CV2015Spring—Assignment #2

Due: Thursday, Apr 30 10:00 AM

Zhao Haiwei

Apr 16, 2015

1. Assignment requirement

For this assignment, you will implement a version of the Object Detection technique. See Figure 1 for an example.

(a) (b) (c)

Figure 1: Object detection.

Your method must be Featuers2D-based, and at least one kind of Feature2D[1] and matching strategy should be used. I will give you some tips for the implementation in the following sections.

2. Tips

The whole framework of the implementation for object detection is shown in Figure 2, it may serve as a reference for your assignment. Next, I will introduce the implementation and requirement of each part of the framework for you.

2.1 Step 1: Input image

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

[1] Please refer to http://docs.opencv.org/modules/features2d/doc/features2d.html.

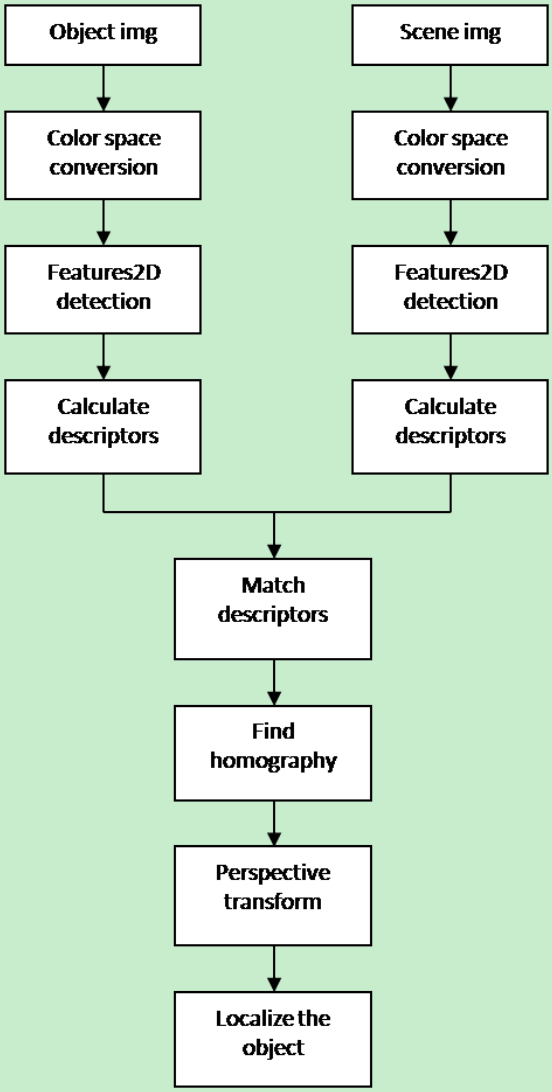


Figure 2: Framework of the implementation for object detection.

2.2 Step 2: Color space conversion (5 points)

Input The input color image ($m \times n \times 3$ matrix), m is the width of the input image, n is the height of the input image.

Output $m \times n$ matrix.

Implementation This step determines the kind of color space you want to use in step 2, and you should converse the color space from RGB to GRAY.

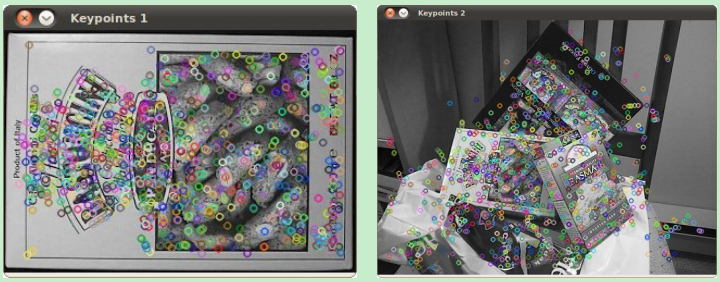
2.3 Step 3: Features2D detection (15 points)

Input The gray image ($m \times n \times 1$ matrix), m is the width of the input image, n is the length of the input image.

Output Keypoints of object image and scene image.

Implementation Detect the keypoints using Features2D Detector and store the result.

Hint There are lots of Features2D detection algorithms, such as SIFT, SURF, and so on. You can choose one of them which is fast and easy to implement.



(a) (b)

Figure 3: Features2D detection.

2.4 Step 4: Calculate descriptors (15 points)

Input The gray image ($m \times n \times 1$ matrix), m is the width of the input image, n is the length of the input image.The keypoints detected in Step 3.

Output One matrix describes the keypoints in the object image and one matrix describes the keypoints in the scene image.

Implementation Descriptors of the keypoints in the object image are stored in one Matrix and descriptors of the keypoints in the scene image are stored in another Matrix.

2.5 Step 5: Match descriptors (20 points)

Input One matrix describes the keypoints in the object image and one matrix describes the keypoints in the scene image.

Output M good matched keypoints between object image and scene image.

Implementation

- Step 5-1: You should quick calculate $x \times y$ (x, y are the number of keypoints) distances (value) between keypoints, each value indicates the similarity of one keypoint from object image and one keypoint from scene image.
- Step 5-2: Fix one keypoint from object image and select the min or max distance between the keypoint and one keypoint from scene image to find the best match,then change the fixed keypoint and do the same.
- Step 5-3: You should draw only "good" matches (i.e. whose distance is less than $3 * \min(distance)$).

Hint There are lots of distance calculation algorithms, such as Hausdorff Distance, Euclidean Distance, and so on. You can choose one of them to calculate the similarity between two keypoints and find the best match.

2.6 Step 6: Find homography transformation (20 points)

Input Keypoints of object image and scene image.

Output Matrix of homography transformation between two different point sets.

Implementation

- Step 6-1: Get N keypoints from the good matches and store them in two different point sets.
- Step 6-2: Calculate the matrix of homography transformation between the two different point sets.

Hint There are lots of mapping transformation algorithms, I recommend you to use homography transformation.

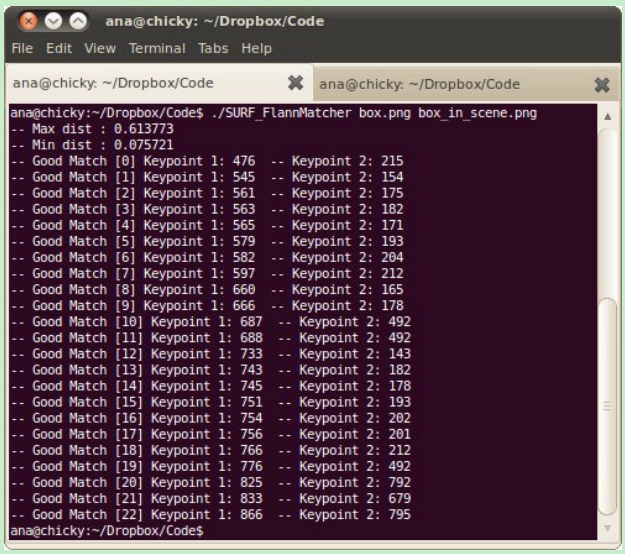
2.7 Step 7: Perspective transform (20 points)

Input Four corners of object image.

Output Four points of scene image which are corresponding to the four corners.

Implementation You can 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.

(a)



(b)

Figure 4: Match descriptors.

2.8 Step 8: Localize the object (5 points)

Input The scene image which has four points (the mapped object in the scene image).

Output The matrix box in the scene image which contains the object in object image.

Implementation Draw lines between the four points (the mapped object in the scene image).

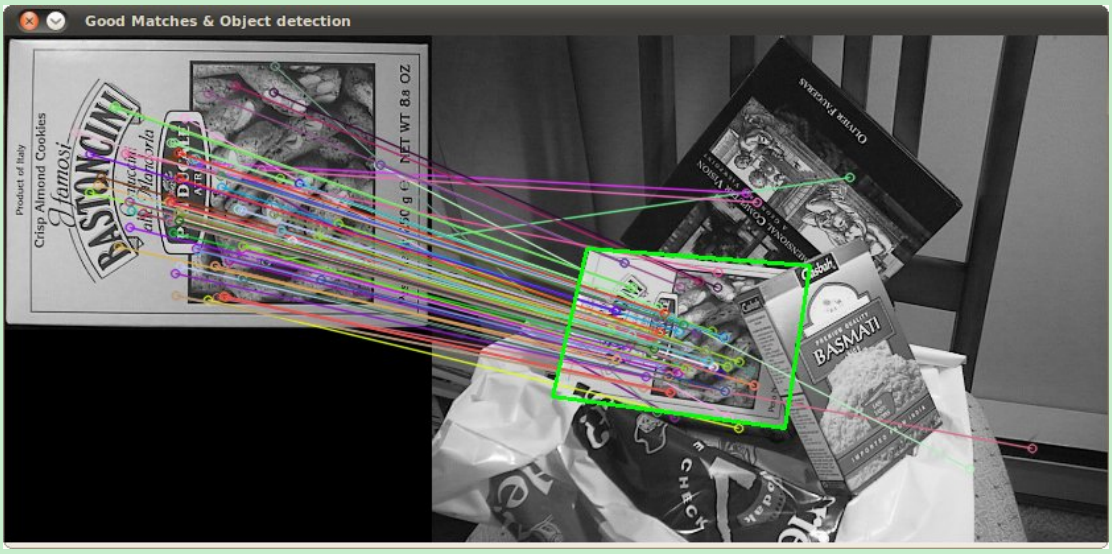


Figure 5: Localize the object.

3. Submission instructions

3.1 What to hand in?

- Your matlab code or C/C++ code (show the result of each step in your code)
- A "Readme" file to illustrate how to run your code
- A report containing the following items:
  - The title and your name at the top
  - A brief explanation of your implementation strategy corresponding to each step (in English)

3.2 Where to hand in?

Submit to Piazza in form of a followup below my assignment note.