Affine measurements

This code demonstrate the estimation of vanishing points and vanishing lines to take affine measurements from an image. Specifically this code allows to estimate the height of an object standing on the ground, provided: - the height of a reference object standingh on the ground, - a vertical vanishing point, - the vanishing line of the ground.

Note that for more accurate results the image should be undirstorted .\\°//.\\°//.\\°//.\\°//.\\°//.\\°//.\\°//.\\°//.\\°//.\\°//.\\°//.\\°//.\\°//.

Image Analysis and Computer Vision Politecnico di Milano Luca Magri for comments and suggestions please send an email to luca.magri@polimi.it

.\\°//.\\°//.\\°//.\\°//.\\°//.\\°//.\\°//.\\°//.\\°//.\\°//.\\°//.\\°//.\\°//.

Contents


```
close all % close all the figures
clear % clear the workspace
clc % clear the command window
```

load image

```
im = imread('poli2.jpg');
im = imresize(im,0.6);
figure(1);
imshow(im);
```

draw "vertical" segments in the image

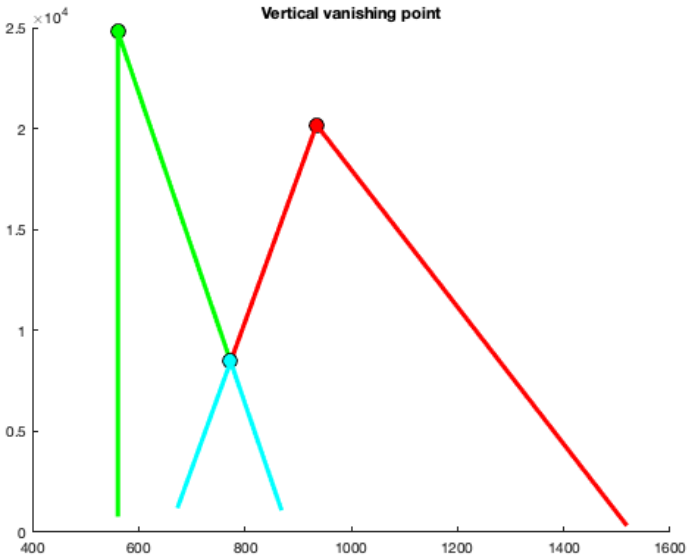
i.e. segments in image that comes from vertical lines drag segments on the image, you can adjust the poistions a little bit, when you are happy of the result press enter.

```
numSegments = 4; % the minimum number of segments is two, the higher the better the result
verticalLines = nan(numSegments,3); % store vertical parallel lines
endPoints = nan(2,numSegments); % store the endpoints of the segments for visualization purpose
count = 1;
while(count <=numSegments)
    figure(1);
    title(['Draw ', num2str(numSegments),' vertical segments: step ',num2str(count) ]);
    seg = drawline('Color','b');
    % convert the end points of the segments into the coefficients of
    % the line
    endPoints(:,count) = seg.Position(1,:)';
    % switch to homoegeous coordinates
    a = [seg.Position(1,:)';1];
    b = [seg.Position(2,:)';1];
    % get the parameters of the line
    l = cross(a,b);
    l = l./norm(l); %normalize it, just to remove the scale factor
    verticalLines(count, :) = l';
    count = count +1;
end
fprintf('Press enter to continue\n');
pause
```

```
Press enter to continue
```

pairwise vanishing points are different because segments are noisy

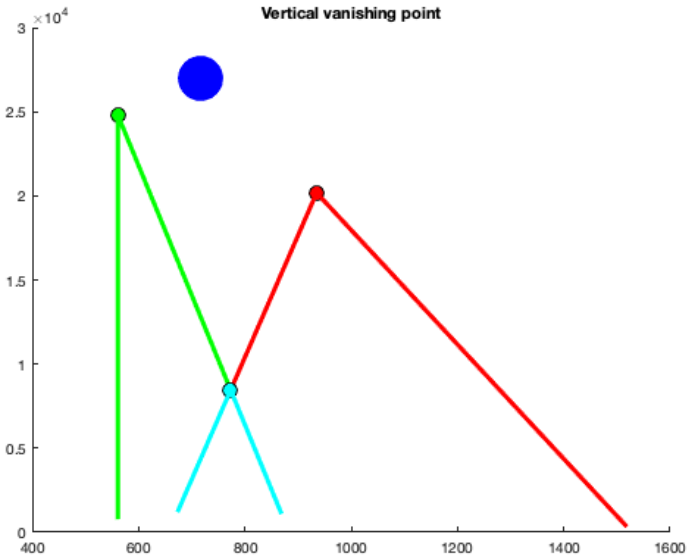
```
% compute vanishing points from pairs of images
v1 = cross(verticalLines(1,:),verticalLines(2,:));
v1 = v1./v1(3);
v2 = cross(verticalLines(3,:),verticalLines(4,:));
v2 = v2./v2(3);
v3 = cross(verticalLines(1,:),verticalLines(4,:));
v3 = v3./v3(3);


figure;
%imshow(im);
hold all;

plot(v1(1),v1(2),'ko','MarkerSize',10,'MarkerFaceColor','r');
line([endPoints(1,1),v1(1)],[endPoints(2,1),v1(2)],'Color','r','LineWidth',3);
line([endPoints(1,2),v1(1)],[endPoints(2,2),v1(2)],'Color','r','LineWidth',3);

plot(v2(1),v2(2),'ko','MarkerSize',10,'MarkerFaceColor','g');
line([endPoints(1,3),v2(1)],[endPoints(2,3),v2(2)],'Color','g','LineWidth',3);
line([endPoints(1,4),v2(1)],[endPoints(2,4),v2(2)],'Color','g','LineWidth',3);

plot(v3(1),v3(2),'ko','MarkerSize',10,'MarkerFaceColor','c');
line([endPoints(1,1),v3(1)],[endPoints(2,1),v3(2)],'Color','c','LineWidth',3);
line([endPoints(1,4),v3(1)],[endPoints(2,4),v3(2)],'Color','c','LineWidth',3);

title('Vertical vanishing point');
% all the vanishing points are different, because they refer to a
% how can we get a unique vanishing point?
```



estimate a unique vertical vanishing point

we can minimize an algebraic error

```
[u,s,v]=svd(verticalLines);
V_vert = v(:,end);
V_vert = V_vert/V_vert(3);


figure(gcf);
hold all;
plot(V_vert(1),V_vert(2),'bo','MarkerSize',30,'MarkerFaceColor','b');

% but this is still suboptimal, it would be better to minimize a geometric
% error
```

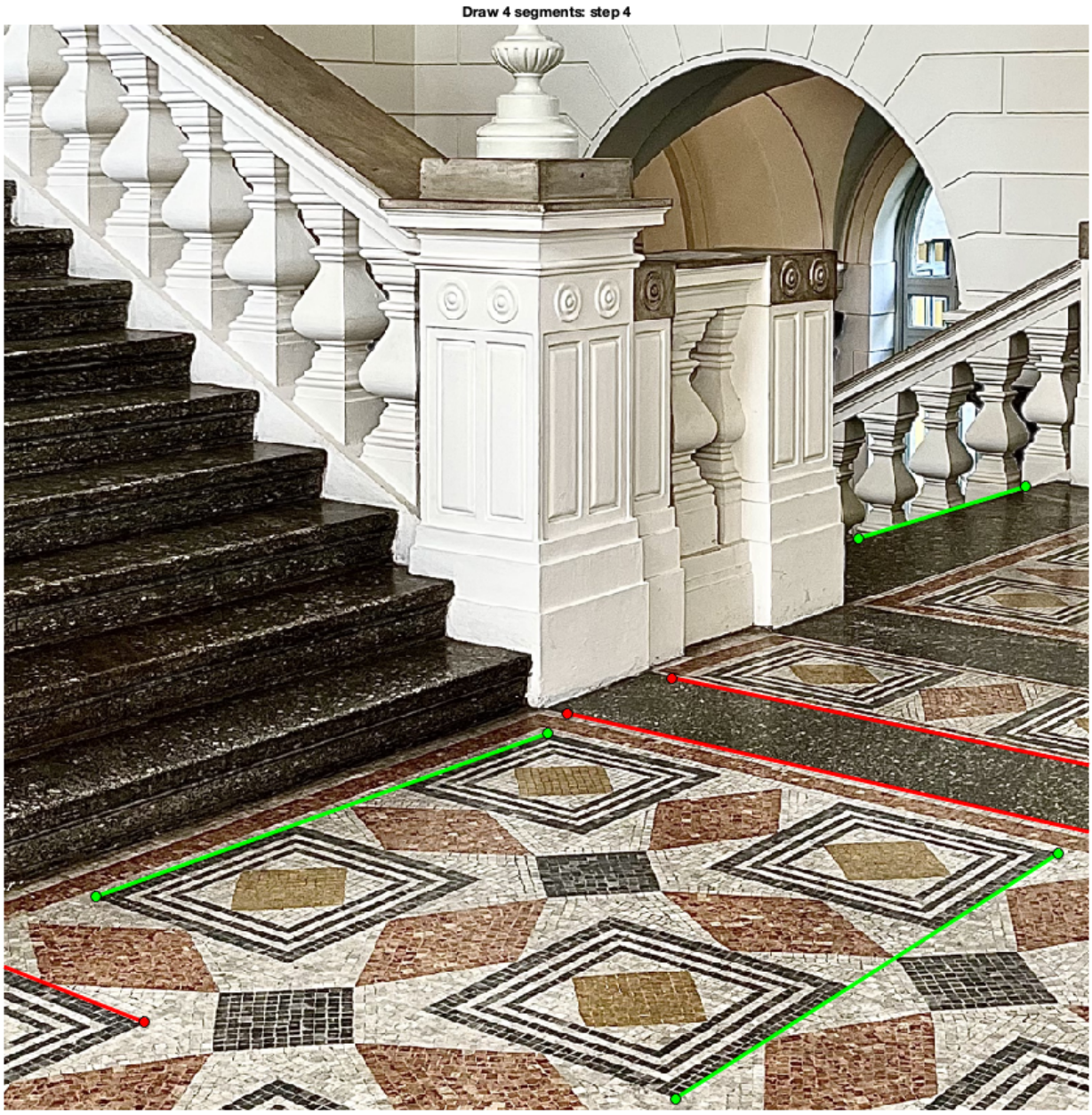


draw segments that comes from parallel lines on the floor

```
f = 2; % number of families of  lines. Two are enough to get the horizon
numSegments = 4;
parallelLines = cell(f,1); % store parallel lines
fprintf(['Draw ', num2str(f) , ' families of parallel segments\n']);
col = 'rgm';
figure;
imshow(im)
for i = 1:f
    count = 1;
    parallelLines{i} = nan(numSegments,3);
    while(count <=numSegments)
        figure(gcf);
        title(['Draw ', num2str(numSegments),' segments: step ',num2str(count) ]);
        seg = drawline('Color',col(i));
```

```
        % convert the end points of the segments into the coefficient of
        % the line
        endPoints = seg.Position;
        % switch to homoegeous coordinates
        a = [endPoints(1,:)';1];
        b = [endPoints(2,:)';1];
        % get the line
        l = cross(a,b);
        l = l./norm(l); %normalize it, just to remove the scale factor
        parallelLines{i}(count, :) = l;
        count = count +1;
    end
    fprintf('Press enter to continue\n');
    pause
end
```

```
Draw 2 families of parallel segments
Press enter to continue
Press enter to continue
```



compute the vanishing points

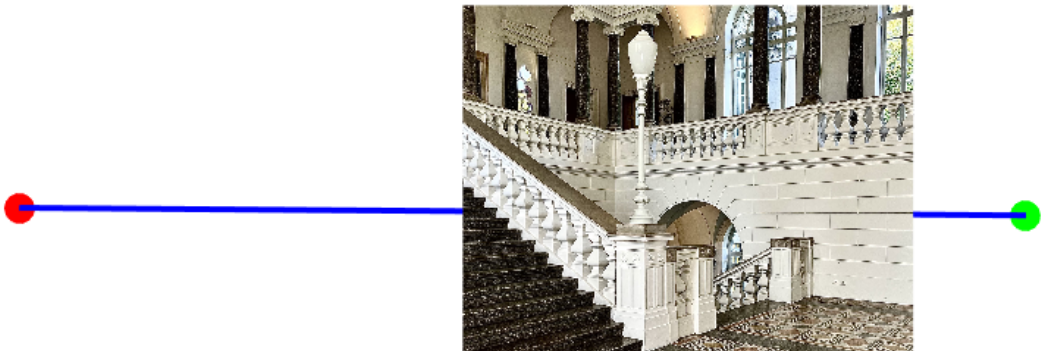
```
V = nan(3,f);
for i =1:f
    % look for a point that belongs to all the lines in the same family
    % this mean you have to solve a system.
    [u,s,v]=svd(parallelLines{i});
    V(:,i) = v(:,end);
```

```
    V(:,i)= V(:,i)./V(3,i);
end
```

compute the horizon

```
horizon = cross(V(:,1),V(:,2));


figure;
hold all;
for i = 1:f
    plot(V(1,i),V(2,i),'o','Color',col(i),'MarkerSize',20,'MarkerFaceColor',col(i));
end
line(V(1,:),V(2,:),'Color','b','Linewidth',4)
%hold all;
imshow(im);
```

Draw 4 segments: step 4



build the rectification matrix

```
H = [eye(2),zeros(2,1); horizon(:)'];
% we can check that H^-T* imLinfty is the line at infinity in its canonical
% form:
fprintf('The vanishing line is mapped to:\n');
disp(inv(H)'*horizon);
```

```
The vanishing line is mapped to:
     0
     0
     1
```

rectify the image and show the result

```
tform = projective2d(H');
J = imwarp(im,tform);

figure;
imshow(J);
```

.

```
figure;
imshow(im);
title('select a reference object')
seg1 = drawline('Color','c');
seg2 = drawline('Color','m');

b1 = [seg1.Position(1,:)';1];
t1 = [seg1.Position(2,:)';1];
b2 = [seg2.Position(1,:)';1];
t2 = [seg2.Position(2,:)';1];
```

dispaly the selected points

```
TXT_OFFSET = 20;
figure;
imshow(im);
hold all;
plot(b1(1),b1(2),'c.','MarkerSize',20,'LineWidth',5);
text(b1(1)+TXT_OFFSET,b1(2),'b1','FontSize',30,'Color','c');
plot(t1(1),t1(2),'.','Color','c','MarkerSize',20,'LineWidth',5);
text(t1(1)+TXT_OFFSET,t1(2),'t1','FontSize',30,'Color','c');
plot(b2(1),b2(2),'m.','MarkerSize',20,'LineWidth',5);
text(b2(1)+TXT_OFFSET,b2(2),'b2','FontSize',30,'Color','m');
plot(t2(1),t2(2),'m.','MarkerSize',20,'LineWidth',5);
text(t2(1)+TXT_OFFSET,t2(2),'t2','FontSize',30,'Color','m');
```

compute the relative distance

```
u = cross(cross(b1,b2),horizon);
u = u./u(3);

l2 = cross(V_vert,b2);
t1_tilde = cross(cross(t1,u),l2);
t1_tilde = t1_tilde/t1_tilde(3);

% compute the distance from b2
dist_t1_tilde = norm(t1_tilde-b2);
dist_t2 = norm(t2-b2);
dist_v = norm(V_vert -b2);

% compute 1D projective transfomration mapping the vanishing point to
% infinity
% (0,1)-> (0,1)
% (V_vert,1)->(1,0);
H = [1 0; 1 -dist_v];

H*[dist_v;1]

% compute the distance ratio
d1_d2_ratio = dist_t1_tilde*(dist_v-dist_t2)/(dist_t2*(dist_v-dist_t1_tilde));

%d2 = 1/d1_d2_ratio
lenght1 = 122;
lenght2 =lenght1/d1_d2_ratio
```

```
ans =

   1.0e+04 *

    2.5787
         0


lenght2 =

  124.1234
```

```
TXT_OFFSET = 20;
FNT_SIZE = 30;
c1 = 'w';
c2 = 'm';
c3 = 'b';
figure;
imshow(im);
hold all;
line(V(1,:),V(2,:),'Color',c3,'Linewidth',2);
line([b1(1),t1(1)],[b1(2),t1(2)],'Color',c1,'Linewidth',2);
line([b2(1),t2(1)],[b2(2),t2(2)],'Color',c2,'Linewidth',2);
plot(b1(1),b1(2),'.','Color',c1,'MarkerSize',20,'LineWidth',2);
text(b1(1)+TXT_OFFSET,b1(2),'b1','FontSize',FNT_SIZE,'Color',c1);
plot(t1(1),t1(2),'.','Color',c1,'MarkerSize',20,'LineWidth',5);
text(t1(1)+TXT_OFFSET,t1(2),'t1','FontSize',FNT_SIZE,'Color',c1);
plot(b2(1),b2(2),'.','Color',c2,'MarkerSize',20,'LineWidth',5);
text(b2(1)+TXT_OFFSET,b2(2),'b2','FontSize',FNT_SIZE,'Color',c2);
plot(t2(1),t2(2),'.','Color',c2,'MarkerSize',20,'LineWidth',5);
text(t2(1)-TXT_OFFSET,t2(2)+TXT_OFFSET,'t2','FontSize',30,'Color',c2);
plot(u(1),u(2),'o','Color',c3,'Markersize',20,'MarkerFaceColor',c3)
text(u(1)+TXT_OFFSET,u(2),'u','FontSize',FNT_SIZE,'Color',c3);
plot(t1_tilde(1),t1_tilde(2),'d','Color',c3,'MarkerSize',10,'LineWidth',5,'MarkerFaceColor', c3);
text(t1_tilde(1)+TXT_OFFSET,t1_tilde(2),'t1 tilde','FontSize',30,'Color',c3);
line([b2(1),u(1)],[b2(2),u(2)],'Color',c3,'Linewidth',2);
line([t1_tilde(1),u(1)],[t1_tilde(2),u(2)],'Color',c3,'Linewidth',2);
hold on;
```


```
c1= 'r';
figure;
imshow(im);
hold all;
line([b1(1),t1(1)],[b1(2),t1(2)],'Color',c1,'Linewidth',2);
text(0.5*(b1(1)+t1(1)),0.5*(b1(2)+t1(2)),num2str(lenght1,"%i"),'FontSize',FNT_SIZE,'Color',c1);
line([b2(1),t2(1)],[b2(2),t2(2)],'Color',c1,'Linewidth',2);
text(0.5*(b2(1)+t2(1)),0.5*(b2(2)+t2(2)),num2str(round(lenght2)),'FontSize',FNT_SIZE,'Color',c1);
```

Published with MATLAB® R2021b