen

June 1, 2021

Contents

1 Basic Transformation

1.1 Code Except Translation

```
%function newImg=transImg(theta,tx,ty,cx,cy,sv,sh, operation)

figure(1)

img1=transImg(0,0,0,0,0,’rotate’);
imshow(img1,[])
title("original image")
print(’origin’,’-dpng’);

figure(2)

img2=transImg(0.25*pi,0,0,0,0,’rotate’);
imshow(img2,[])
title("rotated 45 degree image")
print(’rot45’,’-dpng’);

figure(3)

img3=transImg(0.5*pi,0,0,0,0,’rotate’);
imshow(img3,[])
title("rotated 90 degree image")
print(’rot90’,’-dpng’);

figure(4)

img4=transImg(-0.1389*pi,0,0,0,0,’rotate’);
imshow(img4,[])
title("rotated -25 degree image")
print(’rot-25’,’-dpng’);

figure(5)

img5=transImg(0,0.25,0.75,0,0,’scale’);
imshow(img5,[])
```

```
title("scale image with cx=0.5, cy=1.5")
print('scale','-dpng');

figure(6)

img6=transImg(0,0,0,0.2,0,'shearV');
imshow(img6,[])
title("vertically sheared image with sv=0.2")
print('Vshear','-dpng');

figure(7)

img7=transImg(0,0,0,0,0.3,'shearH');
imshow(img7,[])
title("Horizontally sheared image with sv=0.3")
print('Hshear','-dpng');

function tImg=transImg(theta,cx,cy,sv,sh, operation)

transChoice=operation;
switch transChoice
    case 'rotate'
        T=[cos(theta), sin(theta), 0;
          -sin(theta), cos(theta), 0;
          0, 0, 1];

    case 'scale'
        T=[cx, 0, 0;
           0, cy, 0;
           0, 0, 1];

    case 'shearV'
        T=[1, sv, 0;
           0, 1, 0;
           0, 0, 1];
    case 'shearH'
        T=[1, 0, 0;
           sh, 1, 0;
           0, 0, 1];
end


imc=imread('im1.png');
img=rgb2gray(imc);

[xMax,yMax]=size(img);
corners=[0, 0, 1;
    xMax,0,1;
    0,yMax,1;
    xMax,yMax,1];
newCorners=corners*T;
xmin=min(newCorners(:,1));
xmax=max(newCorners(:,1));
newWidth=round(xmax-xmin);
ymin=min(newCorners(:,2));
ymax=max(newCorners(:,2));
newHeight=round(ymax-ymin);
tImg=zeros(newWidth, newHeight);

for i=1:newWidth
    for j=1:newHeight
        xOffset=round(xmin);
        yOffset=round(ymin);
        temp=[i+xOffset,j+yOffset,1]/T;
        x=round(temp(1));
        y=round(temp(2));
        if x>0&&x<xMax&&y>0&&y<yMax
            %assign the nearest point value to transformed point
            tImg(i,j)=img(x,y);
        end
    end
```

```
end
end
```

1.2 Code For Translation

```
%translation operation is a bit different to the universal transform function
img=imgTranslate(50,45);
figure(1)
imshow(img,[]);
print('trans50-45','-dpng');
function imgTranslation=imgTranslate(tx,ty)
      imc=imread('im1.png');
      img=rgb2gray(imc);
      [xMax,yMax]=size(img);
      imgTranslation=zeros(xMax+tx,yMax+ty);
      for i=1:xMax
          for j=1:yMax
              imgTranslation(i+tx,j+ty)=img(i,j);
          end
      end
end
```

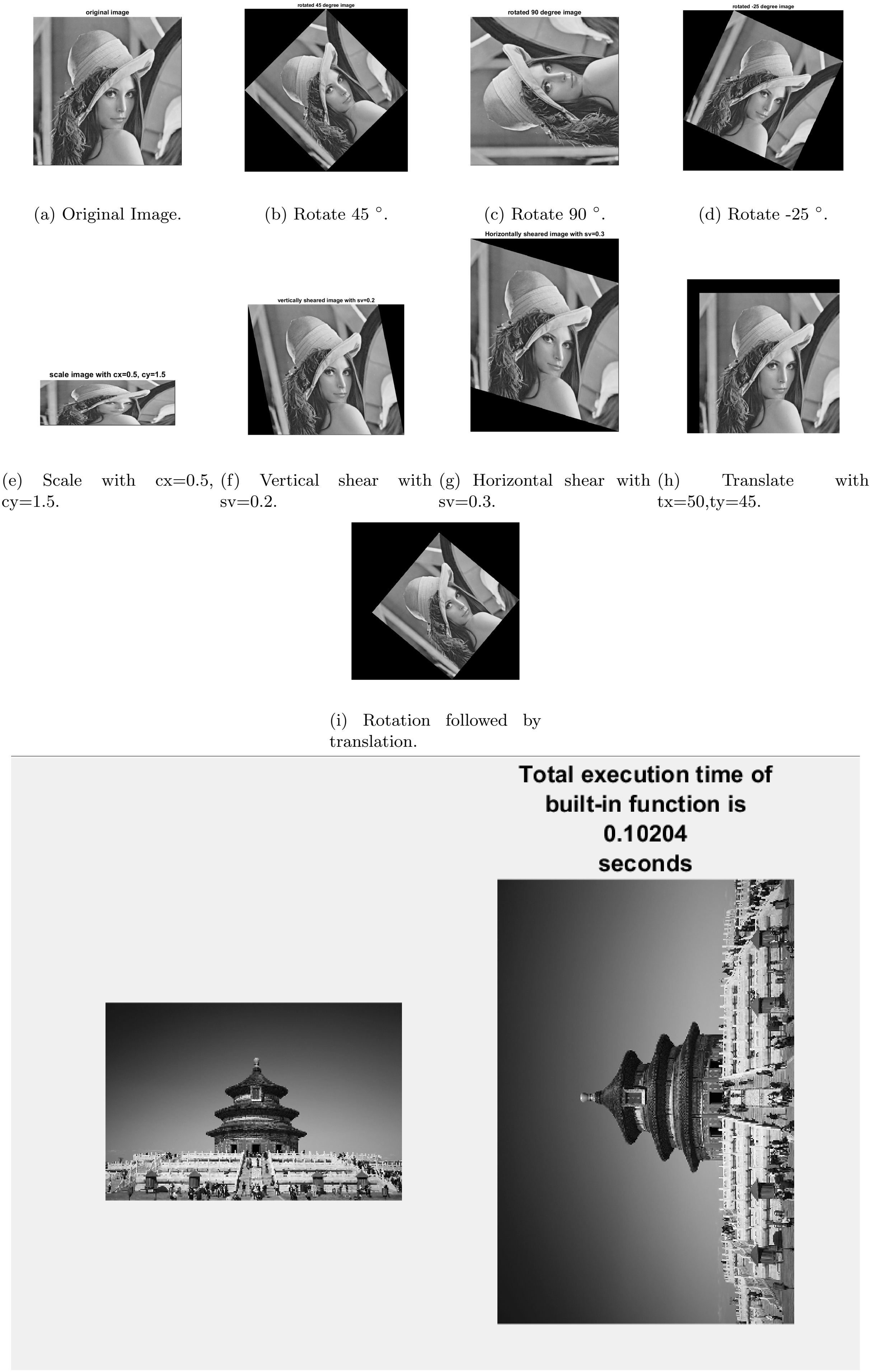
1.3 Code For Rotation Followed by Translation

```
img=rotNtrans(0.2778*pi,50,100);
figure(1)
imshow(img,[]);
print('rotAndtrans','-dpng')

function tImg=rotNtrans(theta,tx,ty)
imc=imread('im1.png');
img=rgb2gray(imc);

 T=[cos(theta), sin(theta), 0;
    -sin(theta), cos(theta), 0;
         0, 0, 1];
[xMax,yMax]=size(img);
corners=[0, 0, 1;
    xMax,0,1;
    0,yMax,1;
    xMax,yMax,1];
newCorners=corners*T;
xmin=min(newCorners(:,1));
xmax=max(newCorners(:,1));
newWidth=round(xmax-xmin);
ymin=min(newCorners(:,2));
ymax=max(newCorners(:,2));
newHeight=round(ymax-ymin);
tImg=zeros(newWidth+tx, newHeight+ty);

for i=1:newWidth
    for j=1:newHeight
        xOffset=round(xmin);
        yOffset=round(ymin);
        temp=[i+xOffset,j+yOffset,1]/T;
        x=round(temp(1));
        y=round(temp(2));
        if x>0&&x<xMax&&y>0&&y<yMax
            %assign the nearest point value to transformed point
            tImg(i+tx,j+ty)=img(x,y);
        end
    end
end
end
```

1.4 Code For Comparison between Built-in and Self-defined Rotation

```
tic

imc=imread('tiantan.jpg');
img=rgb2gray(imc);
subplot(1,2,1)
imshow(img)
theta = 90;
tform = affine2d([cosd(theta) -sind(theta) 0;
                sind(theta) cosd(theta) 0;
                0 0 1]);

outputImage = imwarp(img,tform);
subplot(1,2,2)
imshow(outputImage);

%translation operation is a bit different to the universal transform function

%{
subplot(1,2,1)
imshow(img)

theta=0.5*pi;
imc=imread('tiantan.jpg');
img=rgb2gray(imc);
T=[cos(theta), sin(theta), 0;
   -sin(theta), cos(theta), 0;
         0, 0, 1];
[xMax,yMax]=size(img);
corners=[0, 0, 1;
   xMax,0,1;
   0,yMax,1;
   xMax,yMax,1];
newCorners=corners*T;
xmin=min(newCorners(:,1));
xmax=max(newCorners(:,1));
newWidth=round(xmax-xmin);
ymin=min(newCorners(:,2));
ymax=max(newCorners(:,2));
newHeight=round(ymax-ymin);
tImg=zeros(newWidth, newHeight);

for i=1:newWidth
    for j=1:newHeight
        xOffset=round(xmin);
        yOffset=round(ymin);
        temp=[i+xOffset,j+yOffset,1]/T;
        x=round(temp(1));
        y=round(temp(2));
        if x>0&&x<xMax&&y>0&&y<yMax
            %assign the nearest point value to transformed point
            tImg(i,j)=img(x,y);
        end
    end
end
subplot(1,2,2)
imshow(tImg,[]);
%}
toc
title({'Total execution time of';
'self function is';num2str(toc);'seconds'});
```

1.5 Outcome Set I


original image
rotated 45 degree image
rotated 90 degree image
rotated -25 degree image
scale image with cx=0.5, cy=1.5
vertically sheared image with sv=0.2
Horizontally sheared image with sv=0.3
Total execution time of built-in function is 0.10204 seconds

(a) Original Image.

(b) Rotate 45 °.

(c) Rotate 90 °.

(d) Rotate -25 °.

(e) Scale with cx=0.5, cy=1.5.

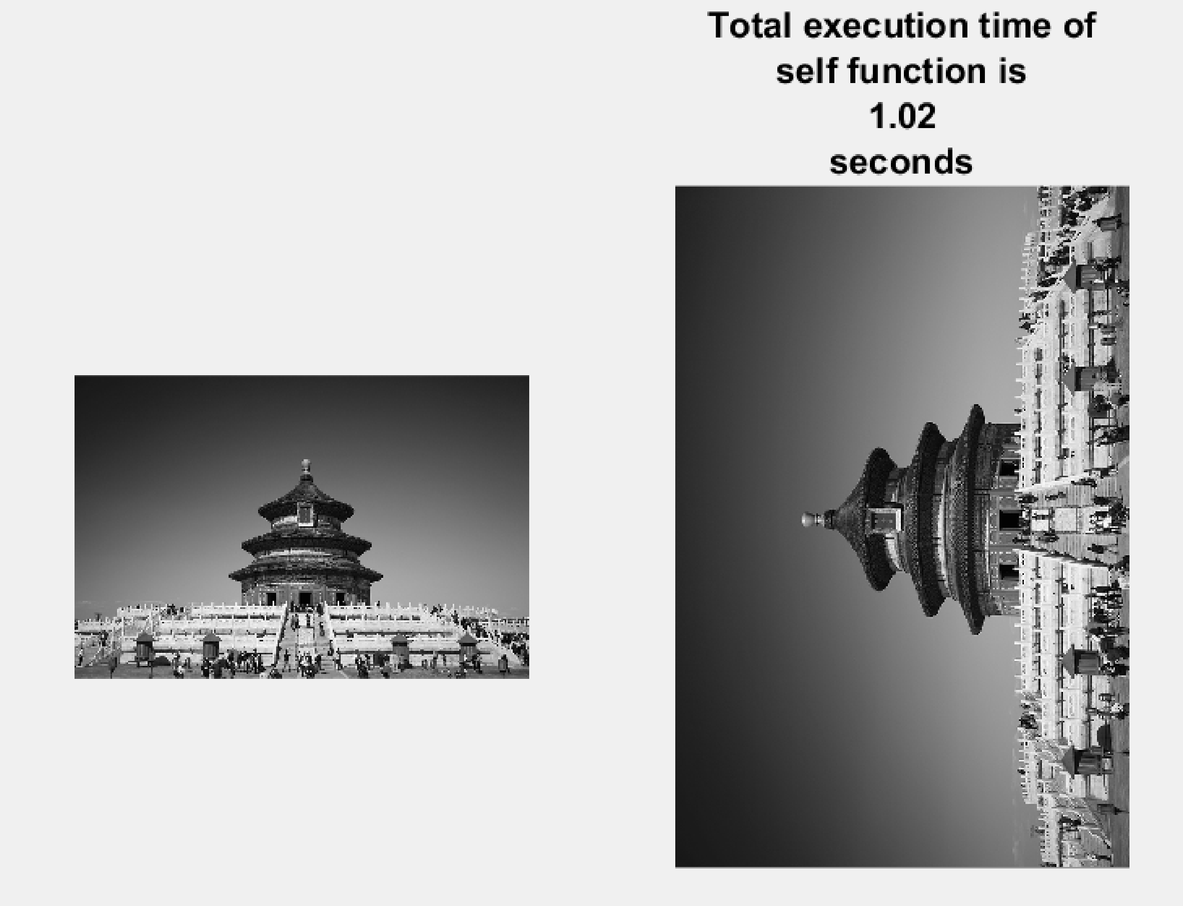
(f) Vertical shear with sv=0.2.

(g) Horizontal shear with sv=0.3.

(h) Translate with tx=50,ty=45.

(i) Rotation followed by translation.

(j) Built-in rotation and execution time.



(a) Self-coded rotation and execution time.

2 Histogram Equalization

2.1 Code For Equalization

```
imc = imread('im2.png');% Read the image
img = rgb2gray(imc); % Convert to grayscale

H=size(img,1); % Read the height of the image
W=size(img,2); % Read the width of the image
Hist_arr=zeros(1,256); % Array for holding the (original) histogram
Hist_eq_arr=zeros(1,256); % Array for holding the (equalized)
CDF_array=zeros(1,length(Hist_arr)); % array to hold

hist_eq_img=uint8(zeros(H,W)); % A 2D array for keeping

for i=1:H
for j=1:W
Hist_arr(1,img(i,j)+1)=Hist_arr (1,img(i,j)+1)+1 ;
end
end

Hist_arr_pdf=Hist_arr/(H*W); % PDF
dummy1=0; % A dummy variable to hold the summation results
for k=1:length(Hist_arr) % Generating the CDF from PDF
dummy1=dummy1+Hist_arr_pdf(k);
CDF_array(k)= dummy1;
end
for l=1:H % Histogram equalization
for m=1:W
hist_eq_img(l,m)= round(CDF_array(img(l,m)+1)*...
(length(Hist_arr_pdf)-1)); % scale to 255 and round to
Hist_eq_arr(1, hist_eq_img(l,m)+1)=...
Hist_eq_arr(1,hist_eq_img(l,m)+1)+1 ; % Its histogram
end
end
```

```
subplot(3,2,1)
bar(Hist_arr);
title("original histogram")

subplot(3,2,2)
bar(Hist_eq_arr);
title("equalized histogram")

subplot(3,2,3)
bar(CDF_array);
title("CDF histogram")
subplot(3,2,5)
imshow(img);
title("original image");

subplot(3,2,6)
imshow(hist_eq_img,[])
title("equalized image");
print('eq','-dpng');
```

2.2 Outcome Set II

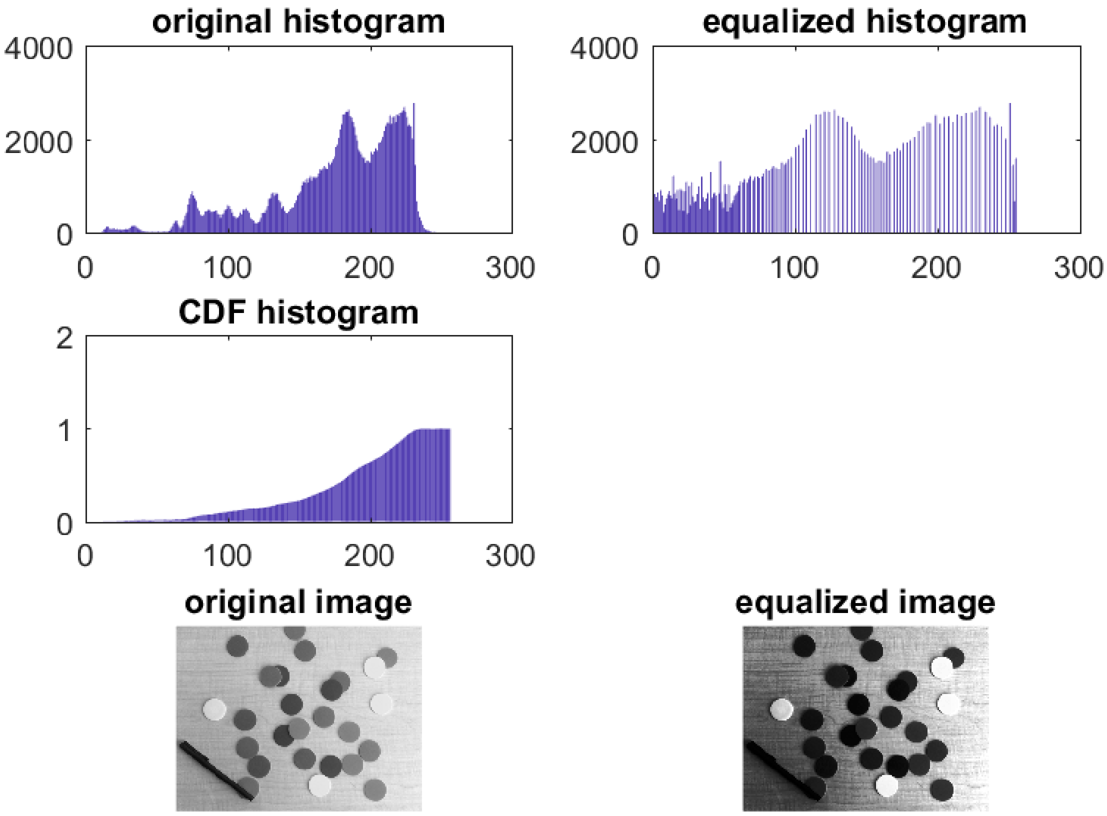


Figure 3: Outcome of Histogram Equalization.

3 Discussion

3.1 Why black dots after rotation?

When we do a forward rotation transformation, there are many pixel not being mapped,which leads to those point unassigned with the intensity values remain 0.

3.2 Why the equalization not quite even?

This is because discretization. For the continuous transformation $s = T(r)$, the outcome is uniform. While for the digital image, we round the value so that there are many specific intensity value was lost.

3.3 Why different execution speed?

From the outcome, we can see that the built-in rotation only need tenth total time of the self-coded rotation. I did not dig into the primitive implementation of built-in function such as maketform and imwarp. Definitely, it is because we implement the rotation by brute-force attack without any optimization.