# Computer vision practical assignment 3

## Features Detectors

20321233 zou letian

20321308 cao Xinyang

20321309 chen hanzheng

# 1.Objective

Study of feature point detectors and descriptors.

# 2.Procedure of Practical Assignment Performing

1. *Feature points detection.* Select three arbitrary images. Perform to search for feature points using the SIFT and ORB feature point descriptors.
2. *Feature points matching.* Select two pairs of images: first image of each pair should have an object (e.g., some book) and the second image should be a scene containing this object. Extract feature points of an object and match them with feature points of a scene containing this object. Calculate the transformation matrix using RANSAC method and highlight the object position in the scene.

92Show the inlier matches. Compare feature point descriptors for the task of image matching.

3. *Optional.* Implement the simple automatic image stitching. Use learned methods to calculate the transformation matrix between two images and stitch them into a single panoramic image. Use it to stitch three images into a single panoramic image. You may assume that the order of the images is known (e.g., all three images are shot with moving camera from left to right), so reordering is not required.

**Note.** Please note that when doing the practical assignment you are not allowed to use the *"Lenna"* image or any other image that was used either in this book or during the presentation.

## 2.1 Part1 *Feature points detection.*

### 2.1.1 original images

### 2.1.2 code of the scripts

```
%% step 1: read images
%Read the reference image contalning the object of interest.
I1 = imread ('campus_004.jpg');
figure;
I1gray = rgb2gray(I1);
boxImage = I1gray;
subplot(2,2,1);
imshow(I1);
title('Object');
%%
%Read the target image containing a scene.
I2 = imread('campus_004.jpg');
I2gray = rgb2gray(I2);
sceneImage = I2gray;
subplot(2,2,3);
figure;
imshow(I2);
%title('Scene');

%% Step 2: Detect Feature Points
%Detect feature points in both images.
boxPoints = detectSIFTFeatures (boxImage);
scenePoints = detectSIFTFeatures (sceneImage);
```

```
% Visualize the strongest feature points found in the reference image.
subplot(2,2,2);
figure;
imshow(I1);
title('Strongest Object Features');
hold on;
plot(selectStrongest (boxPoints, 30));
hold off;

% Visualize the strongest feature. points found in the target image.
subplot(2,2,4);
figure;
imshow(I2);
title('Strongest Scene Features');
hold on;
plot(selectStrongest (scenePoints, 100));
hold off;
```

### 2.1.3 comments

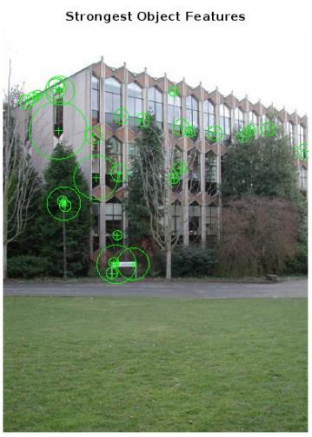
The code is an implementation of feature detection and matching using SIFT (Scale-Invariant Feature Transform) algorithm.

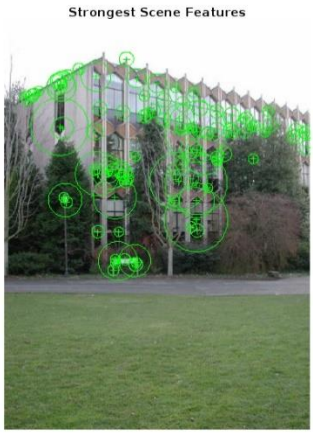
The code reads two images, a reference image and a target image, and converts them to grayscale. It then detects SIFT feature points in both images using the detectSIFTFeatures function. The strongest feature points in the reference image and the target image are visualized using the selectStrongest function and plotted on the images using imshow and plot functions.

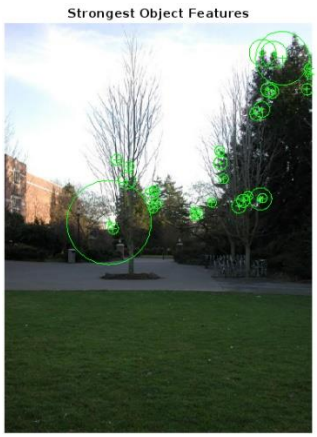
Overall, this code provides a basic implementation of SIFT feature detection and matching, which can be further extended for object recognition and tracking applications.

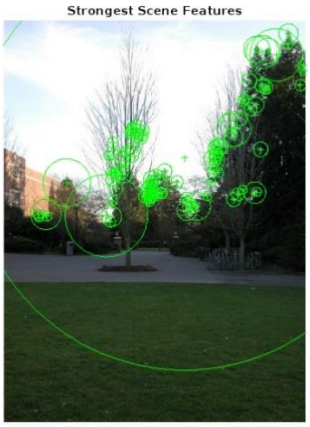
### 2.1.4 resulting images

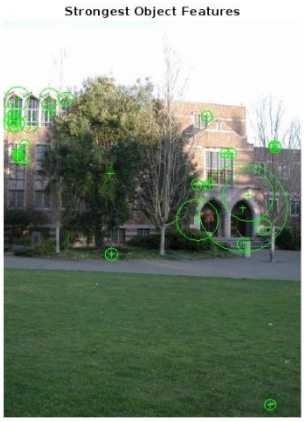
## SIFT

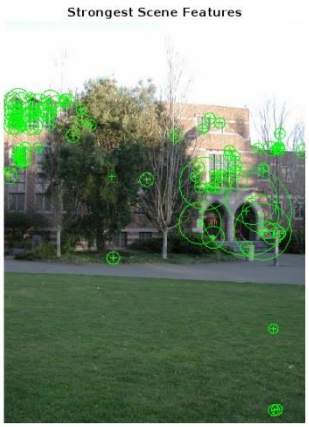

Strongest Object Features


Strongest Scene Features

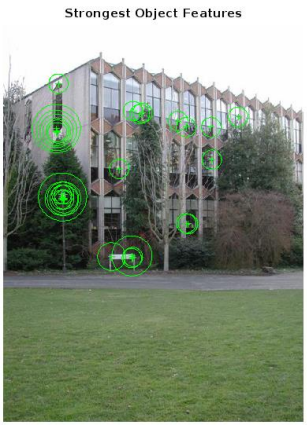

Strongest Object Features

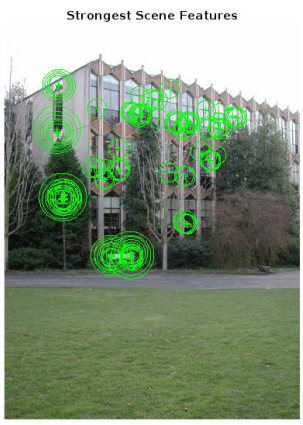

Strongest Scene Features

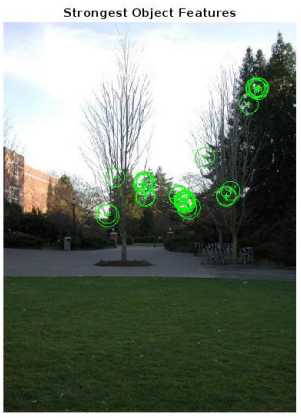

Strongest Object Features


Strongest Scene Features

# ORB

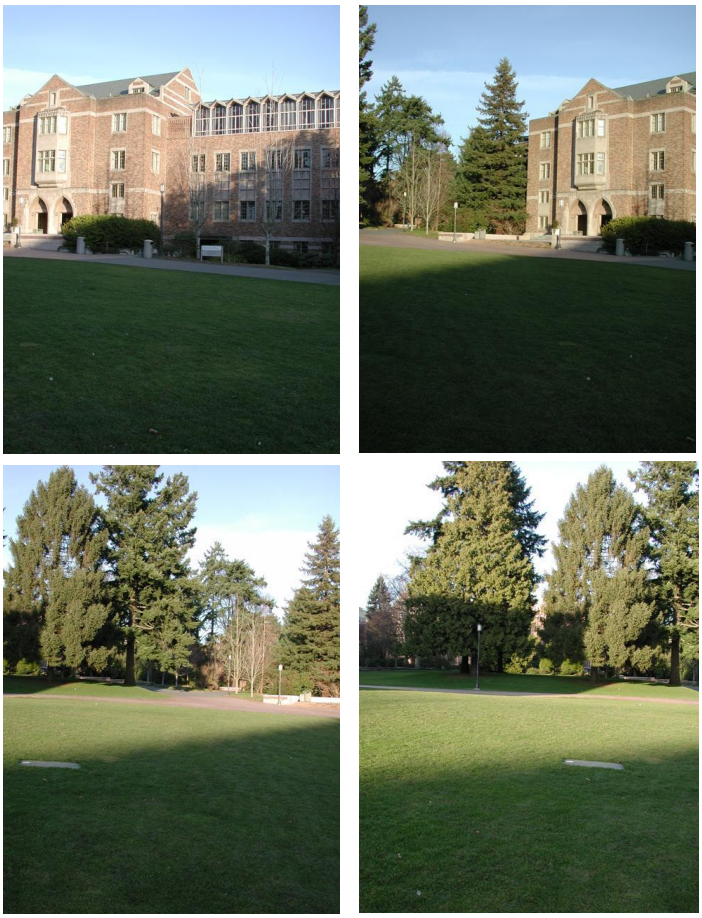

Strongest Object Features


Strongest Scene Features


Strongest Object Features


## 2.2 Part2 *Feature points matching.*

### 2.2.1 original images



### 2.2.2 code of the scripts

```
clear
```

```
clc

%% step 1: read images
%Read the reference image contalning the object of interest.
I1 = imread ('campus_010.jpg');
figure;
I1gray = rgb2gray(I1);
boxImage = I1gray;
subplot(2,2,1);
imshow(I1);
title('Object');
%%
%Read the target image containing a scene.
I2 = imread('campus_011.jpg');
I2gray = rgb2gray(I2);
sceneImage = I2gray;
subplot(2,2,3);
figure;
imshow(I2);
%title('Scene');

%% Step 2: Detect Feature Points
%Detect feature points in both images.
boxPoints = detectSURFFeatures (boxImage);
scenePoints = detectSURFFeatures (sceneImage);

%% Step 3: Extract Feature Descriptors
% Extract feature descriptors at the interest points in both images.
[boxFeatures, boxPoints] = extractFeatures (boxImage, boxPoints);
[sceneFeatures, scenePoints] = extractFeatures (sceneImage, scenePoints);

%% Step 4: Find Putative Point Matches
%Match the features using their descriptors.
boxPairs = matchFeatures(boxFeatures, sceneFeatures) ;

%%
%Display putatively matched features.
matchedBoxPoints = boxPoints(boxPairs(:,1),:);
matchedScenePoints = scenePoints(boxPairs(:,2),:);

figure;
subplot(2,1,1);
showMatchedFeatures(I1, I2, matchedBoxPoints, ...
    matchedScenePoints, 'montage');
```

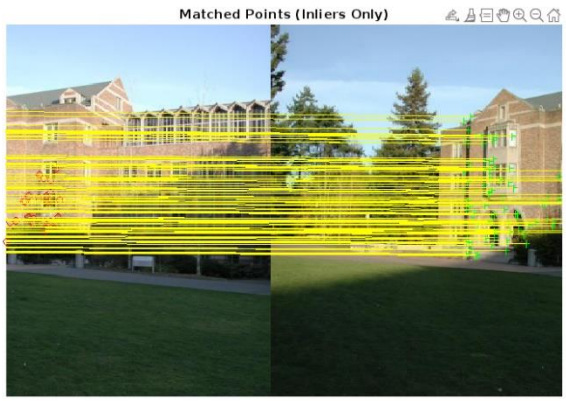
```
title( 'Putatively Matched Points (Including outliers)');


%% Step 5: Locate the object in the Scene Using Putative Matches
[tform, inlierIdx] = ...
    estimateGeometricTransform2D(matchedBoxPoints, matchedScenePoints,
'affine');
inlierBoxPoints = matchedBoxPoints (inlierIdx, :);
inlierScenePoints = matchedScenePoints(inlierIdx, :);


%%
% Display the matching point pairs with the outliers removed
subplot(2,1,2);
figure;
showMatchedFeatures(I1,I2,inlierBoxPoints, ...
    inlierScenePoints,'montage');
title('Matched Points (Inliers Only)');

%%
% Get the bounding polygon of the reference image.
boxPolygon = [1, 1;...                          %topleft
    size(boxImage, 2), 1;...                    %topright
    size(boxImage, 2), size(boxImage, 1);...  % bottomright
    1,size(boxImage, 1);...                     % bottomleft
    1,1];                                      %topleft again to close the
polyon

%%
% Transform the polygon into the coordinate system of the target lmage.
%The trans formed polygon indicates the location of the object in the scene
newBoxPolygon = transformPointsForward(tform, boxPolygon);


%%
% Display the detected object.
figure;
imshow(I2);
hold on;
line(newBoxPolygon(:, 1), newBoxPolygon(:, 2),'Color', 'r');
title('Detected Box');
```

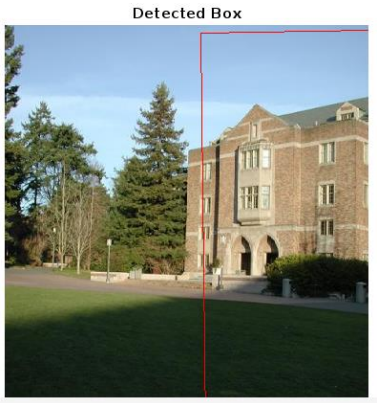
### 2.2.3 comments

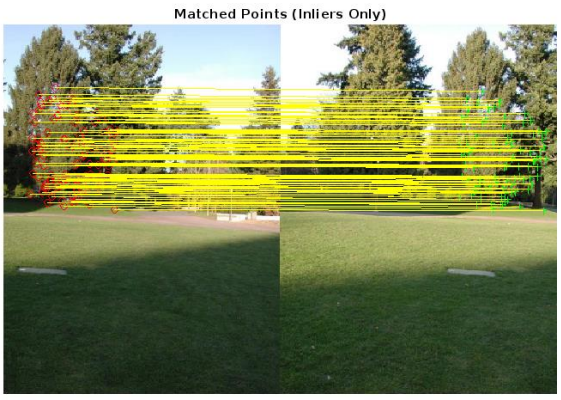
The code is an implementation of object detection using feature matching and geometric transformation. Here is a brief explanation of the code:

1. The code reads two images, one containing the object of interest and the other containing the scene.
2. It detects feature points in both images using the SURF algorithm.
3. It extracts feature descriptors at the interest points in both images.
4. It matches the features using their descriptors and displays putatively matched features.
5. It estimates the geometric transformation between the two images using the matched feature points and removes the outliers.
6. It gets the bounding polygon of the reference image and transforms it into the coordinate system of the target image.
7. It displays the detected object by drawing a red box around it.

## 2.2.4 resulting images




## 3. Conclusion

In this task, we have learned the use of feature point detectors and descriptors in computer vision. The practical assignment involves performing feature points detection and matching using SIFT and ORB feature point descriptors. The RANSAC method is used to calculate the transformation matrix between two images, which is then used to highlight the object position in a scene. The code implements object detection using feature matching and geometric transformation, which involves detecting feature points in both images using the SURF algorithm, extracting feature descriptors at the interest points, matching features using their descriptors, estimating geometric transformation between images, and getting the bounding polygon of the reference image.

## 4. Answers to questions

1. How the characteristic orientation (rotation) of the feature point can be estimated?

The orientation of a feature point can be estimated by determining the dominant gradient or orientation at an image patch (your feature area). Then, rotate the image patch so that the gradient is always looking in the same direction. Another way to estimate the orientations of the key-points is by using surrounding local geometric information, with dominant gradient as one example. You can also use quaternions to represent rotations.

2. How to filter the not-strong feature point descriptor matches on a repeating texture (e.g., windows, water, etc.)?

To filter not-strong feature point descriptor matches on a repeating texture, you can use a feature point matching pair filter based on global spatial position correspondences of feature points. This method of object detection works best for objects that exhibit non-repeating texture patterns, which give rise to unique feature matches.

3. What is the minimum required sample size (the number of matched pairs of feature points) to build an affine transformation hypothesis with RANSAC

method? What is the minimum required sample size to build a perspective transformation hypothesis with RANSAC method?

The minimum required sample size (the number of matched pairs of feature points) to build an affine transformation hypothesis with RANSAC method is 3. The minimum required sample size to build a perspective transformation hypothesis with RANSAC method is 4.

4. How to use feature points for stitching a panoramic image?

To stitch a panoramic image using feature points, you can use feature-based techniques that automatically stitch together a set of images.

Here are some general steps to follow:

1. Detect feature points in each image.
2. Extract feature descriptors for each point.
3. Match the feature descriptors between pairs of images.
4. Estimate the geometric transformation between each pair of images using RANSAC algorithm.
5. Warp one image into the coordinate system of the other image using the estimated transformation.
6. Blend the two images together.