Assignment#2　Object Detection

刘畅

This function reads two images, finds their sift features and displays lines connecting the matched keypoints.　A match is accepted only if its distance is less than distratio times the distance to the second closest match. It returns the number of matches displayed.

Step 1 Input image

The object image and scene image is given. Firstly, we should input the two images.

The code is as below:

```
%Object Detection
%变量名：object、scene 分别代表检测目标图像和场景图像；
function num = match(image1,image2)
tic;
clear;
clc;
tic
%%%%%%%读入图像并运用 sift 提取 keypoints
object='object.png';
scene='scene.png';
```

Step 2: Color space conversion & step 3: Features2D detection & Step 4: Calculate descriptors

We chose the RGB information to describe the image. Then we should converse the color space from RGB to GRAY.　This part will be implemented in the sub function called“sift”. “sift”is a function defined to detect the keypoints.

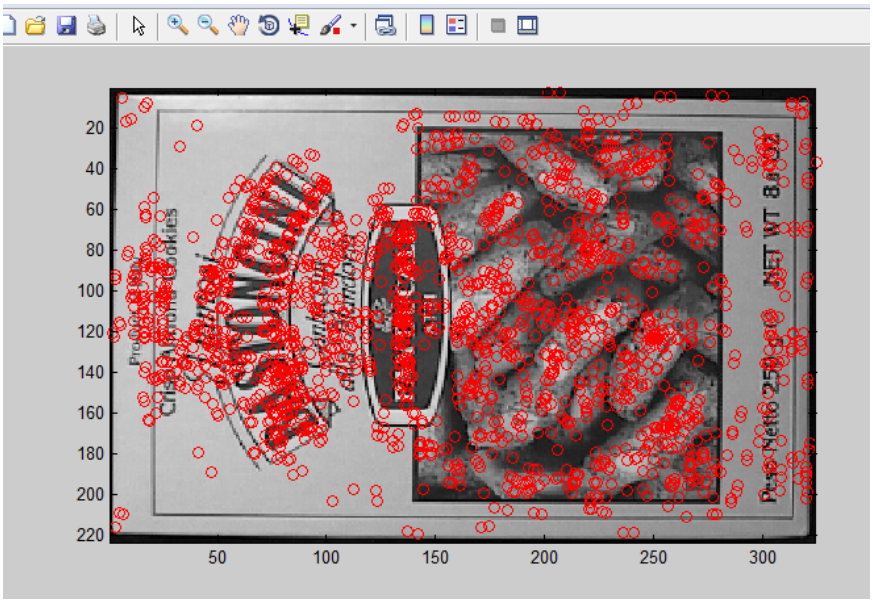
“sift” reads an image and returns its sift keypoints. “sift” contains the input part and the output part. Input parameters: the file name for the image. Returned: image(the image array in double format), descriptors(a K-by-128 matrix, where each row gives an invariant descriptor for one of the K keypoints. The descriptor is a vector of 128 values normalized to unit length.), locs(K-by-4 matrix, in which each row has the 4 values for a keypoint location (row, column, scale, orientation).The orientation is in the range [-PI, PI] radians.)

After the keypoints of object image and scene image are got, we store them in a matrix. The code in main function is as below.

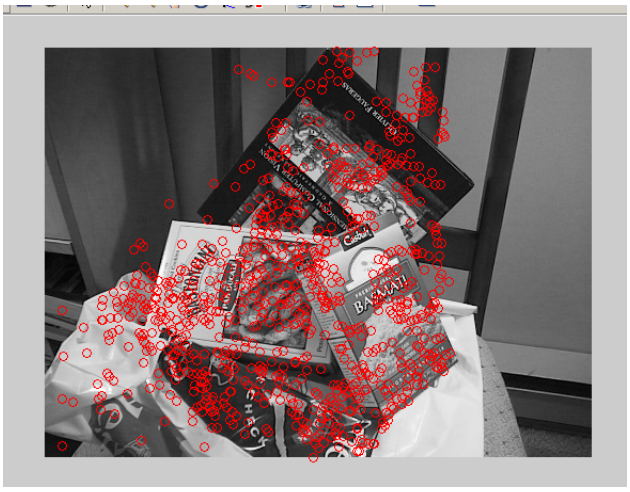
```
[im1, des1, loc1] = sift(object);
[im2, des2, loc2] = sift(scene);
 save('im1.mat','im1');
 save('im2.mat','im2');
```

Then, plot all the keypoints in object image.

```
%%显示检测出的特征点
 showkeys(object, loc1);
 showkeys(scene,loc2);
```



Feature2D Detection of the object image



Feature2D Detection of the object image

We need ‘showkeys’ as a sub function here.

```
%%%showkeys 函数
function showkeys(image, locs)
```

```
image=imread(image);
disp('Drawing SIFT keypoints ...');
% 画图
figure('Position', [50 50 size(image,2) size(image,1)]);
colormap('gray');
imagesc(image);
hold on;
imsize = size(image);
for i=1:size(locs,1)
    len=6*locs(i,3);
    s=sin(locs(i,4));
    c=cos(locs(i,4));
    r1=locs(i,1)-len*(c);
    c1=locs(i,2)+len*(-s);
    plot(c1,r1, 'ro');
end
hold off;
```

Besides,descriptors of the keypoints in the object image are stored in one matrix des1, and descriptors of the keypoints in the scene image are stored in another matrix des2.

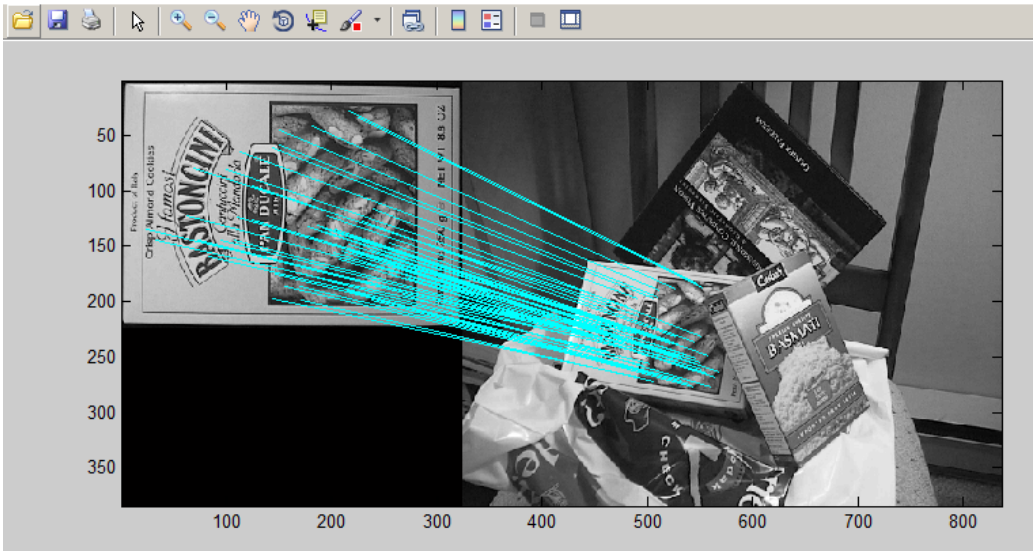
Step 5: Match descriptors & Step6 Find homography transformation

For efficiency in Matlab, it is cheaper to compute dot products between unit vectors rather than Euclidean distances. Note that the ratio of angles is a close approximation to the ratio of Euclidean distances for small angles. Only keep matches in which the ratio of vector angles from the nearest to second nearest neighbor is less than distRatio. Because these matches are regarded as good matches.

```
%%%寻找匹配的特征点
 %定义衡量两点距离的参数
 distRatio=0.5;
 %针对 object 图像中的每一个描述子点，在 scene 图像中寻找与之相匹配的点
 des2t = des2';                          % 预先计算矩阵的转置
 for i=1:size(des1,1)
   dotprods=des1(i,:)*des2t;        % 计算点积向量
   [vals,indx]=sort(acos(dotprods));
   if (vals(1)<distRatio*vals(2))
      match(i)=indx(1);
   else
      match(i)=0;
   end
end
```

After we have got the good matches in the two images, we create a new image showing the two images side by side. Select the image with the fewest rows and fill in enough empty rows to make it the same height as the other image. Then, show a figure with lines joining the accepted (good) matches.

```
% 将 object 与 scene 合并到一张图中用以显示最终的检测结果
% 较小的图像矩阵用 0 元素来填充，使之与较大的图像拥有一样的图像大小
rows1 = size(im1,1);
rows2 = size(im2,1);
if (rows1<rows2)
     im1(rows2,1)=0;
else
     im2(rows1,1)=0;
end
im3=[im1 im2];
%将检测出的相匹配的点用线连接起来
figure('Position', [100 100 size(im3,2) size(im3,1)]);
colormap('gray');
imagesc(im3);
hold on;
cols1 = size(im1,2);
for i = 1: size(des1,1)
  if (match(i)>0)
    line([loc1(i,2) loc2(match(i),2)+cols1],[loc1(i,1)
loc2(match(i),1)], 'Color', 'c');
  end
end
hold off;
num = sum(match > 0);
fprintf('Found %d matches.\n', num);
```



Step 7: Perspective transform & Step 8: Localize the object

We need to find four points on the edge of the scene image, so that we can draw a rectangle to mark the object which is detected through our project. Four points of scene image which are corresponding to the four corners. We use the matrix of homography transformation which was got in Step 6 to perspective transform the four corners of object image to four points of scene image.

```
%将检测出的物体用矩形框出
%矩阵 A 用来存储 scene 中检测出的匹配点
%矩阵 B 用来存储 object 中检测出的匹配点
double A=zero(size(des1,1));
double B=zero(size(des1,1));
j=1;
for i=1:size(des1,1)
  if (match(i)>0)
       A(j,:,:,:)=loc2(i,:,:,:);
       B(j,:,:,:)=loc1(i,:,:,:);
       j=j+1;
  end
end
rectangle('Position',[420,165,200,120],'EdgeColor','blue','LineWidth',
2);
hold off;
num = sum(match > 0);
fprintf('Found %d matches.\n', num);
```

After the four points are obtained, we draw lines between the four points to show the object area.