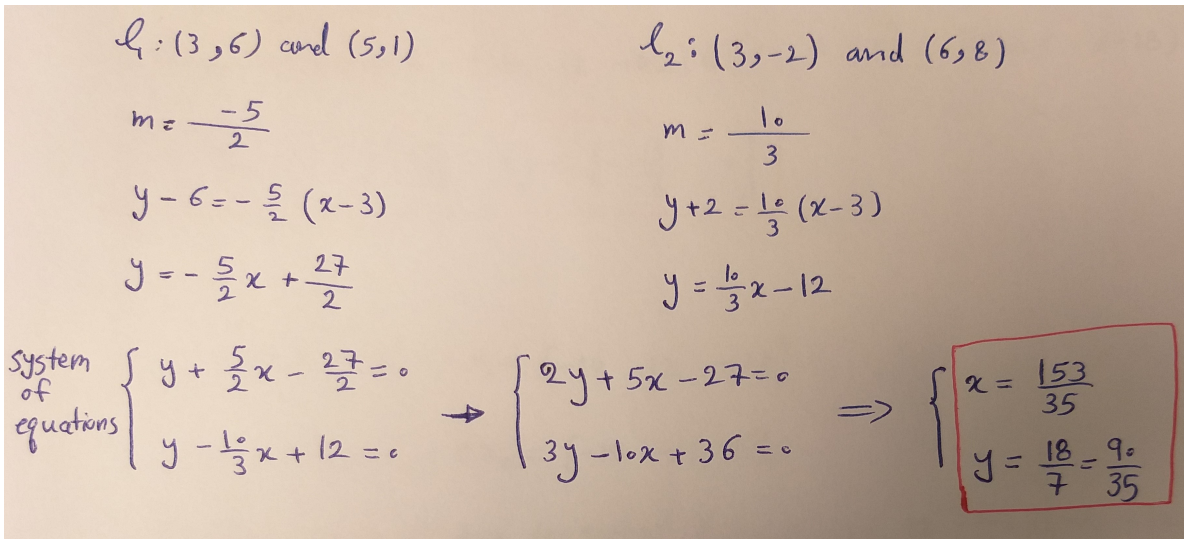
Task5: Homography

Name: Saeed Hatefi Ardakani
Degree: PH
ID: 20793159

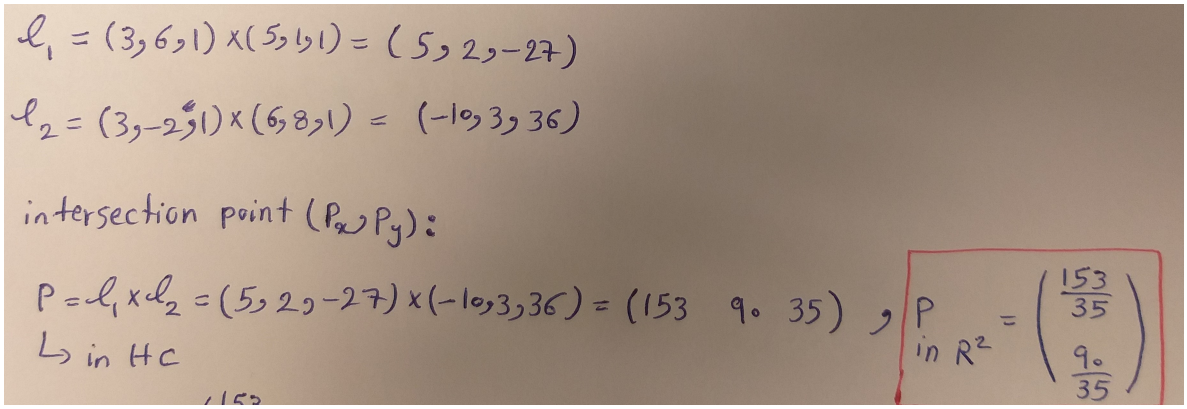
Problem 1: Homogeneous Coordinate (Lines and Points) (10 points)

The intersection of two lines l1 and l2, with l1 passing through the points (3,6) and (5,1), and l2 passing through the points (3,-2) and (6,8).

(a) Compute the intersection without using homogeneous coordinates



(b) Compute the intersection using homogeneous coordinates

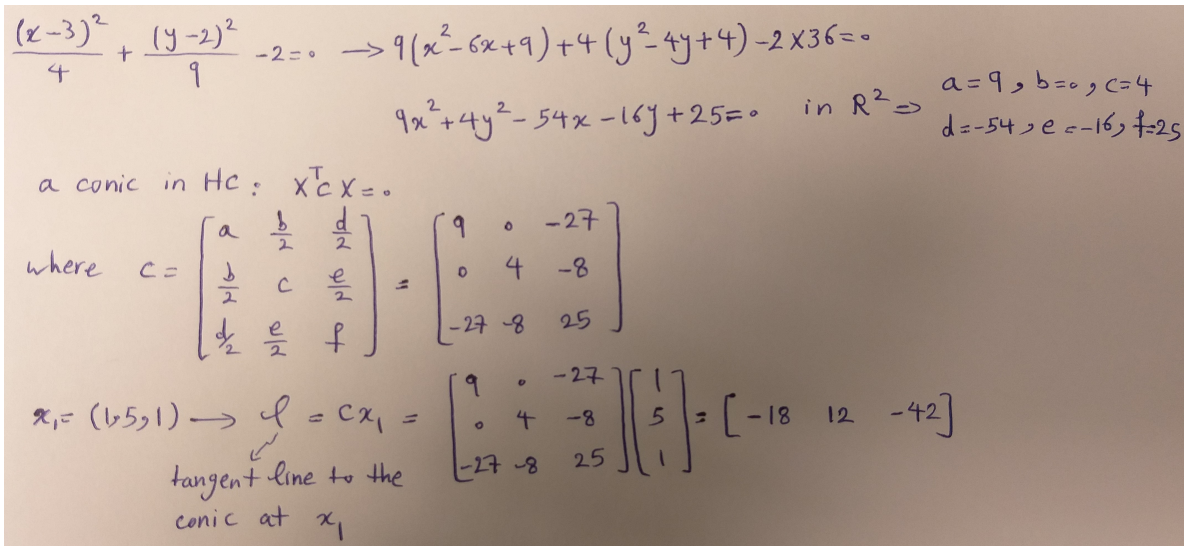


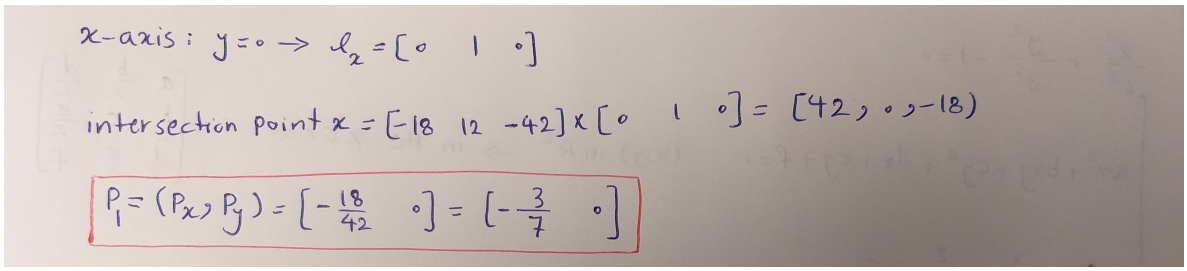
Problem 2: Homogeneous Coordinate (Conic) (10 points)

Given a conic

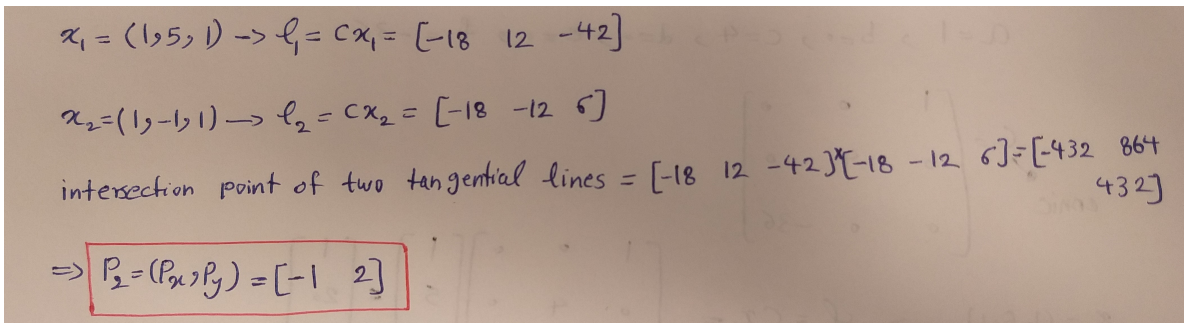
$$\frac{(x-3)^2}{2^2} + \frac{(y-2)^2}{3^2} = 2$$

(a) Compute the intersection of a tangential line to this conic at (1, 5) with the x-axis





(b) Compute the coordinates of the intersection of the tangential lines to this conic at points (1,5) and (1, -1)



Problem 3: Homogeneous Coordinate (Lines) (10 points)

$$
\begin{aligned}
l_1 &: y_1 = 1.25x_1 + 3 \\
l_2 &: y_2 = 0.4x_2 - 5 \\
l_3 &: y_3 = -2.25x_3 + 5 \\
l_4 &: y_4 = -x_4 - 8
\end{aligned}
$$

The four lines (l1, l2, l3, and l4) are intersected at six points. Among the six points, please find four points that form a quadrangle with your code, not manually.

The result of my code is:

In homogeneous coordinate:

```
p1_quad_hg =

   -2.0000
  -13.0000
   -3.5000


p2_quad_hg =

   11.0000
    7.0000
   -2.2500


p3_quad_hg =

  -10.0000
    9.2500
   -2.6500


p4_quad_hg =

    3.0000
    8.2000
   -1.4000
```

In cartesian coordinate:

```
p1_quad_car =

    0.5714    3.7143


p2_quad_car =

   -4.8889   -3.1111


p3_quad_car =

    3.7736   -3.4906


p4_quad_car =

   -2.1429   -5.8571
```

According to the first part of my code, we find all the combinations with 4 members (intersected points) in such a way that each member (point) is exactly on two lines. So the result would be these combinations:

1) 2, 3, 4, 5

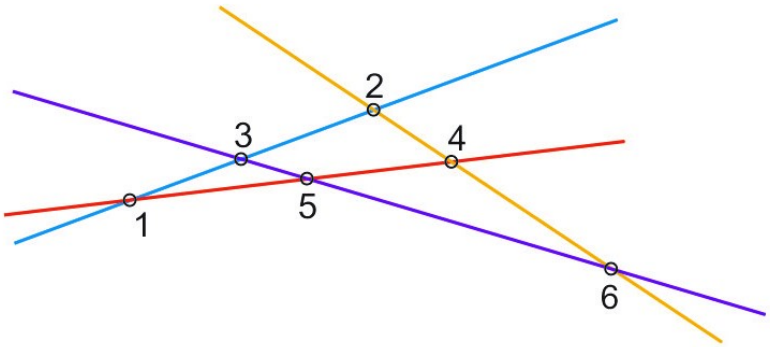
2) 1, 2, 5, 6

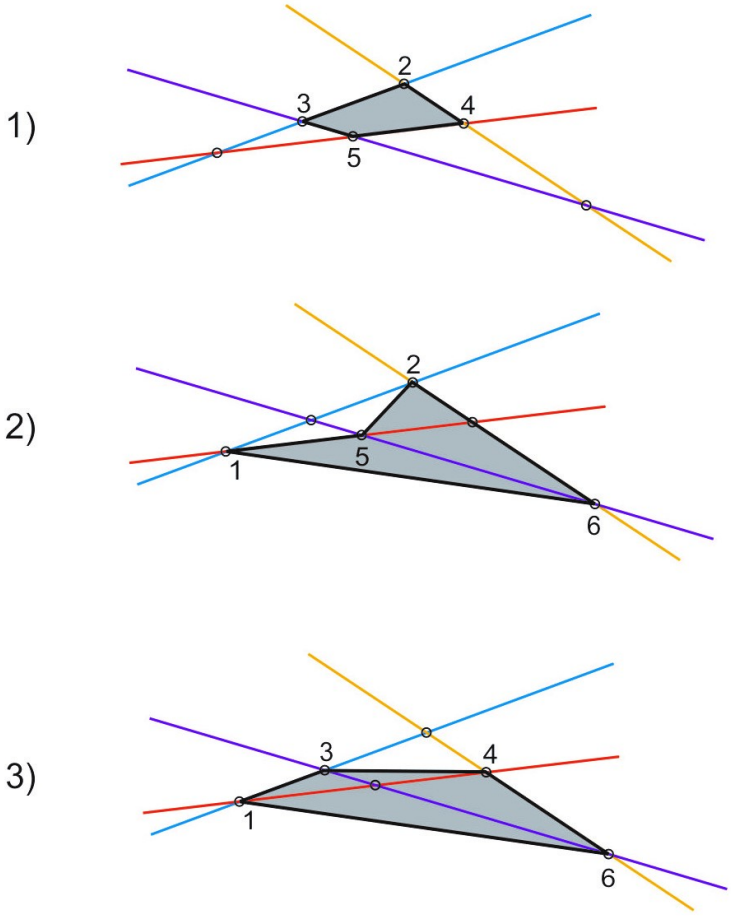
3) 1, 3, 4, 6

First part of my code:

```
clc
clear all

% main four lines
l1 = [1.25 ; -1 ; 3];
l2 = [0.4 ; -1 ; -5];
l3 = [-2.25 ; -1 ; 5];
l4 = [-1 ; -1 ; -8];

% intersection of these four lines: 6 points
p1 = cross(l1,l2);
p2 = cross(l1,l3);
p3 = cross(l1,l4);
p4 = cross(l2,l3);
p5 = cross(l2,l4);
p6 = cross(l3,l4);
points = [p1 p2 p3 p4 p5 p6];


% finding all the combinations with 4 members (intersected points) in such a way
that
% each member (point) is exactly on two lines
combination = combnk(1:6,4);
combinations = [];
for ii = 1 : size(combination,1)
    intersect = [l1 l2 l3 l4]'*points(:,combination(ii,:));

    % check that all the points of the combination are exactly on two lines
    if length(find(abs(intersect(1,:))<1e-6))==2 &&
length(find(abs(intersect(2,:))<1e-6))==2 && length(find(abs(intersect(3,:))<1e-
6))==2 && length(find(abs(intersect(4,:))<1e-6))==2
        combinations = [combinations ; combination(ii,:)];
    end
end
```

I have shown these combinations in the below figures:





For finding four points that form a quadrangle, we need to filter some combinations. So, We use two steps for filtering. First, we do step 1 to filter a combination (combination (2)). Then, we apply step 2 to filter other combination (combination (3)). Finally, we have only one combination as the final result.

Here, I explain briefly step1 and step2:

In step 1, we filter combination (2). We have four points (1, 2, 5, 6) (see above figure). First, we select 3 points of the set (which forms a triangle) and check that 4th point is inside of the triangle or not. We repeat this process for every 3 points selected between 1, 2, 5, 6. If there is a point which fall into the triangle, the combination will be filtered and removed as a final combination. For example, point 5 will fall into the triangle (1, 2, 6). So, the combination (2) is not our final four points.

In step 2, we filter another combination (3). We have four points (1, 3, 4, 6) (see above figure). First, we form a polygon with this four points. Also, we consider other points (points 2, 5) as extra points. Then, we check that whether at least one of the extra points falls into the polygon or not. If there is an extra point which falls into the polygon, the combination will be filtered and removed as a final combination. For example, point 5 will fall into the polygon (1, 3, 4, 6). So, the combination (3) is not our final points.

It should be mentioned that combination (1) will not satisfy steps 1 and 2. So, our final result is combination (1).

Main code:

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%---------- Problem 3 ---------
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

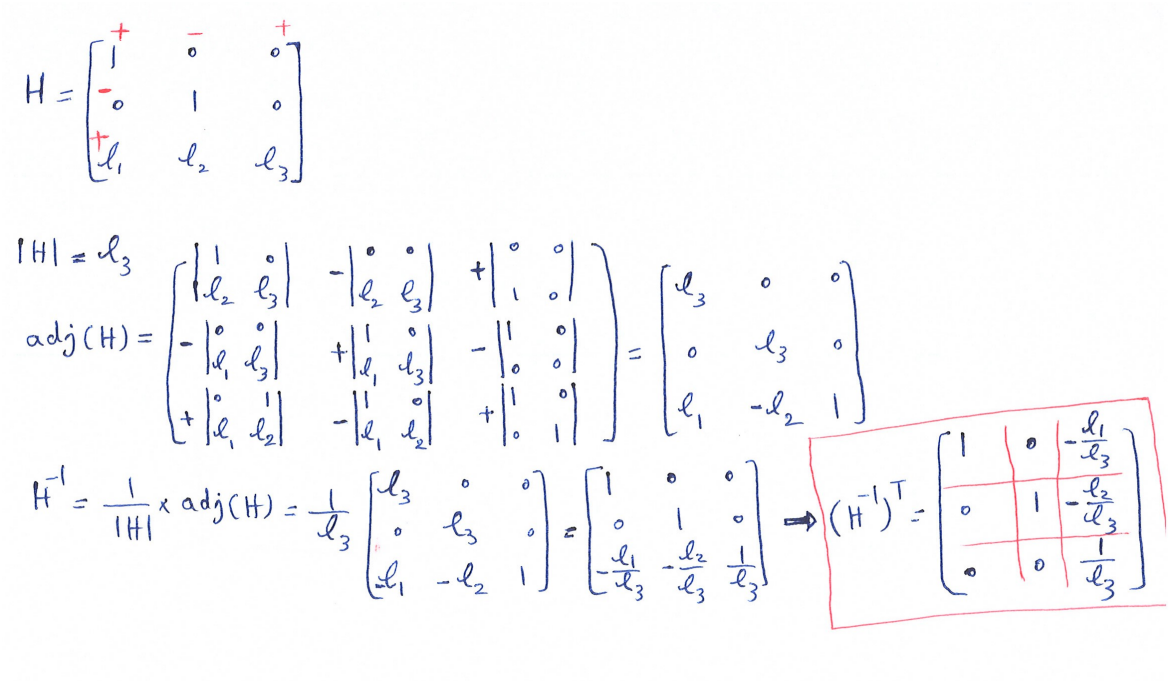
```
clc
clear all

% main four lines
l1 = [1.25 ; -1 ; 3];
l2 = [0.4 ; -1 ; -5];
l3 = [-2.25 ; -1 ; 5];
l4 = [-1 ; -1 ; -8];

% intersection of these four lines: 6 points
p1 = cross(l1,l2);
p2 = cross(l1,l3);
p3 = cross(l1,l4);
p4 = cross(l2,l3);
p5 = cross(l2,l4);
p6 = cross(l3,l4);
points = [p1 p2 p3 p4 p5 p6];


% finding all the combinations with 4 members (intersected points) in such a way
that each member (point) is exactly on two lines
combination = combnk(1:6,4);
combinations = [];
for ii = 1 : size(combination,1)
    intersect = [l1 l2 l3 l4]'*points(:,combination(ii,:));

    % check that all the points of the combination are exactly on two lines
    if length(find(abs(intersect(1,:))<1e-6))==2 &&
length(find(abs(intersect(2,:))<1e-6))==2 && length(find(abs(intersect(3,:))<1e-
6))==2 && length(find(abs(intersect(4,:))<1e-6))==2
        combinations = [combinations ; combination(ii,:)];
    end
end


% Here, we need to filter some combinations to find four points that form a
% quadrangle. We used two steps for filtering. First, we do step 1 to filter
% some combinations. Then, we apply step 2 to filter other combinations.
% Finally, we have only one combination as the final result.
quad_points = [];
for ii = 1 : size(combinations,1)
    index = 0;

    % step 1:
    comb1 = combnk(combinations(ii,:),3);
    for jj = 1 : size(comb1,1)
        p_in = setdiff(combinations(ii,:),comb1(jj,:));
        tri = points(:,[comb1(jj,:) comb1(jj,1)]);

        % to check that the point is inside the polygon or not
        in =
inpolygon(points(1,p_in)/points(3,p_in),points(2,p_in)/points(3,p_in),tri(1,:)./
tri(3,:),tri(2,:)./tri(3,:));
        if in ~= 0
            index = 1;
        end
    end
```

```
    % step 2:
    comb2 = [combinations(ii,1) combinations(ii,3) combinations(ii,2)
combinations(ii,4);
        combinations(ii,1) combinations(ii,3) combinations(ii,4)
combinations(ii,2);
        combinations(ii,1) combinations(ii,2) combinations(ii,3)
combinations(ii,4)];
    extra_points = setdiff([1 2 3 4 5 6],combinations(ii,:));
    for jj = 1 : size(comb2,1)
        tri = points(:,[comb2(jj,:) comb2(jj,1)]);
        in1 =
inpolygon(points(1,extra_points(1))/points(3,extra_points(1)),points(2,extra_poi
nts(1))/points(3,extra_points(1)),tri(1,:)./tri(3,:),tri(2,:)./tri(3,:));
        in2 =
inpolygon(points(1,extra_points(2))/points(3,extra_points(2)),points(2,extra_poi
nts(2))/points(3,extra_points(2)),tri(1,:)./tri(3,:),tri(2,:)./tri(3,:));
        if in1 ~= 0 || in2 ~= 0
            index = 1;
        end
    end

    % final combination
    if index == 0
        % four points that form a quadrangle in HG coordinate
        quad_points = [quad_points ; points(:,combinations(ii,:))];
         % selected combination as a final result
        selected_combination = combinations(ii,:);
    end

end

% final result in homogenuous coordinate
p1_quad_hg = points(:,selected_combination(1))
p2_quad_hg = points(:,selected_combination(2))
p3_quad_hg = points(:,selected_combination(3))
p4_quad_hg = points(:,selected_combination(4))

% final result in cartesian coordinate
p1_quad_car =
[points(1,selected_combination(1))/points(3,selected_combination(1))
points(2,selected_combination(1))/points(3,selected_combination(1))]
p2_quad_car =
[points(1,selected_combination(2))/points(3,selected_combination(2))
points(2,selected_combination(2))/points(3,selected_combination(2))]
p3_quad_car =
[points(1,selected_combination(3))/points(3,selected_combination(3))
points(2,selected_combination(3))/points(3,selected_combination(3))]
p4_quad_car =
[points(1,selected_combination(4))/points(3,selected_combination(4))
points(2,selected_combination(4))/points(3,selected_combination(4))]

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

Problem 4: Linear Algebra (20 points)

(a) Find a transpose of inverse H

$$H = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ l_1 & l_2 & l_3 \end{bmatrix}$$



(b) What is the definition of a row and null space? Please find rank and nullity of A.

$$A = \begin{bmatrix} 1 & 3 & 2 & 1 & 4 \\ 3 & 3 & 5 & 10 & 1 \\ -1 & 5 & 1 & 2 & -4 \end{bmatrix}$$

row(A):
The row space of matrix ***A*** is the subspace of R^n which is spanned by the row vectors of ***A***. So, for understanding the definition of row, we need to define "subspace" and "span" concepts.

Subspace: A nonempty set of vectors in R^n is named a subspace of R^n if we can calculate each of vectors from the scalar multiplication and addition of this set of vectors.

Span: The subspace W of R^n whose vectors satisfy the combination equation ($X = t_1V_1 + t_2V_2 + ... + t_sV_s$) is named the span of $V_1, V_2, ..., V_s$. So, the span is denoted by

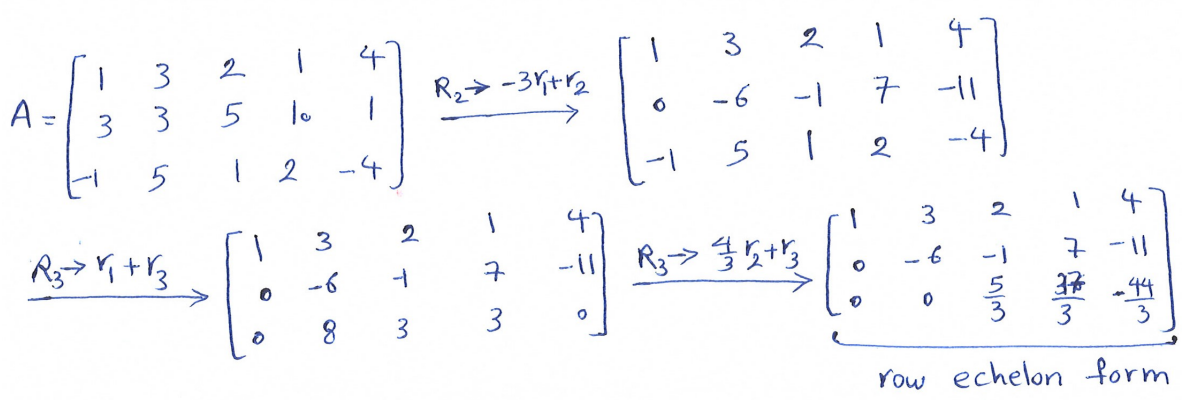
$$W = span\{V_1, V_2, ..., V_s\}$$

null(A): The null space of any matrix ***A*** consists of all the vectors ***X*** such that ***AX*** = **0** and ***A*** is not zero. Therefore, null(***A***) is a subspace of R^n.

Please find rank and nullity of *A*.

The rank of a matrix ***A*** is defined as the maximum number of linearly independent row vectors in the matrix. The maximum number of linearly independent vectors in a matrix is equal to the number of non-zero rows in its row echelon matrix. Therefore, to find the rank of a matrix, we simply transform the matrix to its row echelon form and count the number of non-zero rows.

Here, we use Gaussian elimination method to find row echelon matrix:



The final matrix (in row echelon form) has three non-zero rows and thus the rank of matrix ***A*** is 3:

$$Rank(A) = 3$$

If ***A*** is a mxn matrix, the nullity of ***A*** is equal to:

$$nullity(A) = n - Rank(A) = 5 - 3 = 2$$

(c) Find a non-singular (non-trivial) vector of x that satisfy Ax = 0. Does x exist? What is the null space of A and the nullity of A? Please explain your answer.

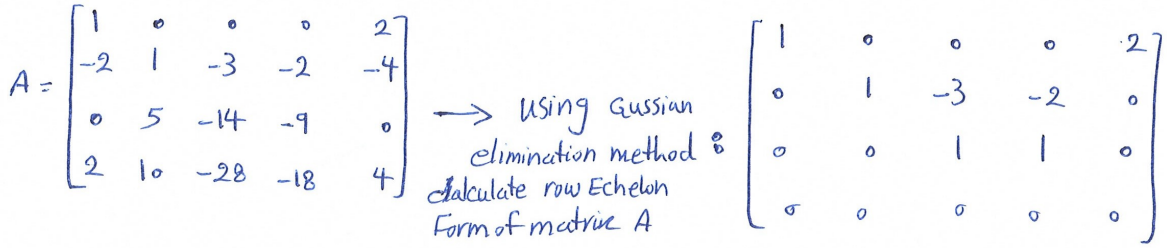
$$A = \begin{bmatrix} 1 & 0 & 0 & 0 & 2 \\ -2 & 1 & -3 & -2 & -4 \\ 0 & 5 & -14 & -9 & 0 \\ 2 & 10 & -28 & -18 & 4 \end{bmatrix}$$

(1) The dimension of the null space of ***A*** is called the nullity of ***A***: $nullity(A)$

(2) Also, the null space of ***A*** is the solution space of ***AX*** = **0**

According to the definitions (1) and (2), the solution for ***AX*** = **0** exists if $nullity(A) > 0$.

So, we need to check $nullity(A)$ for determining that the solution of ***AX*** = **0** exists or not:



⇒ rank(A) = 3 and nullity(A) = 5 − 3 = 2 > 0 ⇒ the solution exists

nullity(A) = 2

Finding vector of X:

According to row Echelon form of matrix A, we can write:

$$\begin{cases} x_1 = -2r \\ x_2 = 3s - 2s = s \\ x_3 = s \\ x_4 = -s \\ x_5 = r \end{cases} \Rightarrow \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{bmatrix} = r\underbrace{\begin{bmatrix} -2 \\ 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}}_{v_1} + s\underbrace{\begin{bmatrix} 0 \\ 1 \\ 1 \\ -1 \\ 0 \end{bmatrix}}_{v_2} \Rightarrow \boxed{X = rv_1 + sv_2}$$

Vector of X would be all combinations of v_1 and v_2.

So, null(A) is the subspace spanned by $\{v_1, v_2\}$:

$$\text{null}(A) = \text{Span}\left\{\underbrace{\begin{bmatrix} -2 \\ 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}}_{v_1}, \underbrace{\begin{bmatrix} 0 \\ 1 \\ 1 \\ -1 \\ 0 \end{bmatrix}}_{v_2}\right\}$$

Also, we can calculate null(A) from matlab function and using $A = \begin{bmatrix} 1 & 0 & 0 & 0 & 2 \\ -2 & 1 & -3 & -2 & -4 \\ 0 & 5 & -14 & -9 & 0 \\ 2 & 10 & -28 & -18 & 4 \end{bmatrix}$

So, the result is:

$$\text{null}(A) = \text{Span}\left\{\underbrace{\begin{bmatrix} 0.4583 \\ 0.4958 \\ 0.4958 \\ -0.4958 \\ -0.2291 \end{bmatrix}}_{w_1}, \underbrace{\begin{bmatrix} -0.7681 \\ 0.2958 \\ 0.2958 \\ -0.2958 \\ 0.3841 \end{bmatrix}}_{w_2}\right\} \Rightarrow \boxed{X = t_1 w_1 + t_2 w_2}$$

vector of X would be all combinations of w_1 and w_2.

note: we can prove that the result from matlab function is the same with the result of first part. In other words, we can calculate v_1 and v_2 from combinations of w_1 and w_2. For example:

$$\boxed{v_1 = w_1 \times \frac{0.3841}{0.2291 \times -1.1271} + (-1.1271) \times w_2}$$

So, the answers are the same.

(d) Find a non-singular (non-trivial) vector of x that satisfy Ax = 0. Does x exist? What is the null space of A and the nullity of A? Please explain your answer.

$$A = \begin{bmatrix} 1 & 0 & 0 & 0 & 2 \\ -2 & 1 & -3 & -2 & -4 \end{bmatrix}$$

$$A = \begin{bmatrix} 1 & 0 & 0 & 0 & 2 \\ -2 & 1 & -3 & -2 & -4 \end{bmatrix} \rightarrow \text{Row Echelon form of matrix A} : \begin{bmatrix} 1 & 0 & 0 & 0 & 2 \\ 0 & 1 & -3 & -2 & 0 \end{bmatrix}$$

$\Rightarrow$ rank(A) = 2 and nullity(A) = 5−2 = 3 > 0 $\Rightarrow$ the solution exists

nullity(A) = 3

Finding vector X:

According to row echelon form of matrix A, we can write:

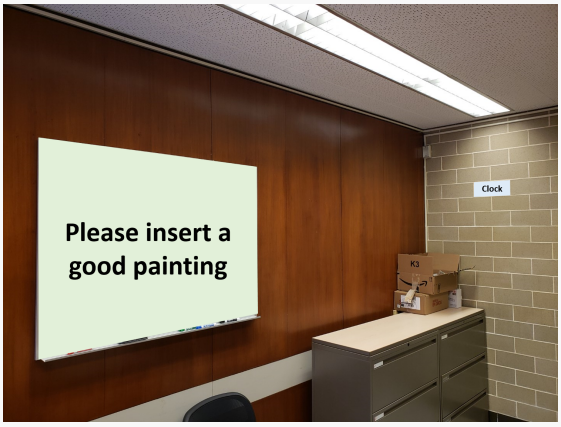
$$\begin{cases} x_1 = -2r \\ x_2 = 3p + 2s \\ x_3 = p \\ x_4 = s \\ x_5 = r \end{cases} \Rightarrow \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{bmatrix} = r \underbrace{\begin{bmatrix} -2 \\ 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}}_{v_1} + s \underbrace{\begin{bmatrix} 0 \\ 2 \\ 0 \\ 1 \\ 0 \end{bmatrix}}_{v_2} + p \underbrace{\begin{bmatrix} 0 \\ 3 \\ 1 \\ 0 \\ 0 \end{bmatrix}}_{v_3} \Rightarrow X = rv_1 + sv_2 + pv_3$$

Vector X would be all combinations of v_1, v_2 and v_3.

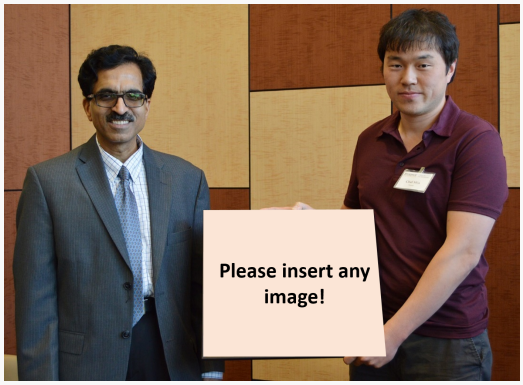
So, null(A) is the subspace spanned by $\{v_1, v_2, v_3\}$:

$$\text{null}(A) = \text{Span}\left\{ \underbrace{\begin{bmatrix} -2 \\ 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}}_{v_1}, \underbrace{\begin{bmatrix} 0 \\ 2 \\ 0 \\ 1 \\ 0 \end{bmatrix}}_{v_2}, \underbrace{\begin{bmatrix} 0 \\ 3 \\ 1 \\ 0 \\ 0 \end{bmatrix}}_{v_3} \right\}$$

Problem 5: Image Overlay (20 points)



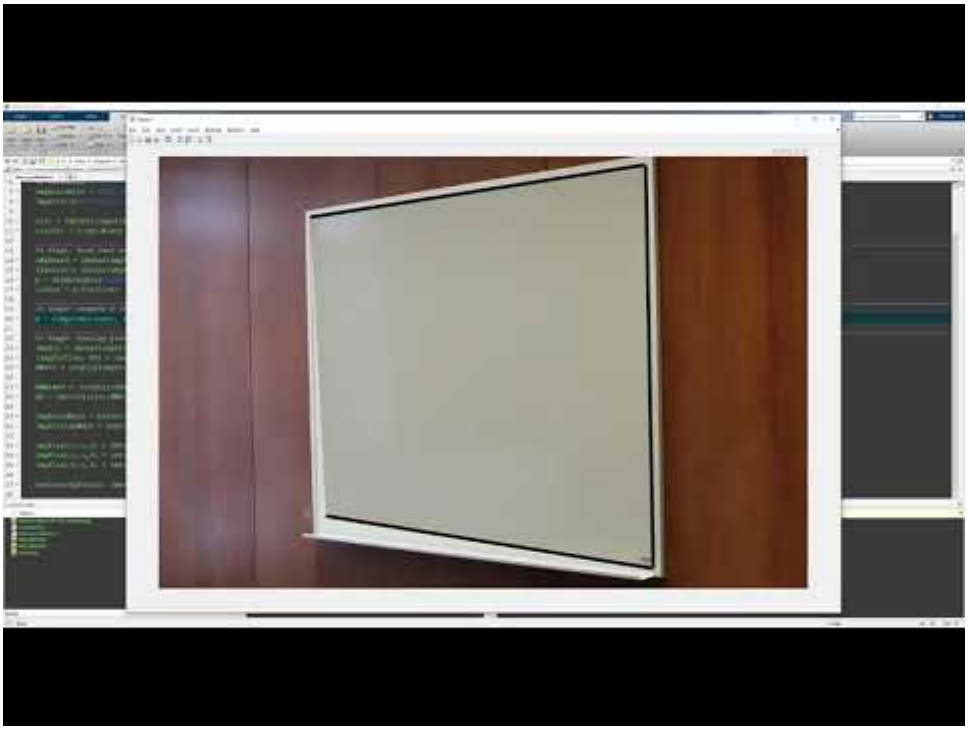
(a) original image: prob5_1.jpg



(b) original image: prob5_2.jpg

You are going to hang a good painting or image. Please find an image putting on the white board and award frame, and a digital clock (see *clock_wall.png*) is placed on the designated block (*marked as 'clock'*). Your images and the clock image are projected to the marked regions in (a) and (b). Write your own code to project your image. You need to find homography.

Here is a sample demo for one image (`demo_problem5.m` in problem5).



You can make your tool using the code provided or from the scratch. Note that you should not use `fitgeotrans` in MATLAB and `findhomography`, `getPerspectiveTransform` in Python. You need to write your own code to find homography (perspective transformation), meaning that you need to write your own `ComputeH`.

You need to manually provide the coordinates of four corners of each region on the image where your images overaly. Please do not make an interactive code for picking the corner like the one in the demo.

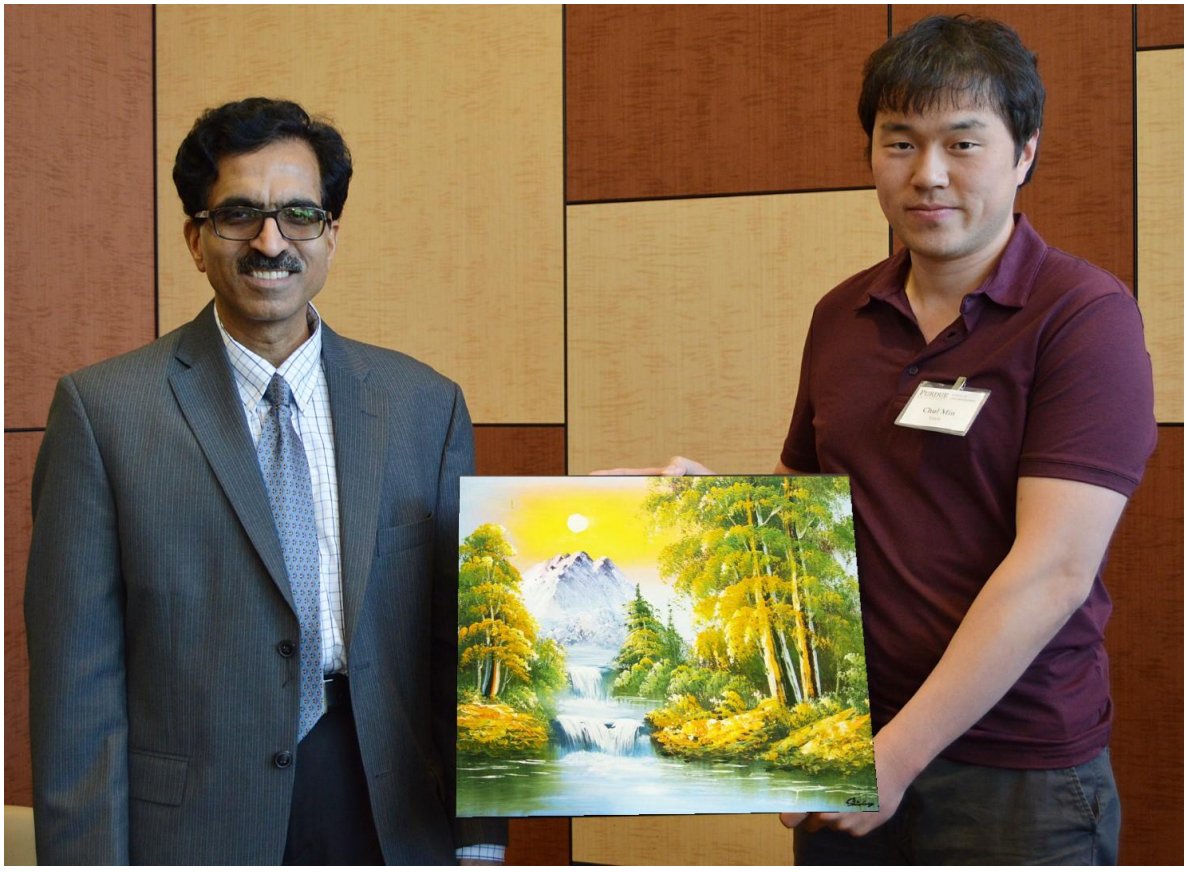
(a) putting an image on the white board and a digital clock on the designated block (marked as 'clock'):


```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%---------- Problem 5a ----------
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
clear;
close all;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Overlay image on the whtile board
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Parameter
imgBoardFile = 'prob5_1.JPG';
imgPicFile = 'flower_saeed.JPG';

info = imfinfo(imgPicFile);
sizePic = [info.Width info.Height];

%% Step1: Pick four corners of your white board in (a)
imgBoard = imread(imgBoardFile);

figure(1);
imshow(imgBoard);
p=ginput(4); % Please click on the 4 corners in the following order:
             % upper left, uper right, lower right, lower left (clockwise)
corner = p;

%% Step2: Compute H
H = ComputeH(sizePic, corner);

%% Step3: Overlay your picture
imgPic = imread(imgPicFile);
[imgPicTran, RB] = imwarp(imgPic, projective2d(H));
```

```
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, 'prob5a.jpg');


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Overlay image on the clock_wall
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Parameter
imgBoardFile = 'prob5a.JPG';
imgPicFile = 'clock_wall.png';

info = imfinfo(imgPicFile);
sizePic = [info.Width info.Height];
%% Step1: Pick four corners of your white board in (a)
imgBoard = imread(imgBoardFile);

figure(1);
imshow(imgBoard);
p_clock=ginput(4); % Please click on the 4 corners in the following order:
              % upper left, uper right, lower right, lower left (clockwise)
corner = p_clock;

%% Step2: Compute H
H = ComputeH(sizePic, corner);

%% Step3: Overlay clock_wall picture
imgPic = imread(imgPicFile);
[imgPicTran, RB] = imwarp(imgPic, projective2d(H));
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, 'prob5a.jpg');


%%
function H = ComputeH(sizePic, corner)
    points1 = [1,1;sizePic(1),1;sizePic(1),sizePic(2);1,sizePic(2)];
    points2 = corner;
    x_xp = points1(:,1).*points2(:,1);
    x_yp = points1(:,1).*points2(:,2);
    y_xp = points1(:,2).*points2(:,1);
    y_yp = points1(:,2).*points2(:,2);

    A = zeros(size(points1,1)*2,9);
    A(1:2:end,3) = 1;
    A(2:2:end,6) = 1;
    A(1:2:end,1:2) = points1;
    A(2:2:end,4:5) = points1;
    A(1:2:end,7) = -x_xp;
    A(1:2:end,8) = -y_xp;
    A(2:2:end,7) = -x_yp;
    A(2:2:end,8) = -y_yp;
    A(1:2:end,9) = -points2(:,1);
    A(2:2:end,9) = -points2(:,2);

    T = null(A);
    T = T/T(end);
    H = (reshape(T,3,3));
end
```

(b) putting an image on the award frame:



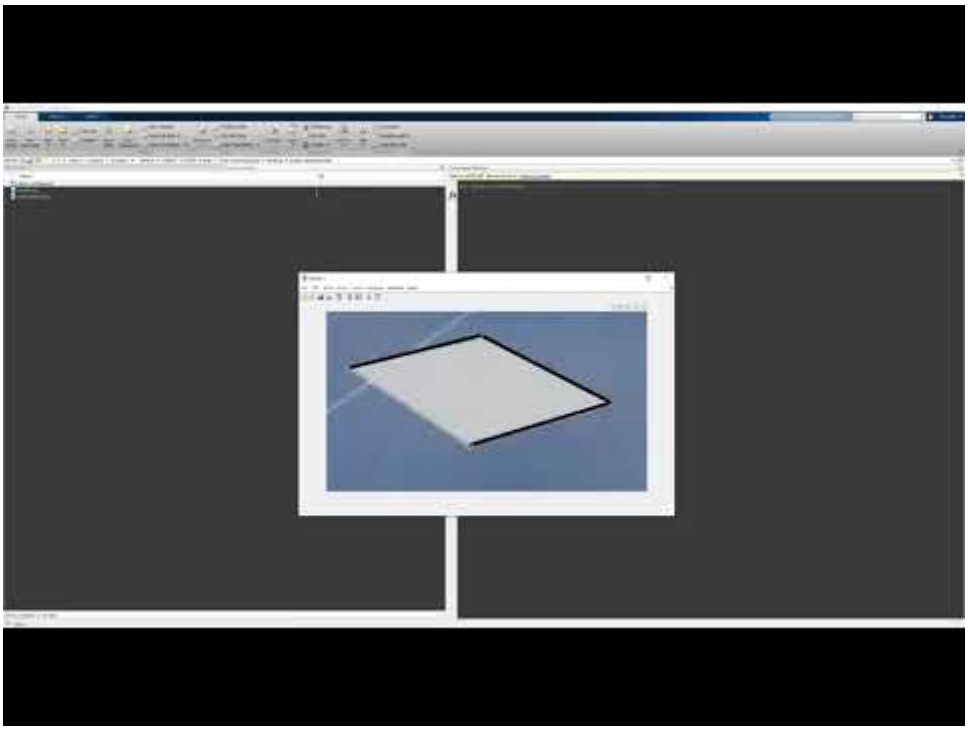
```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%---------- Problem 5b ----------
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
clear all;
close all;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Overlay image on the whtile board
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Parameter
imgBoardFile = 'prob5_2.JPG';
imgPicFile = 'painting.JPG';

info = imfinfo(imgPicFile);
sizePic = [info.Width info.Height];

%% Step1: Pick four corners of your white board in (a)
imgBoard = imread(imgBoardFile);

figure(1);
imshow(imgBoard);
p=ginput(4); % Please click on the 4 corners in the following order:
             % upper left, uper right, lower right, lower left (clockwise)
corner = p;

%% Step2: Compute H
H = ComputeH(sizePic, corner);

%% Step3: Overlay your picture
imgPic = imread(imgPicFile);
[imgPicTran, RB] = imwarp(imgPic, projective2d(H));
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, 'prob5b.jpg');


%%
function H = ComputeH(sizePic, corner)
    points1 = [1,1;sizePic(1),1;sizePic(1),sizePic(2);1,sizePic(2)];
    points2 = corner;
    x_xp = points1(:,1).*points2(:,1);
    x_yp = points1(:,1).*points2(:,2);
    y_xp = points1(:,2).*points2(:,1);
```

```
    y_yp = points1(:,2).*points2(:,2);

    A = zeros(size(points1,1)*2,9);
    A(1:2:end,3) = 1;
    A(2:2:end,6) = 1;
    A(1:2:end,1:2) = points1;
    A(2:2:end,4:5) = points1;
    A(1:2:end,7) = -x_xp;
    A(1:2:end,8) = -y_xp;
    A(2:2:end,7) = -x_yp;
    A(2:2:end,8) = -y_yp;
    A(1:2:end,9) = -points2(:,1);
    A(2:2:end,9) = -points2(:,2);

    T = null(A);
    T = T/T(end);
    H = (reshape(T,3,3));
end
```

Problem 6: Build your 3D Planar Measurement Software (30 points)

You are going to make a 3D planar measurement tool using homography.

Here is a sample demo.



A sample code is not provided in this problem because you will implement a "very" similar code used in Problem 6. `drawpolyline` is useful for selecting the line for measurement.

You need to demonstrate the functionality of your tool using your own image as well as physical measurements. Any size of a calibration paper can be used. Please provide at least two measurements on the plane that the calibration paper is placed. Note that you should not use `fitgeotrans` in MATLAB and `getPerspectiveTransformin` Python. You need to write your own code to find homography (perspective transformation).

The width and length of the table measured physically are equal to 73.4 cm and 117.6 cm, respectively. The result calculated by the code is close to the the physical values:

```
d2 =

   73.5680


d3 =

  117.9813
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%---------- Problem 6 ----------
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
clc;
clear all;
close all;

%% Parameter
imgBoardFile = 'desk.JPG';
sizePic = [21.6 27.9];

%% Step1: Pick four corners of the paper on the desk
imgBoard = imread(imgBoardFile);

figure(1);
imshow(imgBoard);
```

```
p=ginput(4); % Please click on the 4 corners of the paper in the following
             % order: upper left of the paper, uper right, lower right,
             % lower left (clockwise)
corner = p;

%% Step2: Compute H
H = ComputeH6(sizePic, corner);
H_inv = inv(H');

%% Step3: Pick the corners of the desk for measurement
figure(1);
imshow(imgBoard);
p_measure1=ginput(2); % Please select the short side of the table
p_measure2=ginput(2); % Please select the long side of the table

%% Step4:measurment for short and long edges of the table
% corners of the paper and corners of the desk in homogenuous coordinate
corner_hg = [corner [1;1;1;1]];
p_measure1_hg = [p_measure1 [1;1]];
p_measure2_hg = [p_measure2 [1;1]];

% solve x'=Hx. Here, we have the coordinates of different points in our
% image. They are x'. So, we want to calculate their real coordinates:
% x=H^-1x'
corner_r = H_inv*corner_hg';
p_measure1_r = H_inv*p_measure1_hg';
p_measure2_r = H_inv*p_measure2_hg';

% calculate the coordinates of the points in the cartesian coordinate
corner_cr = corner_r(1:2,:)./corner_r(3,:);
p_measure1_cr = p_measure1_r(1:2,:)./p_measure1_r(3,:);
p_measure2_cr = p_measure2_r(1:2,:)./p_measure2_r(3,:);

d1 = norm(corner_cr(:,2)-corner_cr(:,1)); % As we expect, corner_cr is the
% same with actual corners of the paper : 21.6 cm and 27.9 cm

% measurment for short edge of the table
d2 = norm(p_measure1_cr(:,2)-p_measure1_cr(:,1));
% measurment for short edge of the table
d3 = norm(p_measure2_cr(:,2)-p_measure2_cr(:,1));

%%
function H = ComputeH6(sizePic, corner)
    points1 = [0,sizePic(2);sizePic(1),sizePic(2);sizePic(1),0;0,0];
    points2 = corner;
    x_xp = points1(:,1).*points2(:,1);
    x_yp = points1(:,1).*points2(:,2);
    y_xp = points1(:,2).*points2(:,1);
    y_yp = points1(:,2).*points2(:,2);

    A = zeros(size(points1,1)*2,9);
    A(1:2:end,3) = 1;
    A(2:2:end,6) = 1;
    A(1:2:end,1:2) = points1;
    A(2:2:end,4:5) = points1;
    A(1:2:end,7) = -x_xp;
    A(1:2:end,8) = -y_xp;
    A(2:2:end,7) = -x_yp;
```

```
    A(2:2:end,8) = -y_yp;
    A(1:2:end,9) = -points2(:,1);
    A(2:2:end,9) = -points2(:,2);

    T = null(A);
    T = T/T(end);
    H = (reshape(T,3,3));
end
```