Name: Armina Soleymani

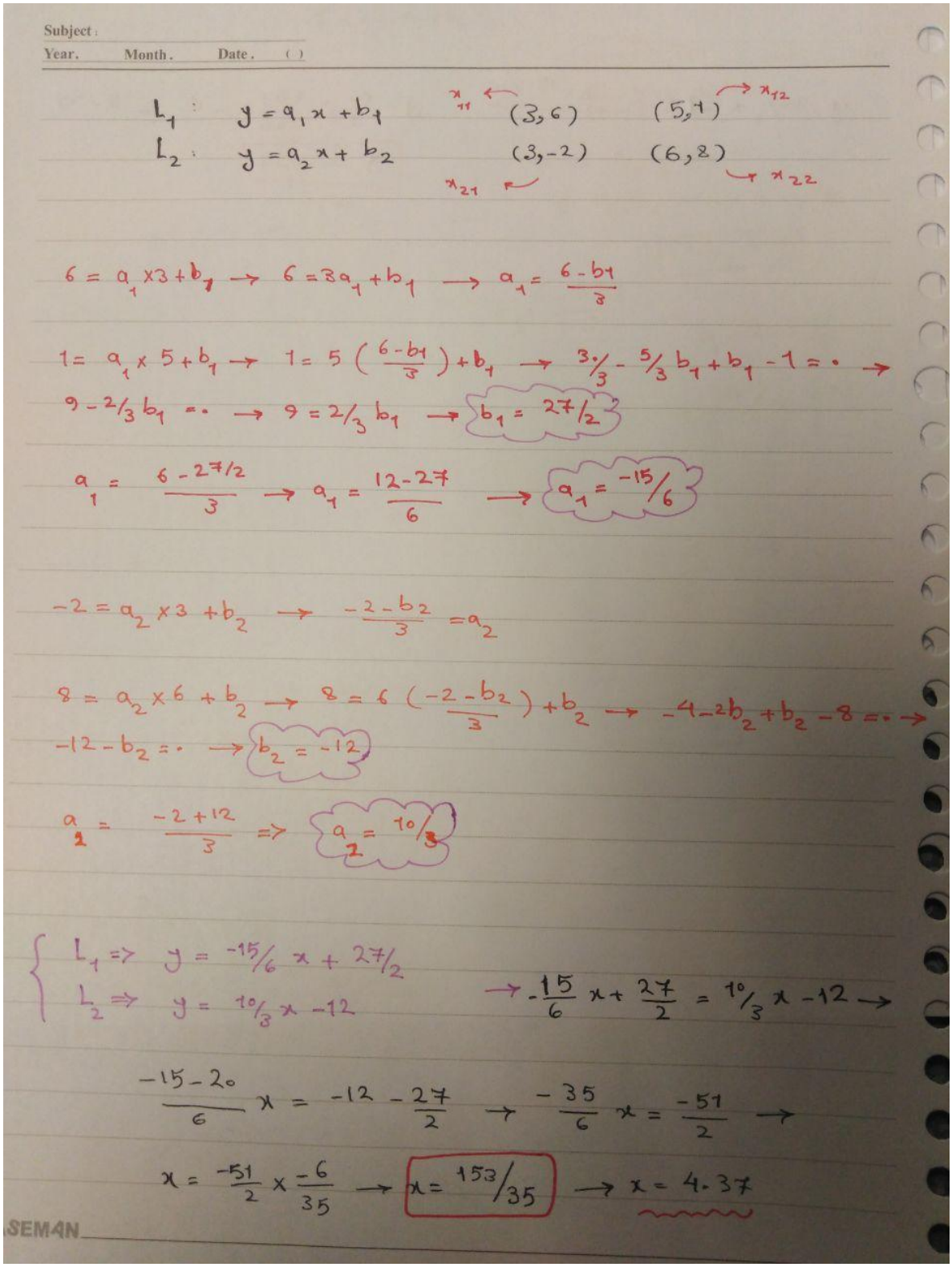
Degree: PH

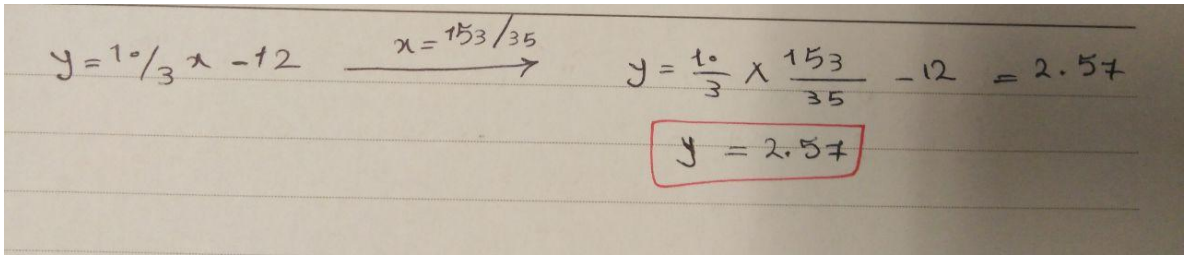
ID: 20850595

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

```
clc;
clear;
close all;
```

```
a= [3,6,1];
b= [5,1,1];
L1= cross(a,b);

d= [3,-2,1];
e= [6,8,1];
L2= cross(d,e);

P= cross(L1,L2);
P= P/P(3);

intersection:(4.37,2.57,1)
```

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

```
clc;
clear;
close all;

Conic= [1/4,0,-3/4;0,1/9,-2/9;-3/4,-2/9,25/36];
point= [1,5,1];
l= Conic*point';  %l=c*x: tangent line to a conic
lx= [0,1,0]; %x-axix: y=0, ax+by+c=0, so a=0, b=1, c=0

intersec= cross(l,lx);
intersec= intersec/intersec(3);

intersection: [-2.33,0,1]
```

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

```
clc;
clear;
close all;

Conic= [1/4,0,-3/4;0,1/9,-2/9;-3/4,-2/9,25/36];
point1= [1,5,1];
l1= Conic*point1';  %l=c*x: tangent line to a conic

point2=[1,-1,1];
l2= Conic*point2';

P= cross(l1,l2);
intersec= P/P(3);
```

```
intersection: [-1,2,1]
```

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

$$l_1 : y_1 = 1.25x_1 + 3$$
$$l_2 : y_2 = 0.4x_2 - 5$$
$$l_3 : y_3 = -2.25x_3 + 5$$
$$l_4 : y_4 = -x_4 - 8$$

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.

```
clc;
clear all;
close all
filename = 'quadrangle';
n=30;                           % number of points for each line
x1s=-100;                       % start point of line #1
x1e=100;                        % end point of line #1
x1=linspace(x1s,x1e,n);         % x values of Cartesian coodinate of line #1

a11 = 1.25;                     % slope of line #1
a12 = 3;                        % distance from origin of line #1
a21 = 0.4;                      % slope of line #2
a22 = -5;                       % distance from origin of line #2
a31 = -2.25;                    % slope of line #3
a32=5;                          % distance from origin of line #3
a41=-1;                         % slope of line #4
a42=-8;                         % distance from origin of line #4

x2s=-100;                     % start point of line #2
x2e=100;                      % end point of line #2
x2=linspace(x2s,x2e,n);       % x values of Cartesian coordinate of line #2

x3s=-100;                     % start point of line #3
x3e=100;                      % end point of line #3
x3=linspace(x3s,x3e,n);       % x values of Cartesian coordinate of line #3

x4s=-100;                     % end point of line #4
x4e=100;                      % end point of line #4
x4=linspace(x4s,x4e,n);       % x values of Cartesian coordinate of line #4

y1=a11*x1+a12;                % eq. for line #1
y2=a21*x2+a22;                % eq. for line #2
y3=a31*x3+a32;                % eq. for line #3
y4=a41*x4+a42;                % eq. for line #4

%plot(x1,y1,x2,y2,x3,y3,x4,y4)

x(:,1)=x1;
x(:,2)=x2;
x(:,3)=x3;
```

```
x(:,4)=x4;

y(:,1)=y1;
y(:,2)=y2;
y(:,3)=y3;
y(:,4)=y4;

y=y';
x=x';
Coeff=zeros(2,size(x,1));
for k = 1:size(x,1)
    Coeff(:,k) = [x(k,:)' ones(n,1)] \ y(k,:)';    % Making a matrix based on
                                                   % coefficients of x and
                                                   % distances from origin for
each line
end

%
Intersec = zeros(size(Coeff,2),size(Coeff,2),size(Coeff,1));
for k1 = 1:size(Coeff,2)
    for k2 = 1:size(Coeff,2)
     % solving for variables in perspective coordinate which
     % are intersections of lines in 2D Cartesian coordinate
        Intersec(k1,k2,:) = [-Coeff(1,[k1 k2])' [1;1]] \ Coeff(2,[k1 k2])';

    end
end
X = Intersec(:,:,1);
Y = Intersec(:,:,2);
Names = sprintfc('L_%d',1:4);
XInt = table(X(:,1),X(:,2),X(:,3),X(:,4),'VariableNames',Names,'RowNames',Names)
YInt = table(Y(:,1),Y(:,2),Y(:,3),Y(:,4),'VariableNames',Names,'RowNames',Names)

% making quadrangle
% first finding colinear intersecting points
for i=1:size(x,1)
    for j=1:size(X,1)
        if(j~=size(X,1))
            DiffX=X(j+1,i)-X(j,i);
            DiffY=Y(j+1,i)-Y(j,i);
        else
            DiffX=X(j,i)-X(j-1,i);
            DiffY=Y(j,i)-Y(j-1,i);
        end
        m_slopes(j,i)=((DiffY./DiffX));
    end
end

Indx=(~isnan(m_slopes));
for i=1:size(x,1)
    for j=1:size(X,1)
        ff=find(Indx(j,:),1,'first');
        if(~isempty(ff))
            ki= squeeze(ff) ;
        else
            ki= 0;
        end
        gg=find(Indx(j,:),1,'last');
```

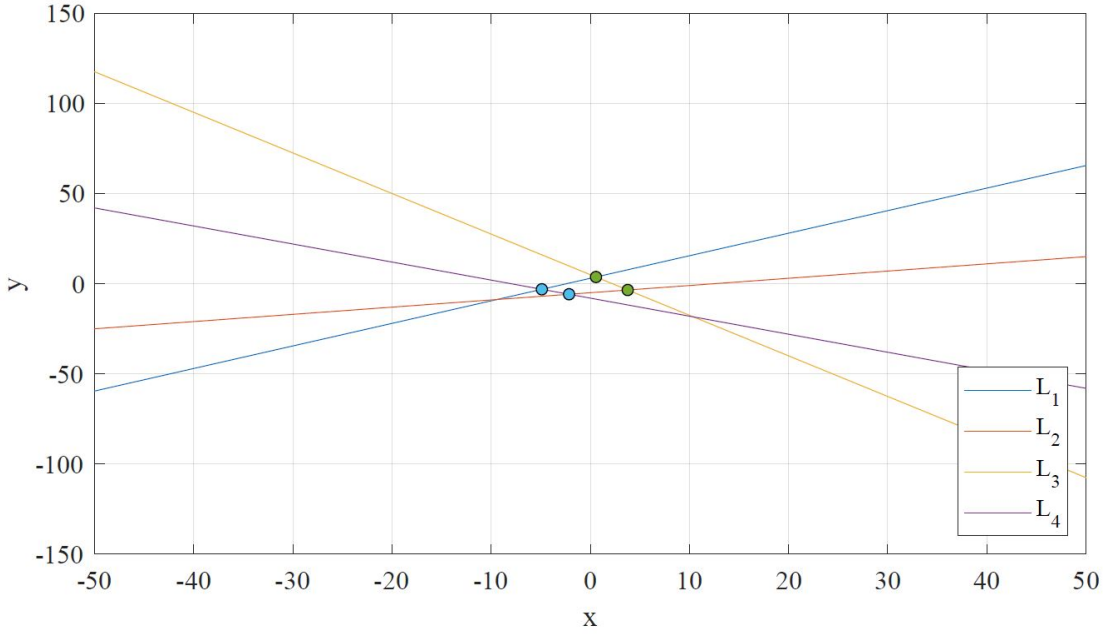
```
        if(~isempty(gg))
            mi= squeeze(gg);
        else
            mi=0;
        end
        if((ki==mi-1)&&(ki~=0)&&(mi~=0))
            xsol(i,j)=X(i,j);
            ysol(i,j)=Y(i,j);
            id(i,j)=j;
        end

    end
end
Xsol=triu(xsol);
Ysol=triu(ysol);
Xsol = Xsol(1:2,3:end);
Ysol = Ysol(1:2,3:end);
Names = sprintfc('L_%d',1:4);
xv = x1; %linspace(x1s,x1e,n);
Ym = Coeff(1,:)'*xv + Coeff(2,:)';

h1=figure('units','normalized','outerposition',[0 0 1 1],'visible','off');
plot(xv,Ym)
hold on
%plot(X, Y, '+m')                        % Plot Intersections
h=plot(Xsol, Ysol, 'o','MarkerSize',5,...
    'MarkerEdgeColor','k','MarkerFaceColor','k');
set(h, {'MarkerFaceColor'}, get(h,'Color'));
hold off
xlim([x1s/2 x1e/2])
xlabel('x')
ylabel('y')
set(gca,'FontSize',12,'FontName','Times')
grid
legend(Names, 'Location','SE')
fig = gcf;
fig.Units               = 'centimeters';
% fig.Position(3)         = 15;
% fig.Position(4)         = 10;
%tightfig(gcf);
print (gcf,filename,'-dpdf','-bestfit')
close(h1)
```

X_Intersection =

	L_1	L_2	L_3	L_4
L_1	NaN	-9.4118	0.57143	-4.8889
L_2	-9.4118	NaN	3.7736	-2.1429
L_3	0.57143	3.7736	NaN	10.4
L_4	-4.8889	-2.1429	10.4	NaN

Y_Intersection =

	L_1	L_2	L_3	L_4
L_1	NaN	-8.7647	3.7143	-3.1111
L_2	-8.7647	NaN	-3.4906	-5.8571
L_3	3.7143	-3.4906	NaN	-18.4
L_4	-3.1111	-5.8571	-18.4	NaN



Problem 4: Linear Algebra (20 points)

(a) Find a transpose of inverse H

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

```
clear all;
clc;
close all;
syms  l1 l2 l3 real
H=[1,0,0;0,1,0;l1,l2,l3];
g= inv(H);
f= g';

f= [1,0,-l1/l3; 0,1,-l2/l3; 0,0,1/l3]
```

(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 space of (A): the subspace of Rn spanned by the row vectors of A.

null space of (A): the set of all vectors x which satisfy Ax=0.

```
clc;
clear;
close all;
A= [1,3,2,1,4;3,3,5,10,1;-1,5,1,2,-4];
R= rank(A);
N= null(A);

Rank: 3
Null:
   -0.3496   -0.8620
    0.1314   -0.1977
   -0.7776    0.4427
    0.4272    0.0844
    0.2708    0.1213
Nullity: 2 (#columns - Rank: 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.

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

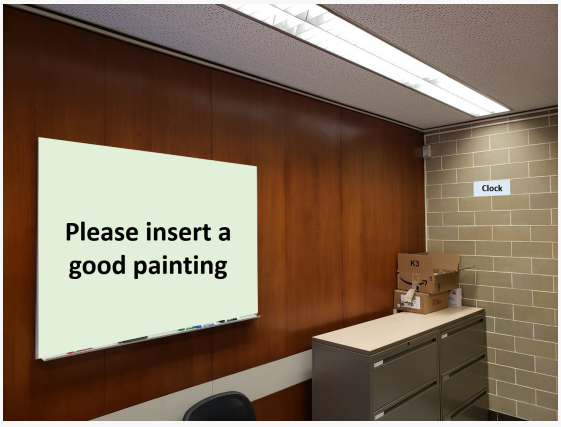
```
clc;
clear;
close all;
A= [1,0,0,0,2;-2,1,-3,-2,-4;0,5,-14,-9,0;2,10,-28,-18,4];
R= rank(A);
N= null(A);

Rank: 3
Because Nullity= #columns - Rank, and 3<5 so  x exist
Nullity= 2
Null (vectorx)
    0.1568   -0.8806
    0.5684    0.1012
    0.5684    0.1012
   -0.5684   -0.1012
   -0.0784    0.4403
```

(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.

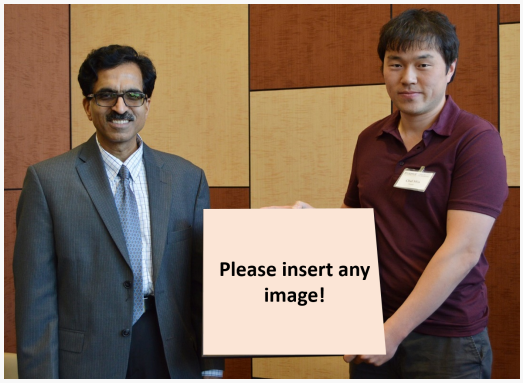
$$A = \begin{bmatrix} 1 & 0 & 0 & 0 & 2 \\ -2 & 1 & -3 & -2 & -4 \end{bmatrix}$$

```
clc;
clear;
close all;
A= [1,0,0,0,2;-2,1,-3,-2,-4];
R= rank(A);
N= null(A);

Rank: 2
Because Nullity= #columns - Rank, and 2<5 so  x exist
Nullity= 3
Null (vectorx):
    0.0000    0.0000   -0.8944
    0.8018    0.5345    0.0000
    0.4927   -0.3382   -0.0000
   -0.3382    0.7745   -0.0000
   -0.0000   -0.0000    0.4472
```

Problem 5: Image Overlay (20 points)



(a) original image: *prob5_1.jpg*



(b) original image: *prob5_2.jpg*

```
clear; close all; clc; format shortG;

imgBoardFile = 'my_office.jpg';
imgPicFile = 'm.jpg';

info = imfinfo(imgPicFile);
sizePic = [info.Width info.Height];
imgPic=imread(imgPicFile);
figure; imshow(imgBoardFile);
imgBoard = imread(imgBoardFile);
figure(1); imshow(imgBoard);

p = drawpolygon('LineWidth',5,'Color','c'); % pick white board points as follow:
upper left,upper right,lower left,lower right
```

```
corner = p.Position;
H = ComputeH(corner, sizePic);
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, 'result.jpg');


function H = ComputeH(C,S);
A=zeros(size(C,1)*2,9);
A(1:2:end,9)=-C(:,1);
A(2:2:end,9)=-C(:,2);
A(1:2:end,3)=1;
A(2:2:end,6)=1;
A(1,1:2)=[1,1];
A(3,1:2)=[S(1),1];
A(5,1:2)=[S(1),S(2)];
A(7,1:2)=[1,S(2)];
A(2,4:5)=[1,1];
A(4,4:5)=[S(1),1];
A(6,4:5)=[S(1),S(2)];
A(8,4:5)=[1,S(2)];
x1x2=C(:,1).*[1;S(1);S(1);1];
x1y2=C(:,1).*[1;1;S(2);S(2)];
y1x2=C(:,2).*[1;S(1);S(1);1];
y1y2=C(:,2).*[1;1;S(2);S(2)];
A(1:2:end,7)=-x1x2;
A(1:2:end,8)=-x1y2;
A(2:2:end,7)=-y1x2;
A(2:2:end,8)=-y1y2;
K = null(A)
% [junk1,junk2,V]= svd(A);
% h=V(:,end)./V(end,end);
H= reshape(K,3,3)
end
```

```
clear; close all; clc; format shortG;

imgBoardFile = 'q.jpg';
imgPicFile = 'clock.png';
```

```
info = imfinfo(imgPicFile);
sizePic = [info.Width info.Height];
imgPic=imread(imgPicFile);
figure; imshow(imgBoardFile);
imgBoard = imread(imgBoardFile);
figure(1); imshow(imgBoard);

p = drawpolygon('LineWidth',5,'Color','c'); % pick white board points as follow:
upper left,upper right,lower left,lower right
corner = p.Position;
H = ComputeH(corner, sizePic);
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, 'result.jpg');


function H = ComputeH(C,S);
A=zeros(size(C,1)*2,9);
A(1:2:end,9)=-C(:,1);
A(2:2:end,9)=-C(:,2);
A(1:2:end,3)=1;
A(2:2:end,6)=1;
A(1,1:2)=[1,1];
A(3,1:2)=[S(1),1];
A(5,1:2)=[S(1),S(2)];
A(7,1:2)=[1,S(2)];
A(2,4:5)=[1,1];
A(4,4:5)=[S(1),1];
A(6,4:5)=[S(1),S(2)];
A(8,4:5)=[1,S(2)];
x1x2=C(:,1).*[1;S(1);S(1);1];
x1y2=C(:,1).*[1;1;S(2);S(2)];
y1x2=C(:,2).*[1;S(1);S(1);1];
y1y2=C(:,2).*[1;1;S(2);S(2)];
A(1:2:end,7)=-x1x2;
A(1:2:end,8)=-x1y2;
A(2:2:end,7)=-y1x2;
A(2:2:end,8)=-y1y2;
K = null(A)
% [junk1,junk2,V]= svd(A);
% h=V(:,end)./V(end,end);
```

```
H= reshape(K,3,3)
end
```


```
clear; close all; clc; format shortG;

imgBoardFile = 'award.jpg';
imgPicFile = 'a.jpg';
info = imfinfo(imgPicFile);
sizePic = [info.Width info.Height];
imgPic=imread(imgPicFile);
figure; imshow(imgBoardFile);
imgBoard = imread(imgBoardFile);
figure(1); imshow(imgBoard);

p = drawpolygon('LineWidth',5,'Color','c'); % pick white board points as follow:
upper left,upper right,lower left,lower right
corner = p.Position;
H = ComputeH(corner, sizePic);
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, 'result.jpg');


function H = ComputeH(C,S);
A=zeros(size(C,1)*2,9);
A(1:2:end,9)=-C(:,1);
A(2:2:end,9)=-C(:,2);
A(1:2:end,3)=1;
A(2:2:end,6)=1;
A(1,1:2)=[1,1];
A(3,1:2)=[S(1),1];
A(5,1:2)=[S(1),S(2)];
A(7,1:2)=[1,S(2)];
A(2,4:5)=[1,1];
A(4,4:5)=[S(1),1];
A(6,4:5)=[S(1),S(2)];
A(8,4:5)=[1,S(2)];
x1x2=C(:,1).*[1;S(1);S(1);1];
x1y2=C(:,1).*[1;1;S(2);S(2)];
y1x2=C(:,2).*[1;S(1);S(1);1];
y1y2=C(:,2).*[1;1;S(2);S(2)];
A(1:2:end,7)=-x1x2;
A(1:2:end,8)=-x1y2;
A(2:2:end,7)=-y1x2;
A(2:2:end,8)=-y1y2;
K = null(A)
% [junk1,junk2,V]= svd(A);
% h=V(:,end)./V(end,end);
H= reshape(K,3,3)
end
```

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

```
clear; close all; clc; format shortG;
my_pic = 'table.jpg';
paper_width_Height = [11,16];

image = imread(my_pic);
figure(1);
imshow(image);

%pick the paper corner as following: upper left,upper right,lower rigth,lower
left
p1 = drawpolygon('LineWidth',5,'Color','c');
p = p1.Position;

%compute H
H = computeH (paper_width_Height,p)
H_inv = inv(H');

%pick the table width and height
figure(1);
imshow(my_pic);
table_width = ginput(2); %width(short size)
table_height = ginput(2); %height(long side)

%transfer to homogeneous coordinate
p_Homog = [p,[1;1;1;1]];
table_width_Homog = [table_width,[1;1]];
table_height_Homog = [table_height,[1;1]];

%find real coordinate (X): X'=HX so X=inv(H)*X'
```

```
p_real = H_inv * p_Homog';
table_width_real = H_inv * table_width_Homog';
table_height_real = H_inv * table_height_Homog';

%transfer to cartesian coordinate
p_cart = p_real(1:2,:) ./ p_real(3,:);
table_width_cart = table_width_real(1:2,:) ./ table_width_real(3,:);
table_height_cart = table_height_real(1:2,:) ./ table_height_real(3,:);

%measure width and height for table
table_width = norm(table_width_cart(:,2)-table_width_cart(:,1));
table_height = norm(table_height_cart(:,2)-table_height_cart(:,1));

table_width
table_height

function H = computeH (paper_width_Height,p)
q =
[0,paper_width_Height(2);paper_width_Height(1),paper_width_Height(2);paper_width
_Height(1),0;0,0];
A= zeros(size(q,1)*2,9);
A(1:2:end,3)=1;
A(2:2:end,6)=1;
A(1:2:end,1:2)=q;
A(2:2:end,4:5)=q;
xaxb=q(:,1).*p(:,1);
xayb=q(:,1).*p(:,2);
yaxb=q(:,2).*p(:,1);
yayb=q(:,2).*p(:,2);
A(1:2:end,7)=-xaxb;
A(1:2:end,8)=-xayb;
A(2:2:end,7)=-yaxb;
A(2:2:end,8)=-yayb;
A(1:2:end,9)=-p(:,1);
A(2:2:end,9)=-p(:,2);
K = null(A)
H = reshape(K,3,3)
end


Real width: 40, Measured width:40.313
Real Height: 44, Measured Height:43.469
```