# Task6: Image Filter & Edge Detection

**Announcement**: February 27, 2020
**Due date**: March 08, 2020 (Sunday) before 11:59 pm

**Name**: Jasmine Zou
**Degree**: BA
**ID**: 20602200

The main goal of this task is to understand the fundamental of the linear filter, edge detection, and Hough transform. You must study the tutorials at edge detection and linear filter. Please answer all sub-questions in each problem. You should write your own code to solve these questions *if needed*.

📣📣 **PLEASE INCLUDE YOUR RESULTING IMAGES OR GRAPHS IN THE REPORT**

## Problem 1: 2D Convolution (10 points)

The general expression of a 2D convolution is

$$G[i,j] = \sum_{u=-k}^{k} \sum_{v=-k}^{k} H[u,v]F[i-u, j-v]$$

where G is the filtered image, F is the original image, and H is the kernel.

You first take a color picture or download any color image on the web and resize the image to have 200 pixel in the shortest side. For example, if your image 5000 x 4000 pixel, it becomes 250x200 pixel. Then, please do the convolution of the resized image and the kernel H:

$$h = \begin{bmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{bmatrix}$$

(a) Please compute G using `conv2`. In the shape option, you set `same`.

```
%% Question 1(a)
%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%% ----- Q1a----- %%%%%%%%%%%%%%%%%%%%%%%%%%%%
clear; close all; clc; format shortG;

img = 'original.jpg';
plant = imread(img);
info = imfinfo(img);
w=info.Width; h=info.Height;

%resize the image
plant_new = imresize(plant,200/w);
```

```
% plant_new = rgb2gray(plant_new)
plant_new = im2double(plant_new);

imshow(plant);title('original')
imshow(plant_new); title('resized: w = 200')

%construct the kernal
k1 = zeros(3,3);
k1(2,2) = 4;
k1(1,2)=-1;k1(2,1)=-1;k1(2,3)=-1;k1(3,2) =-1;

%filter each RGB layer with conv2 seperately
plant_new_f1 = zeros(size(plant_new));
for i = 1:3
  plant_new_f1(:,:,i) = conv2(plant_new(:,:,i),k1,'same');
end

%results
figure(11)
subplot(121); imshow(plant_new);title('original')
% subplot(122); imshow(conv2(plant_new,k1,'same'))

subplot(122); imshow(plant_new_f1);title('conv2')
%%
```


(b) Please compute G using your own code (use of a for-loop structure).

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%% ----- Q1b----- %%%%%%%%%%%%%%%%%%%%%%%%%%%%%

clear; close all; clc; format shortG;

img = 'original.jpg';
plant = imread(img);
info = imfinfo(img);
w=info.Width; h=info.Height;

%resize the image
plant_new = imresize(plant,200/w);
A = rgb2gray(plant_new)
plant_new = im2double(plant_new);
A = im2double(A);

imshow(plant);title('original')
imshow(plant_new); title('resized: w = 200')

%construct the kernal
k1 = zeros(3,3);
```

```
k1(2,2) = 4;
k1(1,2)=-1;k1(2,1)=-1;k1(2,3)=-1;k1(3,2) =-1;

%filter each RGB layer with conv2 seperately
plant_new_f1 = zeros(size(A));
k = rot90(k1,2);
for ii = 1:size(A,1)-size(k,1)+1
    for jj = 1:size(A,2)-size(k,2)+1
        plant_new_f1(ii,jj)= sum(sum(k.*A(ii:ii+size(k,1)-1,jj:jj+size(k,2)-1)));
    end
end

%results
figure(11)
subplot(121); imshow(plant_new),title('original')
% subplot(122); imshow(conv2(plant_new,k1,'same'))

subplot(122); imshow(plant_new_f1),title('myconv')
```


Note that answers for (a) and (b) must be the same.

# Problem 2: Different Kernels (20 points)

You are going to filter your image using the following kernels. Please conduct convolution of your image (used in Problem 1) with the kernel H1, H2, H3, and H4 respectively. Then, you need to explain the effect of filtering with the each kernel.

$$H_1 = \frac{1}{9}\begin{bmatrix}1 & 1 & 1\\ 1 & 1 & 1\\ 1 & 1 & 1\end{bmatrix}, \quad H_2 = \begin{bmatrix}0 & 1 & 0\\ 1 & -4 & 1\\ 0 & 1 & 0\end{bmatrix}, \quad H_3 = \begin{bmatrix}-0.55 & -0.55 & -0.55\\ -0.55 & 5.40 & -0.55\\ -0.55 & -0.55 & -0.55\end{bmatrix},$$

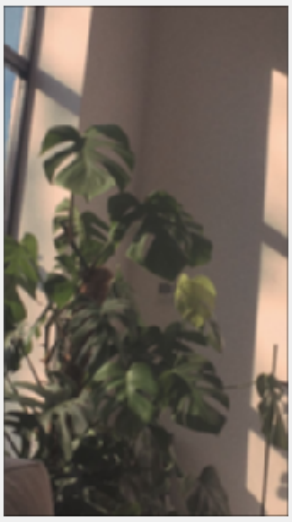
$$H_4 = \begin{bmatrix}0.0030 & 0.0133 & 0.0219 & 0.0133 & 0.0030\\ 0.0133 & 0.0596 & 0.0983 & 0.0596 & 0.0133\\ 0.0219 & 0.0983 & 0.1621 & 0.0983 & 0.0219\\ 0.0133 & 0.0596 & 0.0983 & 0.0596 & 0.0133\\ 0.0030 & 0.0133 & 0.0219 & 0.0133 & 0.0030\end{bmatrix}$$

```
%% Question 2
%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%% ----- Q2 ----- %%%%%%%%%%%%%%%%%%%%%%%%%%%%
clear; close all; clc; format shortG;
img = 'original.jpg';
plant = imread(img);
info = imfinfo(img);
w=info.Width; h=info.Height;

%resize the image
plant_new = imresize(plant,200/w);
% plant_new = rgb2gray(plant_new)
plant_new = im2double(plant_new);

imshow(plant);title('original')
imshow(plant_new); title('resized: w = 200')


h1 = ones(3,3)/9;
h2 = [0,1,0; 1,-4,1;0,1,0];
h3 = ones(3,3)*(-0.55);
h3(2,2)=5.4;
h4 = [0.003,0.0133,0.0219,0.0133,0.003;
0.0133,0.0596,0.0983,0.0596,0.0133;0.0219,0.0983,0.1621,0.0983,0.0133;0.0133,0.0596,0.
0983,0.0596,0.0133;0.003,0.0133,0.0219,0.0133,0.003];

%h1
plant_new_f1 = zeros(size(plant_new));
for i = 1:3
  plant_new_f1(:,:,i) = conv2(plant_new(:,:,i),h1,'same');
end

%h2
```

```
plant_new_f2 = zeros(size(plant_new));
for i = 1:3
  plant_new_f2(:,:,i) = conv2(plant_new(:,:,i),h2,'same');
end

%h3
plant_new_f3 = zeros(size(plant_new));
for i = 1:3
  plant_new_f3(:,:,i) = conv2(plant_new(:,:,i),h3,'same');
end

%h4
plant_new_f4 = zeros(size(plant_new));
for i = 1:3
  plant_new_f4(:,:,i) = conv2(plant_new(:,:,i),h4,'same');
end

figure(22)
subplot(221); imshow(plant_new_f1),title('h1')
subplot(222); imshow(plant_new_f2),title('h2')
subplot(223); imshow(plant_new_f3),title('h3')
subplot(224); imshow(plant_new_f4),title('h4')
```

h1 has a blurring effect.
h2 highlights the edges with white contrast.
h3 has a sharpening effect.
h4 reduces the noises and blurs the image.

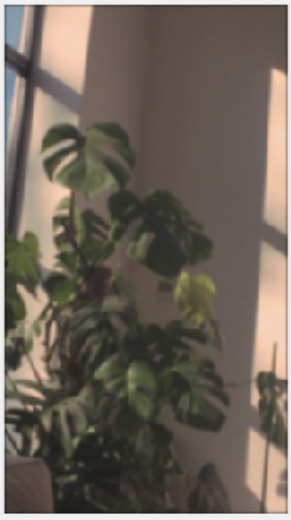
h1



h2

h3

h4



# Problem 3: Gaussian Kernel (20 points)

**See tutorials!!**

(a) Write a code to create the following Gaussian kernel h without using `fspecial`.

```
h = fspecial('gaussian', 5, 2);
```

You need to use theoretical Gaussian curve to get the value of each element in h. Your code produces the kernel close to h obtained from the `fspecial` function.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%% ----- Q3(a) ----- %%%%%%%%%%%%%%%%%%%%%%%%%%%%
clear; close all; clc; format shortG;
h = fspecial('gaussian', 5, 2)
```

```
h2= mygaussian(5,2)

% problem 3
function f= mygaussian(n,sigma)
    [x,y] = meshgrid(1:n);
    x=x-(round(n/2));
    y=y-(round(n/2));
    f = 1/(2*pi*sigma^2)*exp(-(x.^2+y.^2)/(2*sigma^2));
    f = f/sum(f(:));
end

%%
```

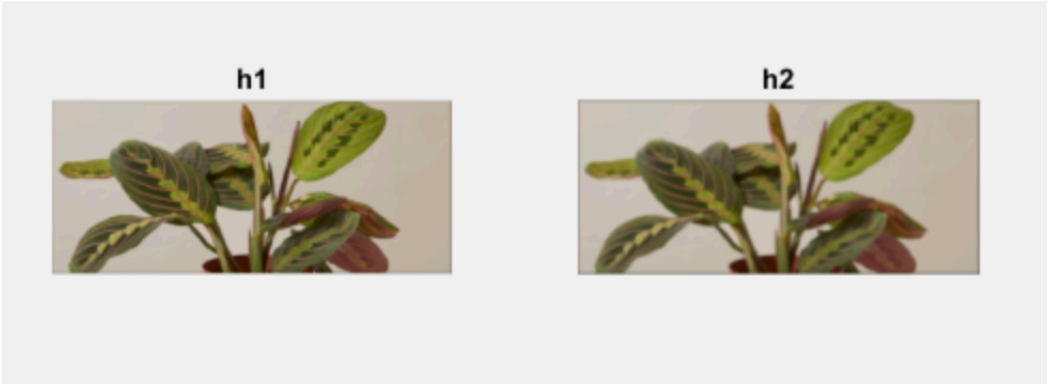
```
h = 5×5
    0.023247    0.033824    0.038328    0.033824    0.023247
    0.033824    0.049214    0.055766    0.049214    0.033824
    0.038328    0.055766    0.063191    0.055766    0.038328
    0.033824    0.049214    0.055766    0.049214    0.033824
    0.023247    0.033824    0.038328    0.033824    0.023247

h2 = 5×5
    0.023247    0.033824    0.038328    0.033824    0.023247
    0.033824    0.049214    0.055766    0.049214    0.033824
    0.038328    0.055766    0.063191    0.055766    0.038328
    0.033824    0.049214    0.055766    0.049214    0.033824
    0.023247    0.033824    0.038328    0.033824    0.023247
```

(b) What is the difference between the kernel h1 and h2 in terms of the effect on the filtered images with these kernels? You can explain it with sample results.

```
h1 = fspecial('gaussian', 20, 2);
h2 = fspecial('gaussian', 20, 3);
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%% ----- Q3(b) ----- %%%%%%%%%%%%%%%%%%%%%%%%%%%%
img = 'plant2.jpg';
plant = imread(img);
imshow(plant);title('original')
plant_new = im2double(plant);

h1 = fspecial('gaussian', 20, 2);
h2 = fspecial('gaussian', 20, 3);

plant_new_f1 = zeros(size(plant_new));
for i = 1:3
  plant_new_f1(:,:,i) = conv2(plant_new(:,:,i),h1,'same');
end

plant_new_f2 = zeros(size(plant_new));
for i = 1:3
  plant_new_f2(:,:,i) = conv2(plant_new(:,:,i),h2,'same');
end
```

```
%results
figure(11)
subplot(121); imshow(plant_new_f1),title('h1')
subplot(122); imshow(plant_new_f2),title('h2')

%evaluate edge details
BW_filt_f1 = edge(rgb2gray(plant_new_f1), 'Canny');
BW_filt_f2 = edge(rgb2gray(plant_new_f2), 'Canny');

figure(22)
subplot(221); imshow(BW_filt_f1),title('h1');
subplot(222); imshow(BW_filt_f2),title('h2');
```





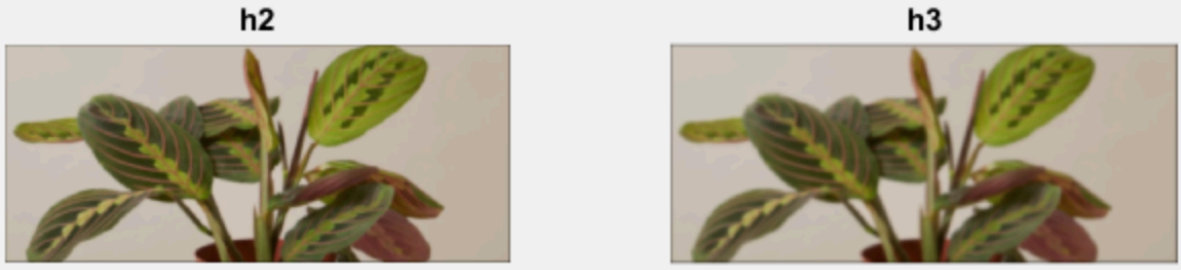
h2 blurrs and smoothens the mage more than h1. h2 has a higher standard deviation. h2 detects large scale edges rather than fine details.

(c) What is the difference between the kernel h2 and h3 in terms of filtered images with these kernels? Are they different? You can explain it with sample results.

```
h2 = fspecial('gaussian', 20, 3);
h3 = fspecial('gaussian', 50, 3);
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%% ----- Q3(c) ----- %%%%%%%%%%%%%%%%%%%%%%%%%%%%
img = 'plant2.jpg';
plant = imread(img);
info = imfinfo(img);
w=info.Width; h=info.Height;

h2 = fspecial('gaussian', 20, 3);
```

```
h3 = fspecial('gaussian', 50, 3);

plant_new_f2 = zeros(size(plant_new));
for i = 1:3
  plant_new_f2(:,:,i) = conv2(plant_new(:,:,i),h1,'same');
end

plant_new_f3 = zeros(size(plant_new));
for i = 1:3
  plant_new_f3(:,:,i) = conv2(plant_new(:,:,i),h2,'same');
end

%results
figure(11)
subplot(121); imshow(plant_new_f2),title('h2')
subplot(122); imshow(plant_new_f3),title('h3')

%evaluate edge details
BW_filt_f2 = edge(rgb2gray(plant_new_f2), 'Canny');
BW_filt_f3 = edge(rgb2gray(plant_new_f3), 'Canny');

figure(22)
subplot(221); imshow(BW_filt_f2),title('h2');
subplot(222); imshow(BW_filt_f3),title('h3');
%%
% h3 blurrs and smoothens the mage more than h2. h3 has a much higher fitler
% size and captures less details.
%%
```



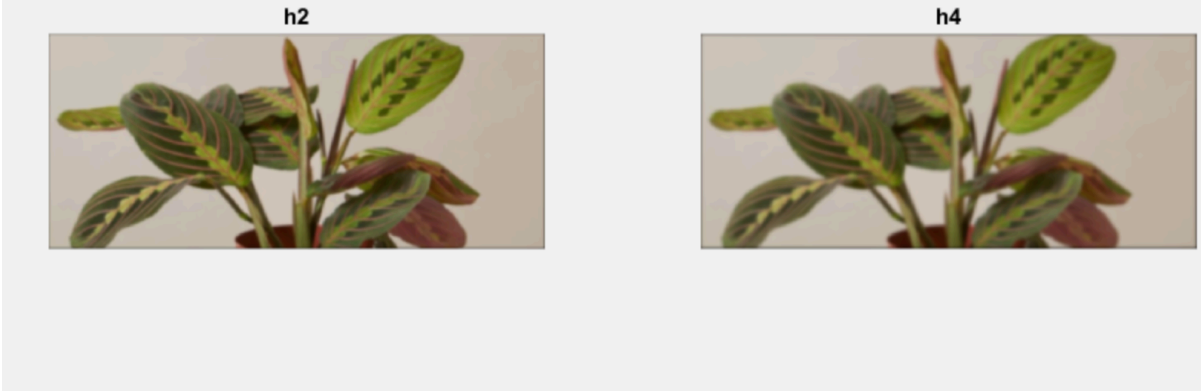

h3 blurrs and smoothens the mage more than h2. h3 has a much higher fitler size and captures less details.

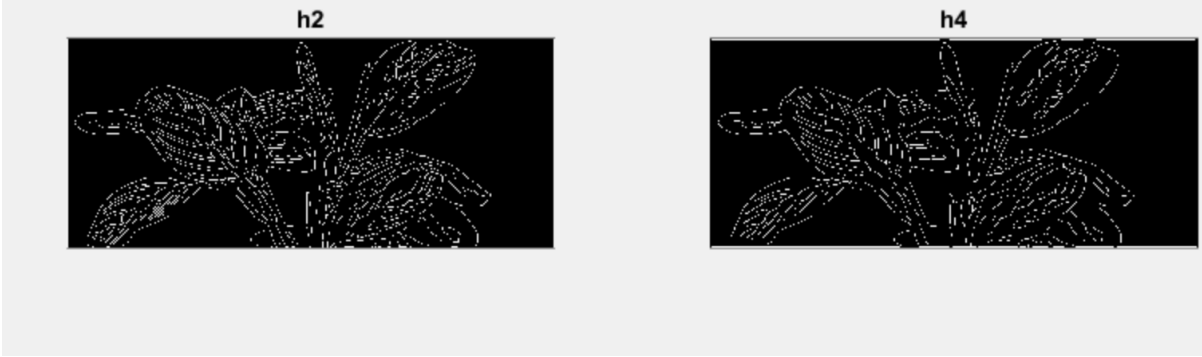
(d) What is the difference between the kernel h2 and h4 in terms of filtered images with these kernels? Are they different? You can explain it with sample results.

```
h2 = fspecial('gaussian', 20, 3);
h4 = fspecial('gaussian', 20, 20);
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%% ----- Q3(d) ----- %%%%%%%%%%%%%%%%%%%%%%%%%%%%
img = 'plant2.jpg';
plant = imread(img);
info = imfinfo(img);
w=info.Width; h=info.Height;

h2 = fspecial('gaussian', 20, 3);
h4 = fspecial('gaussian', 20, 20);

plant_new_f2 = zeros(size(plant_new));
for i = 1:3
  plant_new_f2(:,:,i) = conv2(plant_new(:,:,i),h1,'same');
end

plant_new_f4 = zeros(size(plant_new));
for i = 1:3
  plant_new_f4(:,:,i) = conv2(plant_new(:,:,i),h2,'same');
end

%results
figure(11)
subplot(121); imshow(plant_new_f2),title('h2')
subplot(122); imshow(plant_new_f4),title('h4')

%evaluate edge details
BW_filt_f2 = edge(rgb2gray(plant_new_f2), 'Canny');
BW_filt_f4 = edge(rgb2gray(plant_new_f4), 'Canny');

figure(22)
```

```
subplot(221); imshow(BW_filt_f2),title('h2');
subplot(222); imshow(BW_filt_f4),title('h4');
```



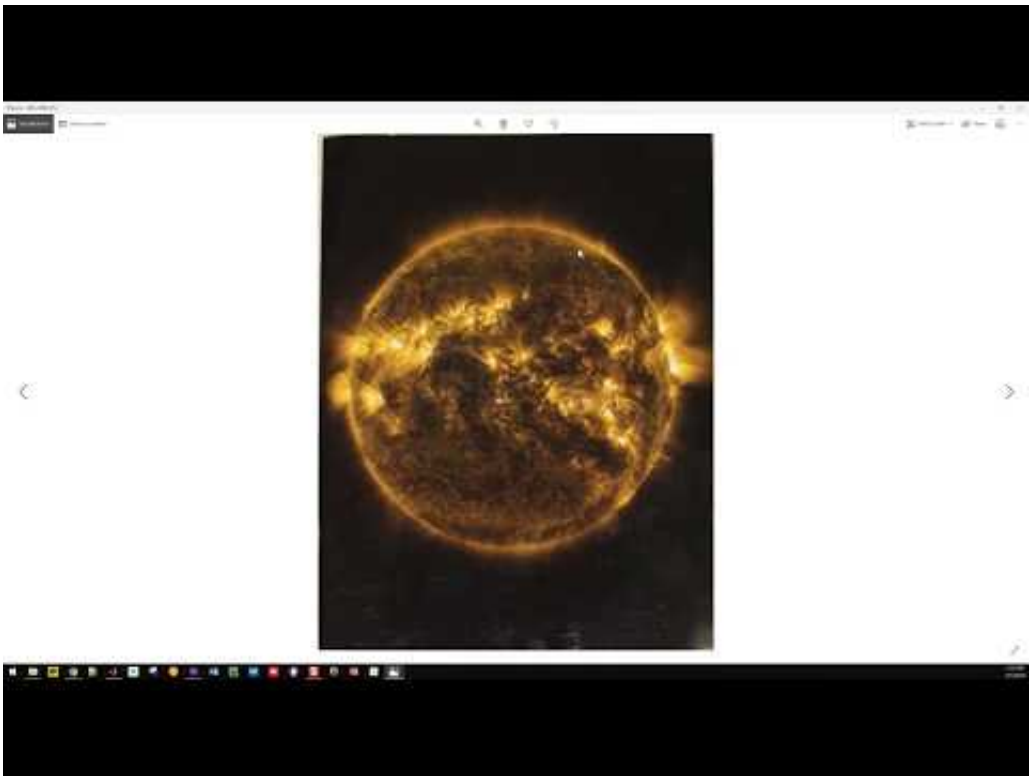

h4 blurrs and smoothens the image more than h2. h4 has a much higher standard deviation and captures larger features (lesser fine details).

Please **ignore** the boundary effect. You can do either zero-padding (`same` option in `conv2` in MATLAB) or get the valid area (`valid` option in `conv2` in MATLAB).

# Problem 4: Hough Transform (25 points)

This problem is to write your own program that extracts a booklet image from each image provided and remove its projective distortion in an automated way.

Here is a sample demo.



A sample code is provided. You need to write your own code for extracting four corners of the booklet using edge detection and hough transform. In other word, you need to write your own `FindCorner` function. In addition, you need to reuse your `ComputeH`, which was used in Task 3. You only remove perspective distortion, meaning that resulting images may be holizontally or vertically flipped.

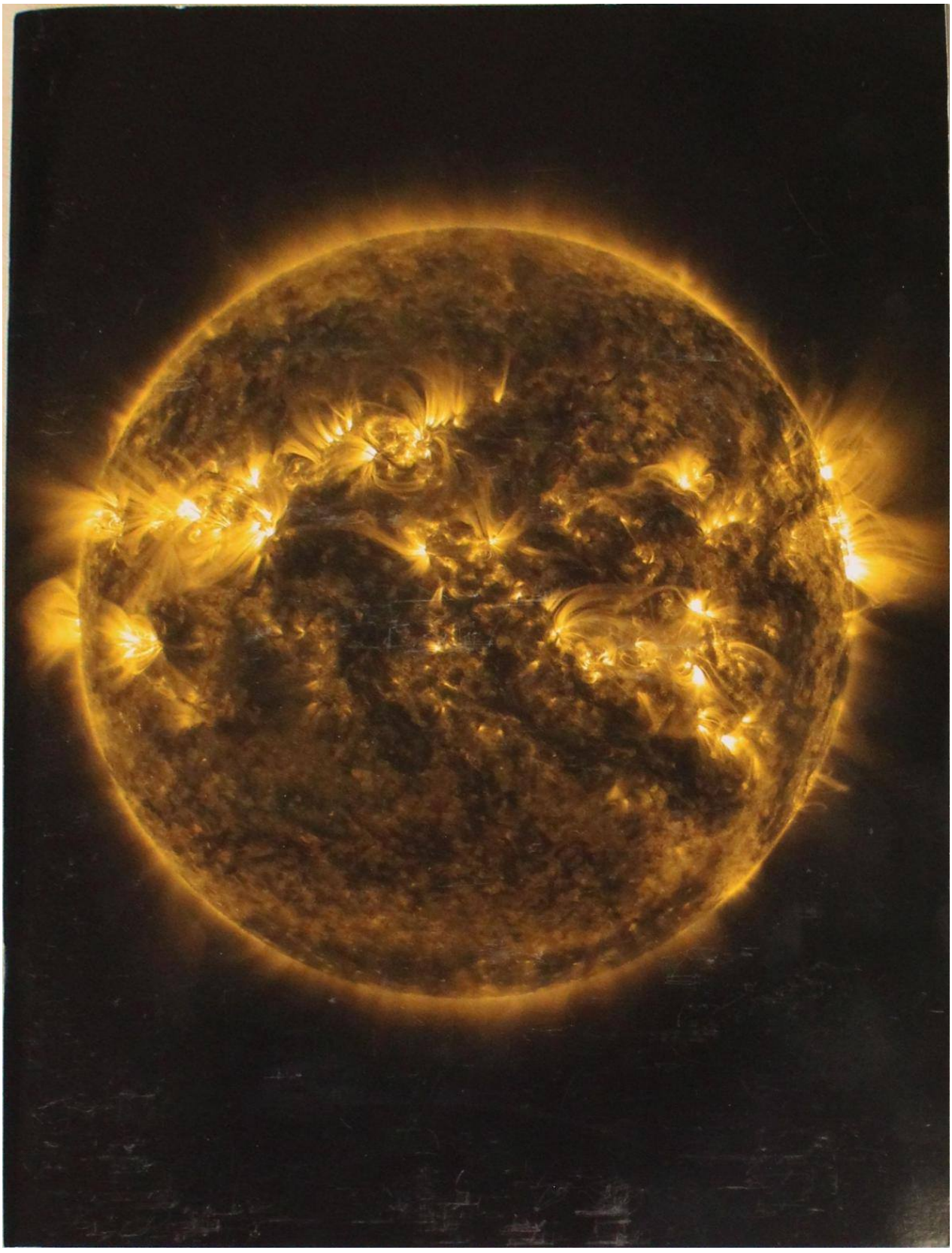
Hint: I used functions of `edge`, `imgaussfilt`, `houghpeaks`, `hough`, `cross`, `norm` in MATLAB to write the `FindCorner`. However, you can use any functions in OpenCV or MATLAB. You can use `fitgeotrans` in MATLAB and `getPerspectiveTransform` in Python.

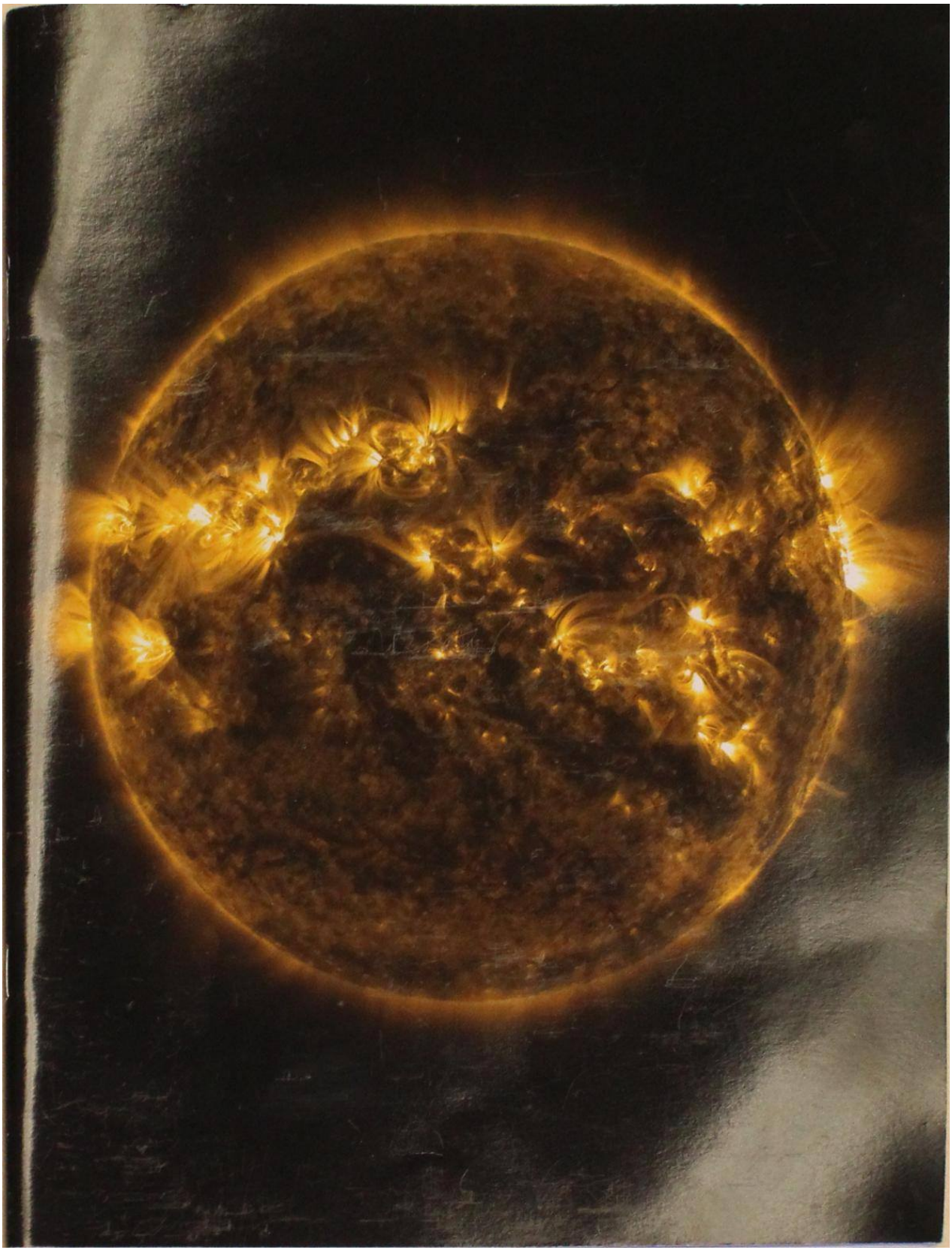
```
%% Question 4
%%%%%%%%%%%%%%%%%%%%%%%%%%%% ----- Q4 ----- %%%%%%%%%%%%%%%%%%%%%%%%%%%%
clear; close all; clc; format shortG;

%% Parameter
bookletSize = [24 31.5]; % cm
bookletImgSize = bookletSize*50; % output image size
w1 = 24*50; h1 = 31.5*50;
bookletCorner = [1,1;w1,1;1,h1;w1,h1];
dirImg = 'img'; % image folder
dirOut = 'out'; % output image folder
imgList = dir('img/*.JPG');
nImg = numel(imgList);

%% Processing
for ii=1:nImg
    img = imread(fullfile(dirImg, imgList(ii).name));
```

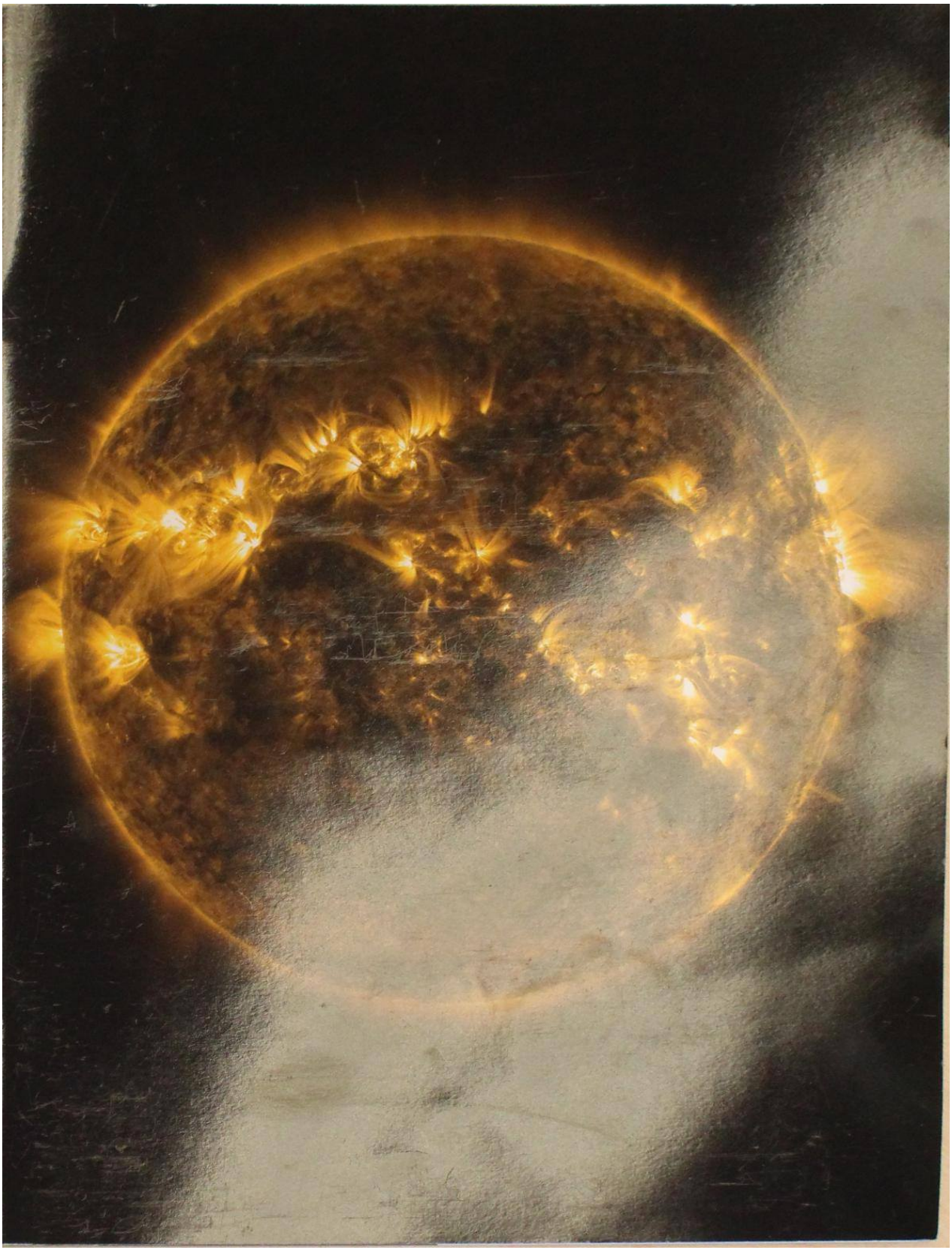
```
    corner = FindCorner(img); % find ordered four corners
    H = fitgeotrans(corner, bookletCorner,'projective'); % compute a homography
    [imgTran, RA] = imwarp(img,H);
    bookletImg = imcrop(imgTran, [-RA.XWorldLimits(1), -RA.YWorldLimits(1)
bookletImgSize]);
    imwrite(bookletImg, fullfile(dirOut,imgList(ii).name));
end

%% function
function corners = FindCorner(img)
img = rgb2gray(img);
blurred = imgaussfilt(img,3);
BW = edge(blurred, 'canny',[0.1,0.4]); % sobel, canny
% imshow(BW)

% hough transformation
[H,T,R] = hough(BW,'RhoResolution',0.5,'Theta',-90:0.5:89);
%
% imshow(H,[],'XData',T,'YData',R,...
%         'InitialMagnification','fit');
% xlabel('\theta'), ylabel('\rho');
% axis on, axis normal, hold on;

% find hough peaks
P  = houghpeaks(H,4,'Threshold',0.2*max(H(:)));
x = T(P(:,2)); y = R(P(:,1));
% plot(x,y,'s','color','green');

% Once a set of candidate peaks has been identified in the Hough transform,
% it remains to be determined if there are line segments associated with those
% peaks, as well as start and ending points.
% For each peak, the first step is to find the location of all nonzero pixels
% in the image that contributed to that peak and construct line segments based
% on those pixels. The houghlines function can do this.

% Detect lines and overlay on top of image
lines = houghlines(BW, T, R, P);
% figure, imshow(img), hold on
% for k = 1:length(lines)
% xy = [lines(k).point1; lines(k).point2];
% plot(xy(:,1), xy(:,2), 'g.-', 'LineWidth',2);
% end
%
% hold off

% get unique points
[~, idx] = unique([lines.theta].','rows','stable');
lines_u = lines(idx);
```

```
% find m and b for y = mx + b // line
% line 1
point_1 = lines_u(1).point1 ;
point_2 = lines_u(1).point2 ;
% m_1 = (point_2(2)-point_1(2))/(point_2(1)-point_1(1))
% b_1 = point_1(2)-m_1*point_1(1)

l1 = cross([point_1 1],[point_2,1]);

% line 2
point_1 = lines_u(2).point1 ;
point_2 = lines_u(2).point2 ;
% m_2 = (point_2(2)-point_1(2))/(point_2(1)-point_1(1))
% b_2 = point_1(2)-m_1*point_1(1)

l2 = cross([point_1 1],[point_2,1]);

% line 3
point_1 = lines_u(3).point1 ;
point_2 = lines_u(3).point2 ;
% m_3 = (point_2(2)-point_1(2))/(point_2(1)-point_1(1))
% b_3 = point_1(2)-m_1*point_1(1)

l3 = cross([point_1 1],[point_2,1]);

% line 4
point_1 = lines_u(4).point1 ;
point_2 = lines_u(4).point2 ;
% m_4 = (point_2(2)-point_1(2))/(point_2(1)-point_1(1))
% b_4 = point_1(2)-m_1*point_1(1)

l4 = cross([point_1 1],[point_2,1]);

% line1 = [m_1,-1,b_1]
% line2 = [m_2,-1,b_2]
% line3 = [m_3,-1,b_3]
% line4 = [m_4,-1,b_4]
%
% p1 = cross(line1,line2)
% p2 = cross(line1,line3)
% p3 = cross(line1,line4)
% p4 = cross(line2,line3)
% p5 = cross(line2,line4)
% p6 = cross(line3,line4)

p1 = cross(l1,l2);
p2 = cross(l1,l3);
p3 = cross(l1,l4);
p4 = cross(l2,l3);
```

```
p5 = cross(l2,l4);
p6 = cross(l3,l4);

c1 = [p1(1)/p1(3),p1(2)/p1(3)];
c2 = [p2(1)/p2(3),p2(2)/p2(3)];
c3 = [p3(1)/p3(3),p3(2)/p3(3)];
c4 = [p4(1)/p4(3),p4(2)/p4(3)];
c5 = [p5(1)/p5(3),p5(2)/p5(3)];
c6 = [p6(1)/p6(3),p6(2)/p6(3)];

corners =[];
if (c1(1)> 0) && (c1(2)>0)
corners=[corners;c1];
end

if (c2(1)> 0) && (c2(2)>0)
corners=[corners;c2];
end

if (c3(1)> 0) && (c3(2)>0)
corners=[corners;c3];
end

if (c4(1)> 0) && (c4(2)>0)
corners=[corners;c4];
end

if (c5(1)> 0) && (c5(2)>0)
corners=[corners;c5];
end

if (c6(1)> 0) && (c6(2)>0)
corners=[corners;c6];
end

% imshow(img); hold on
% plot(corners(:,1),corners(:,2),'ro')
%
% hold off

corners = [corners(1,:); corners(3,:); corners(2,:); corners(4,:)];
%BL, BR, TL, TR

end
```



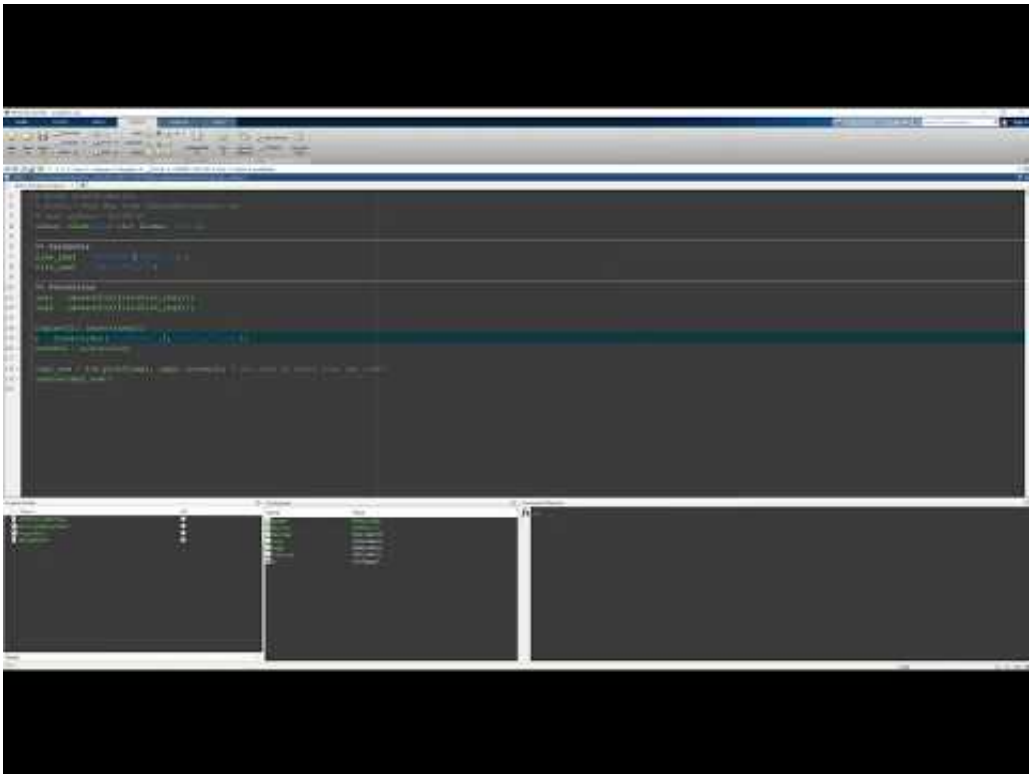




## Problem 5 (Challenging): Hough Transform & Image Overlay (25 points)

This problem is to write your own program that extracts a booklet image from an image provided and overlay the extracted image into the white board. Note the height of the booklet image should be placed at the width of the board.

Here is a sample demo.



Your goal is to write your own `fun_prob5` function. A sample code is provided in **problem5**. You can reuse any code that you wrote for solving Problem 4 in Task 6 or Problem 5 or 6 in Task 5.

```
%% Question 5
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% ----- Q5 ----- %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
clear; close all; clc; format shortG;

%% Parameter
file_img1 = '20200227_104101.jpg';
file_img2 = 'IMG_0080.JPG';

%% Processing
img1 = imread(fullfile(file_img1)); % whiteboard
img2 = imread(fullfile(file_img2)); % desired pic

figure(1); imshow(img1);
p = impoly();
corner1 = p.getPosition;

img2_new = fun_prob5(img1, img2, corner1); % you need to write your own code!
imshow(img2_new);imwrite(img2_new, 'q5.jpg')

function imgFinal = fun_prob5(imgBoard, img2, wboard)
% cut out the desired image to be pasted in the whiteboard
corners = FindCorner(img2)
bookletSize = [24 31.5]; % cm
bookletImgSize = bookletSize*50; % output image size
w1 = 24*50; h1 = 31.5*50;
```

```
bookletCorner = [1,1;w1,1;1,h1;w1,h1];
H = fitgeotrans(corners, bookletCorner,'projective'); % compute a homography
[imgTran, RA] = imwarp(img2,H);
% this is the desired image to be pasted
bookletImg = imcrop(imgTran, [-RA.XWorldLimits(1), -RA.YWorldLimits(1)
bookletImgSize]);
%imwrite(bookletImg, 'result_image.jpg');

% pasting to the whiteboard
ugh = [w1,1;...
       w1,h1;...
       1,1;...
       1,h1];

corner(1:2,:)= wboard(1:2,:);
corner(3,:)= wboard(4,:);
corner(4,:)= wboard(3,:);

H_ugh = calcH(ugh, wboard);

[imgPicTran, RB] = imwarp(bookletImg, projective2d(H_ugh));
BWPic = roipoly(imgPicTran, corner(:,1)-RB.XWorldLimits(1), corner(:,2)-
RB.YWorldLimits(1));

BWBoard = ~roipoly(imgBoard, corner(:,1), corner(:,2));
RA = imref2d(size(BWBoard));

imgBoardMask = bsxfun(@times, imgBoard, cast(BWBoard, 'like', imgBoard));
imgPicTranMask = bsxfun(@times, imgPicTran, cast(BWPic, 'like', imgPicTran));

imgFinal(:,:,1) = imfuse(imgBoardMask(:,:,1),RA, imgPicTranMask(:,:,1),RB,'diff');
imgFinal(:,:,2) = imfuse(imgBoardMask(:,:,2),RA, imgPicTranMask(:,:,2),RB,'diff');
imgFinal(:,:,3) = imfuse(imgBoardMask(:,:,3),RA, imgPicTranMask(:,:,3),RB,'diff');

%imshow(imgFinal); %imwrite(imgFinal, 'result_1.jpg');

end

function corners = FindCorner(img)
img = rgb2gray(img);
blurred = imgaussfilt(img,3);
BW = edge(blurred, 'canny',[0.1,0.4]); % sobel, canny
% imshow(BW)

% hough transformation
[H,T,R] = hough(BW,'RhoResolution',0.5,'Theta',-90:0.5:89);
%
% imshow(H,[],'XData',T,'YData',R,...
%         'InitialMagnification','fit');
```

```
% xlabel('\theta'), ylabel('\rho');
% axis on, axis normal, hold on;

% find hough peaks
P  = houghpeaks(H,4,'Threshold',0.2*max(H(:)));
x = T(P(:,2)); y = R(P(:,1));
% plot(x,y,'s','color','green');

% Once a set of candidate peaks has been identified in the Hough transform,
% it remains to be determined if there are line segments associated with those
% peaks, as well as start and ending points.
% For each peak, the first step is to find the location of all nonzero pixels
% in the image that contributed to that peak and construct line segments based
% on those pixels. The houghlines function can do this.

% Detect lines and overlay on top of image
lines = houghlines(BW, T, R, P);
% figure, imshow(img), hold on
% for k = 1:length(lines)
% xy = [lines(k).point1; lines(k).point2];
% plot(xy(:,1), xy(:,2), 'g.-', 'LineWidth',2);
% end
%
% hold off

% get unique points
[~, idx] = unique([lines.theta].','rows','stable');
lines_u = lines(idx);

% find m and b for y = mx + b // line
% line 1
point_1 = lines_u(1).point1 ;
point_2 = lines_u(1).point2 ;
% m_1 = (point_2(2)-point_1(2))/(point_2(1)-point_1(1))
% b_1 = point_1(2)-m_1*point_1(1)

l1 = cross([point_1 1],[point_2,1]);

% line 2
point_1 = lines_u(2).point1 ;
point_2 = lines_u(2).point2 ;
% m_2 = (point_2(2)-point_1(2))/(point_2(1)-point_1(1))
% b_2 = point_1(2)-m_1*point_1(1)

l2 = cross([point_1 1],[point_2,1]);

% line 3
point_1 = lines_u(3).point1 ;
point_2 = lines_u(3).point2 ;
```

```
% m_3 = (point_2(2)-point_1(2))/(point_2(1)-point_1(1))
% b_3 = point_1(2)-m_1*point_1(1)

l3 = cross([point_1 1],[point_2,1]);

% line 4
point_1 = lines_u(4).point1 ;
point_2 = lines_u(4).point2 ;
% m_4 = (point_2(2)-point_1(2))/(point_2(1)-point_1(1))
% b_4 = point_1(2)-m_1*point_1(1)

l4 = cross([point_1 1],[point_2,1]);

% line1 = [m_1,-1,b_1]
% line2 = [m_2,-1,b_2]
% line3 = [m_3,-1,b_3]
% line4 = [m_4,-1,b_4]
%
% p1 = cross(line1,line2)
% p2 = cross(line1,line3)
% p3 = cross(line1,line4)
% p4 = cross(line2,line3)
% p5 = cross(line2,line4)
% p6 = cross(line3,line4)

p1 = cross(l1,l2);
p2 = cross(l1,l3);
p3 = cross(l1,l4);
p4 = cross(l2,l3);
p5 = cross(l2,l4);
p6 = cross(l3,l4);

c1 = [p1(1)/p1(3),p1(2)/p1(3)];
c2 = [p2(1)/p2(3),p2(2)/p2(3)];
c3 = [p3(1)/p3(3),p3(2)/p3(3)];
c4 = [p4(1)/p4(3),p4(2)/p4(3)];
c5 = [p5(1)/p5(3),p5(2)/p5(3)];
c6 = [p6(1)/p6(3),p6(2)/p6(3)];

corners =[];
if (c1(1)> 0) && (c1(2)>0)
corners=[corners;c1];
end

if (c2(1)> 0) && (c2(2)>0)
corners=[corners;c2];
end

if (c3(1)> 0) && (c3(2)>0)
```

```
corners=[corners;c3];
end

if (c4(1)> 0) && (c4(2)>0)
corners=[corners;c4];
end

if (c5(1)> 0) && (c5(2)>0)
corners=[corners;c5];
end

if (c6(1)> 0) && (c6(2)>0)
corners=[corners;c6];
end

% imshow(img); hold on
% plot(corners(:,1),corners(:,2),'ro')
%
% hold off

corners = [corners(1,:); corners(3,:); corners(2,:); corners(4,:)];
%BL, BR, TL, TR


end

function H = calcH(pin,pout)

mat = zeros(size(pin,1)*2,9);

x1x1p = pout(:,1).* pin(:,1);
x1y1p = pout(:,1).* pin(:,2);
yax1p = pout(:,2).* pin(:,1);
y1y1p = pout(:,2).* pin(:,2);

mat(1:2:end,3) = 1;
mat(2:2:end,6) = 1;

mat(1:2:end,1:2) = pin;
mat(2:2:end,4:5) = pin;
mat(1:2:end,7) = -x1x1p;
mat(1:2:end,8) = -x1y1p;
mat(2:2:end,7) = -yax1p;
mat(2:2:end,8) = -y1y1p;
mat(1:2:end,9) = -pout(:,1);
mat(2:2:end,9) = -pout(:,2);
[svd_u, svd_s, svd_v] = svd(mat);
h = svd_v(:,end) ./ svd_v(end,end);
H = reshape(h',3,3);
```

```
end
```