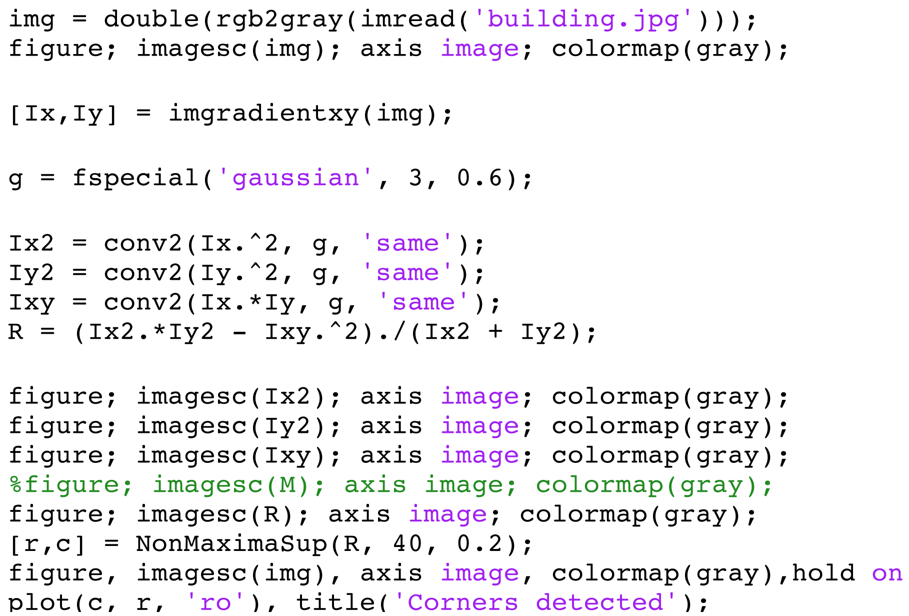
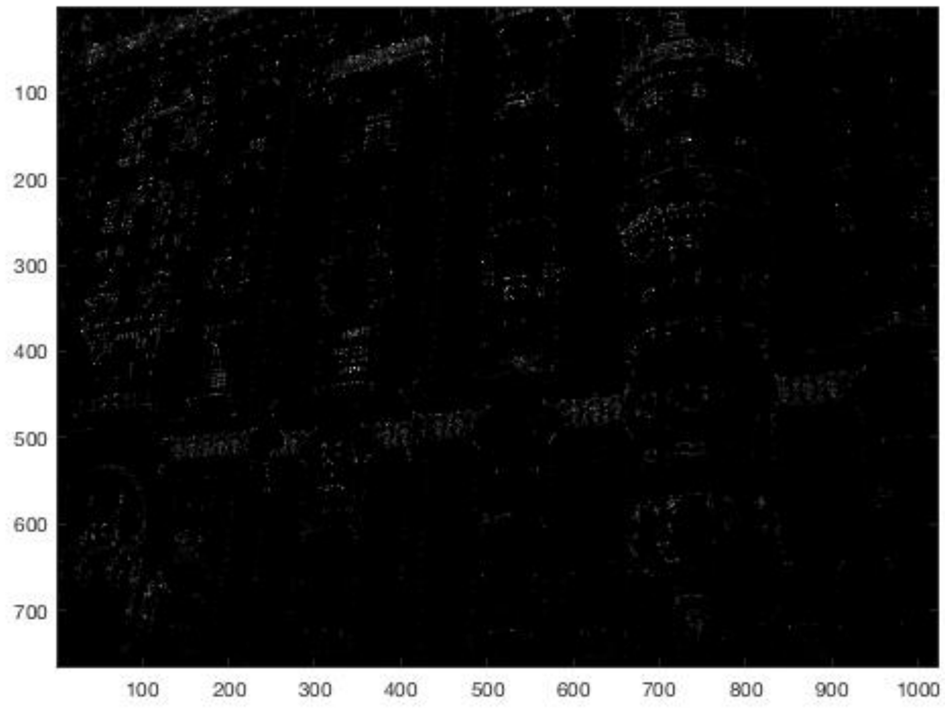
1.

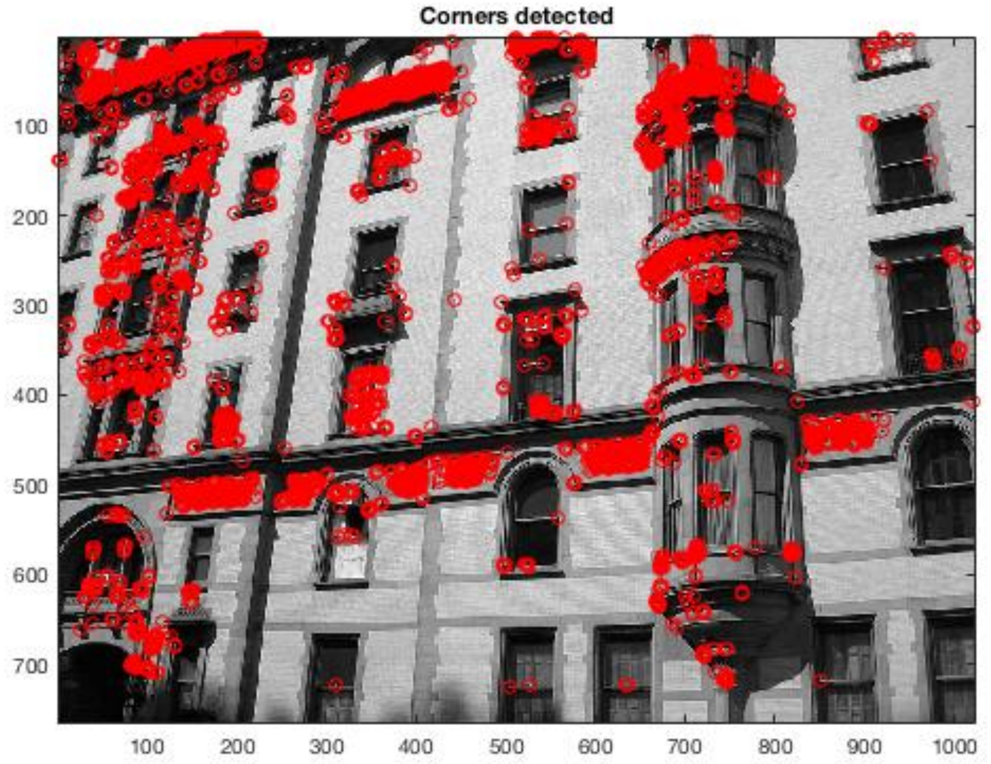
a)



```
img = double(rgb2gray(imread('building.jpg')));
figure; imagesc(img); axis image; colormap(gray);

[Ix,Iy] = imgradientxy(img);

g = fspecial('gaussian', 3, 0.6);

Ix2 = conv2(Ix.^2, g, 'same');
Iy2 = conv2(Iy.^2, g, 'same');
Ixy = conv2(Ix.*Iy, g, 'same');
R = (Ix2.*Iy2 - Ixy.^2)./(Ix2 + Iy2);

figure; imagesc(Ix2); axis image; colormap(gray);
figure; imagesc(Iy2); axis image; colormap(gray);
figure; imagesc(Ixy); axis image; colormap(gray);
%figure; imagesc(M); axis image; colormap(gray);
figure; imagesc(R); axis image; colormap(gray);
[r,c] = NonMaximaSup(R, 40, 0.2);
figure, imagesc(img), axis image, colormap(gray),hold on
plot(c, r, 'ro'), title('Corners detected');
```



b)



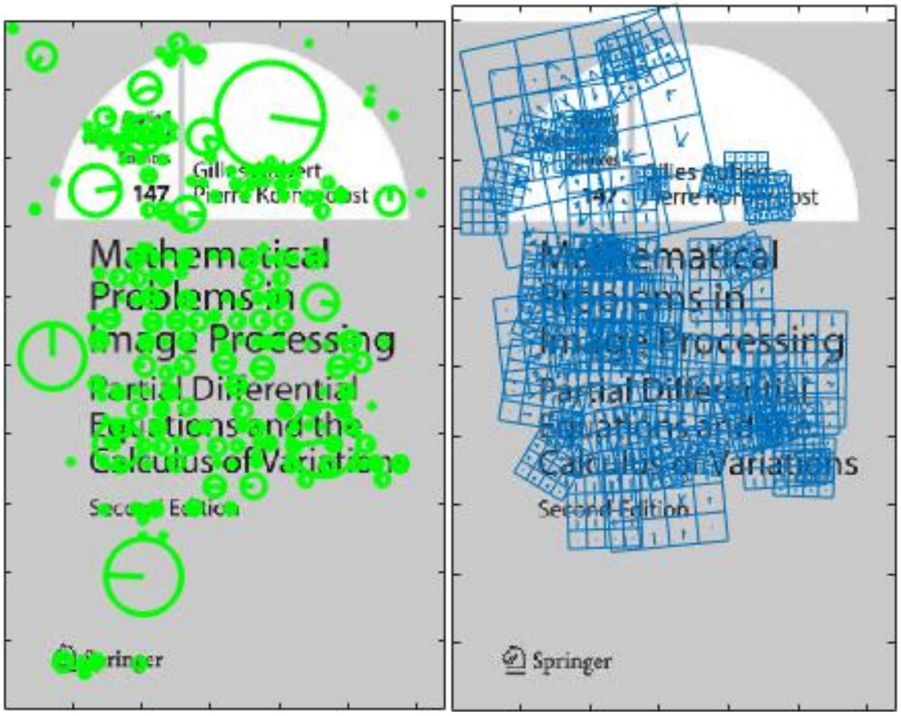
```
function [r,c] = NonMaximaSup(cornerness, radius, threshold)
% perform non-maximal suppression using ordfilt
    n = ordfilt2(cornerness, radius*2, strel('disk', radius).Neighborhood);

        % set threshold of the maximum value
    t = threshold*max(n(:));

        % find local maxima greater than threshold
    [r,c] = find(n>=t);
```

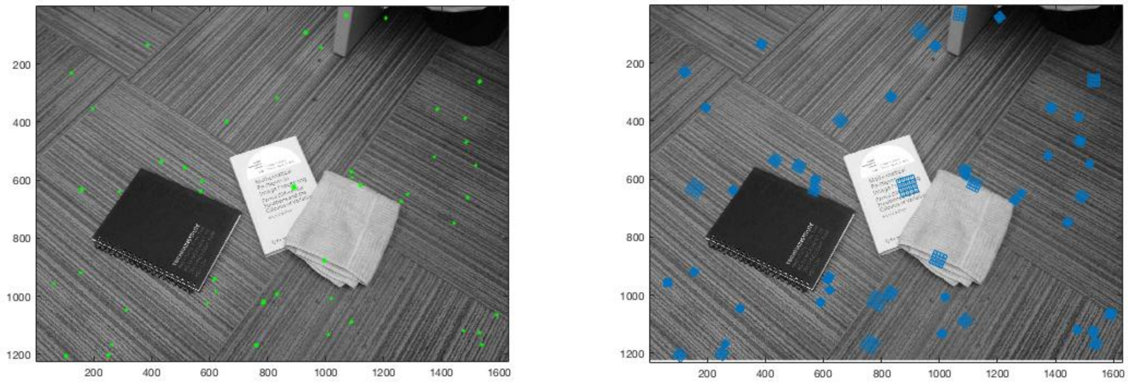
c)

```
%img = rand(imSize,imSize);
img = imread('synthetic.png');
img = double(img);
img = mean(img,3);

%%perform base-level smoothing to supress noise
imgS = img;%conv2(img,fspecial('Gaussian',[25 25],0.5),'same');%Base smoothing
cnt = 1;
clear responseDoG responseLoG
k = 1.1;
sigma = 2.0;
s = k.^(1:50)*sigma;
responseDoG = zeros(size(img,1),size(img,2),length(s));
responseLoG = zeros(size(img,1),size(img,2),length(s));
imG = zeros(size(img,1),size(img,2),length(s));

d = [1 1]';
%% Filter over a set of scales
for si = 1:length(s);
    sL = s(si);
    hs= max(25,min(floor(sL*3),300));
    HL = fspecial('log',[hs hs],sL);
    H = fspecial('Gaussian',[hs hs],sL);
    if(si<length(s))
        Hs = fspecial('Gaussian',[hs hs],s(si+1));
    else
        Hs = fspecial('Gaussian',[hs hs],sigma*k^(si+1));
    end
    imgFiltL = conv2(imgS,HL,'same');
    imgFilt = conv2(imgS,H,'same');
    imG(:,:,si) = imgFilt;
    imgFilt2 = conv2(imgS,Hs,'same');
    %Compute the DoG
    responseDoG(:,:,si)  = (imgFilt2-imgFilt);
    %Compute the LoG
    responseLoG(:,:,si)  = (sL^2)*imgFiltL;

    n = ordfilt2(responseLoG(:,:,si), si, strel('disk',
floor(si/2)).Neighborhood);
    t = 0.9*min(n(:));
    [r,c] = find(n<=t);
    rc = [r c];
    d = [d rc'];
end

% figure, imagesc(img), axis image, colormap(gray),hold on
% plot(d(1,:), d(2,:), 'ro'), title('Corners detected');
% hold off;

%Why do responseDoG and responseLoG look different for larger k?

%% Explore the scale space
t = 1;
DESC = 1;

for i = size(d(1,:))
    x = d(1,i);
    y = d(2,i);
```

2.

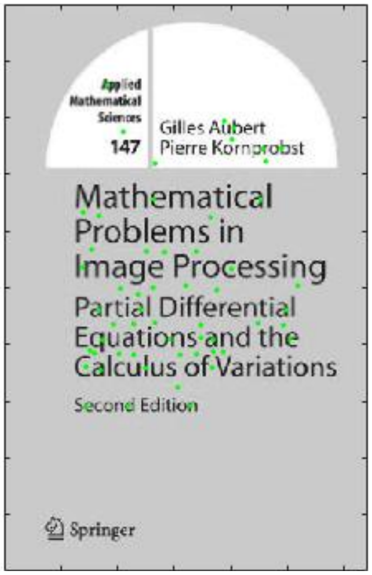
a)

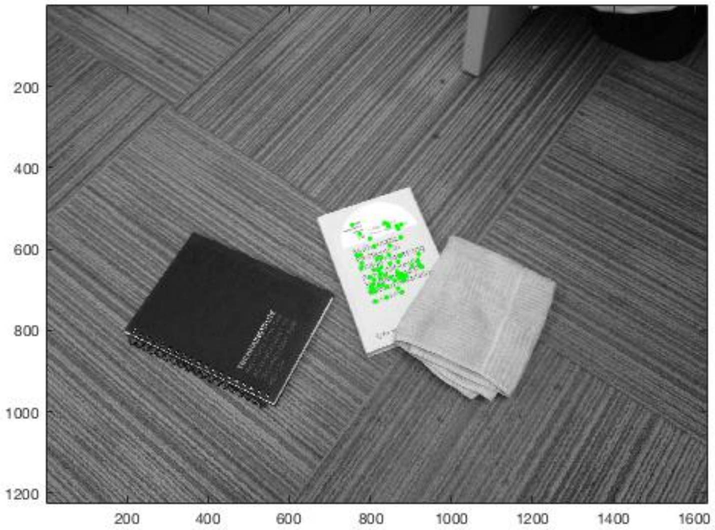




```
I = imread('book.jpg');
I = single(rgb2gray(I)) ;
Ib = imread('findbook.jpg');
Ib = single(rgb2gray(Ib)) ;
[f,d] = vl_sift(I) ;

[fb, db] = vl_sift(Ib) ;
```

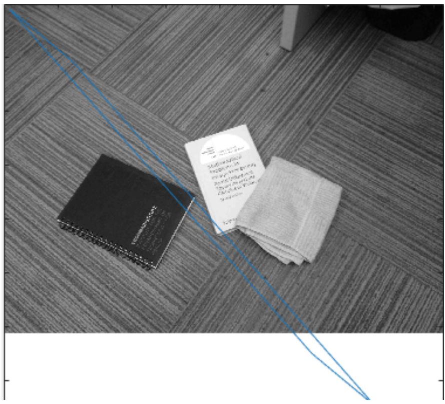
b)



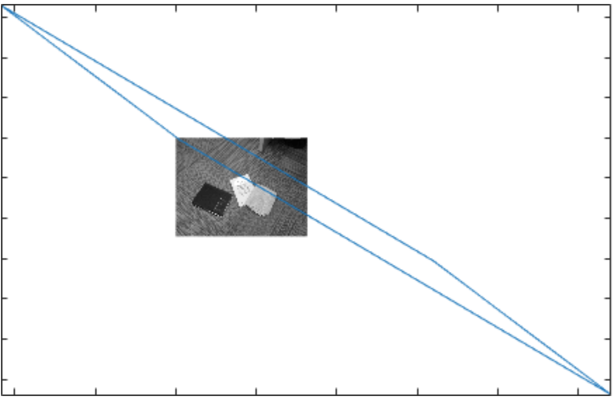


```
d = double(d);
db = double(db);
euc= pdist2(d', db', 'euclidean');
sorted = sort(euc, 2);
ratios=sorted(:,1)./sorted(:,2);

%copy_f(:,129)= ratios';
%matches = sortrows(copy_f', 129)';

threshold = 0.6;
matches = zeros(size(find(ratios<=threshold),1), 3);
for i = 1:size(euc,1)
    if ratios(i) < threshold
        matches(i,1) = ratios(i);
        matches(i,2)= i;
        matches(i,3)=find(euc(i,:)==sorted(i,1));
    end
end

matches( ~any(matches,2), : ) = [];
matching_points = zeros(size(matches,1), 5);
```

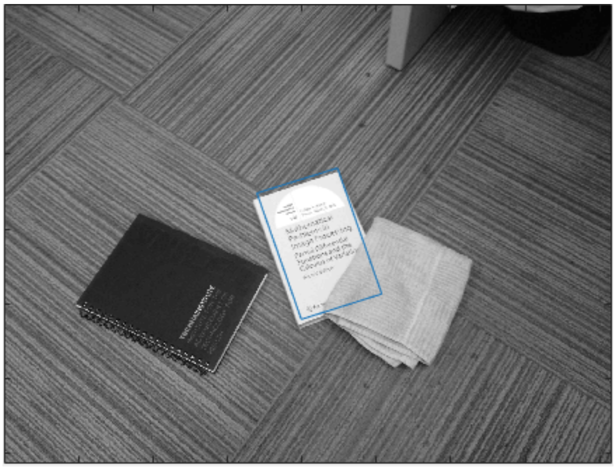
c) & d)



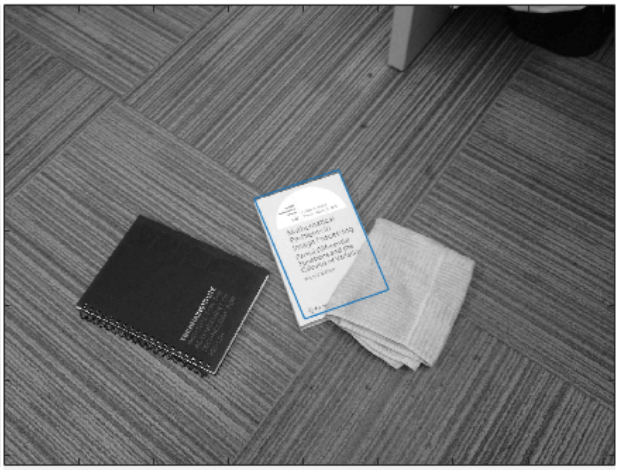
k = 1



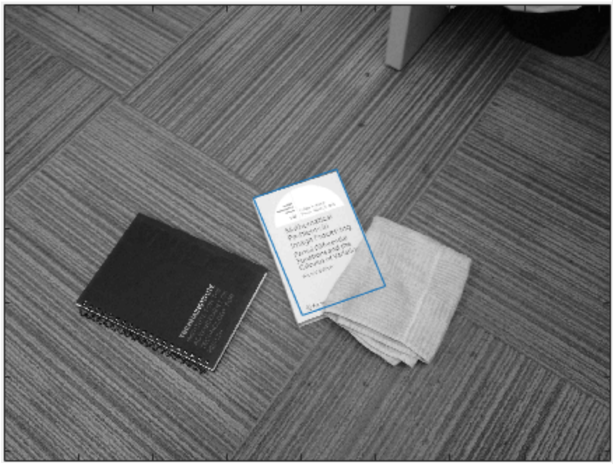
k = 2



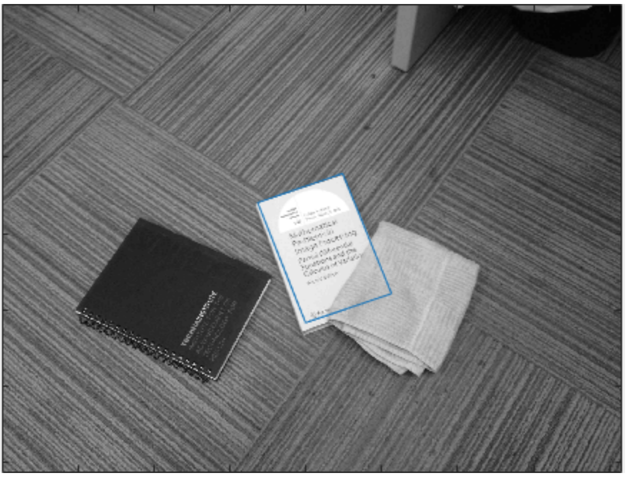
k = 4



k =12



k = 40



k = 75

```
for i = 1:size(matches,1)
    matching_points(i,1) = matches(i,1);
    matching_points(i,2:3) = [f(1,matches(i,2)) f(2,matches(i,2))];
    matching_points(i,4:5) = [fb(1,matches(i,3)) fb(2,matches(i,3))];

end

mmk = matching_points';
figure, imagesc(I), axis image, colormap(gray),hold on
plot(mmk(2,:),mmk(3,:),'g.') ;
hold off;
figure, imagesc(Ib), axis image, colormap(gray),hold on
plot(mmk(4,:),mmk(5,:),'g.') ;
hold off;

matching_pointss = sortrows(matching_points, 1);
k = 1;

fg = figure;imagesc(I);axis image;hold on;colormap gray;
drawnow;
[x,y] = ginput(4);

while (k)
    k = size(matching_pointss, 1);
    k = input(sprintf('Enter a value for k between 1 and %d\n',k));

    P = [];
    Pp = [];

    for i = 1:k
        x1 = matching_pointss(i,2);
        y1 = matching_pointss(i,3);
        x2 = matching_pointss(i,4);
        y2 = matching_pointss(i,5);
        P(size(P,1)+1,:) = [x1 y1 0 0 1 0];
        P(size(P,1)+1,:) = [0 0 x1 y1 0 1];

        Pp(size(Pp,1)+1,:) = x2;
        Pp(size(Pp,1)+1,:) = y2;
    end
    penny = (P'*P)'*inv((P'*P)*(P'*P)');
    affine = penny*P'*Pp;

    O = [];
    for i = 1:size(x)
        O(size(O,1)+1,:) = [x(i) y(i) 0 0 1 0];
        O(size(O,1)+1,:) = [0 0 x(i) y(i) 0 1];
    end

    transform = O*affine;
    yr = transform(2:2:length(transform));
    xr = transform(1:2:length(transform));
    xr= [xr' xr(1)];
    yr= [yr' yr(1)];

    figure, imagesc(Ib), axis image, colormap(gray),hold on
    plot(xr,yr);
    hold off;
end
```