

Image-Crop Association

Problem Statement: The problem is to identify the association between an image and its appropriate cropped image along with the co-ordinate points.

Type of Learning: Un-Supervised Learning.

Language Chosen: Python

Libraries Used:

1. Open-cv
2. Scikit learn
3. numpy

Algorithms Used:

1. Opencv- MatchTemplate
2. Opencv – xfeatures2D:shift
3. Opencv - FlannBasedMatcher
4. Sklearn.cluster – KMeans

Data Analysis:

Given, list of images and list of cropped images.

On observing the list of images, I could classify the distinct images as

- An image of girl near a building and so.
- An image of plain color (For eg., a simple blue colored image).
- An image of combination of colors (For eg., a combination of blue and green).

On observing the list of cropped images, I could classify the distinct images as

- A cropped image, which has a bag and so.
- A cropped image, which is rotated at some angle.
- A cropped image with plain color.
- A cropped image with a combination of plain colors.

Key-Points and Feature Extraction:

As the source images are identified to be in 3 types, the key-points and the descriptors are extracted using **open-cv's xfeatures2d-Shift algorithm**. After the key-points and descriptors extraction, for the single colored images, the key-points and descriptors would be empty; however, for the other images, key-points and descriptors will be used to match the cropped image to the source image.

Identifying the Association:

For each image in the source images and cropped image, will be extracting the key-points and descriptors using open-cv xfeatures2d-shift algorithm.

Scenario#1:

If the key-points and descriptors of the source and cropped images are not empty, **matchTemplate**(a simple crop) will be applied with cropped image and source image to identify the suitable match. If suitable match is available, co-ordinates can be calculated accordingly. If the crop (considering the crop is rotated at some angle) is not matching to the source image, we will be applying the “**FlannBasedMatcher**” to extract the matching points between the crop and the source image. We will be calculating the good points with a ratio index. If the good points meets our similarity threshold, then crop will be considered as a match to the source image.

Scenario#2:

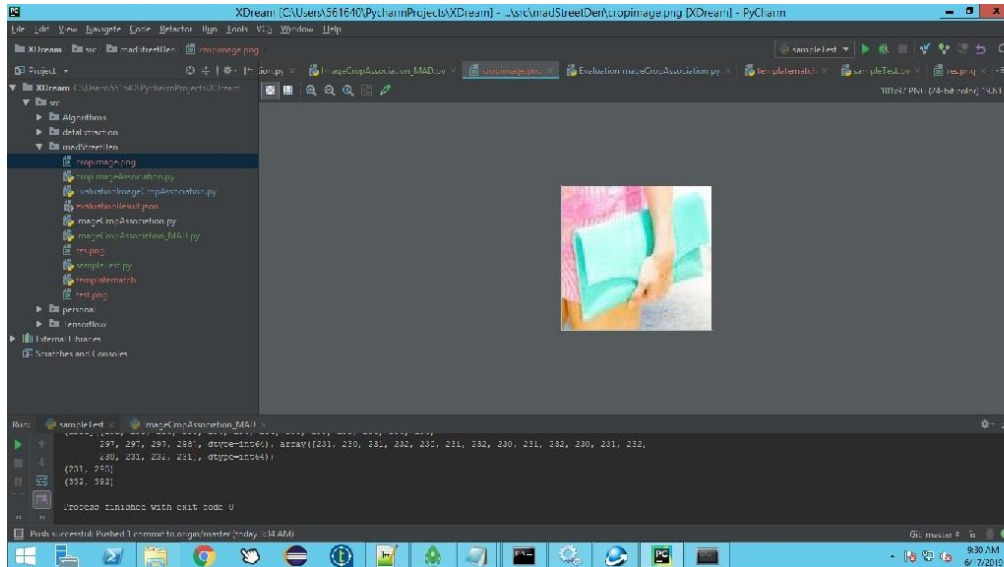
If the key-points and descriptors are empty (assuming that the image is with plain color), KMeans algorithm (number of cluster = 2) will be used to extract the predominant colors of the images from the cluster centers. For the extracted colors(rgb code) of source image, if the key-points and descriptors of the cropped are empty, KMeans algorithm (number of cluster = 3) will be applied to extract the predominant colors of the images from cluster centers. If the color code matches, then the suitable crop is identified to the source image.

Once the association between the images and the crops are identified, the result will be written to a json file.

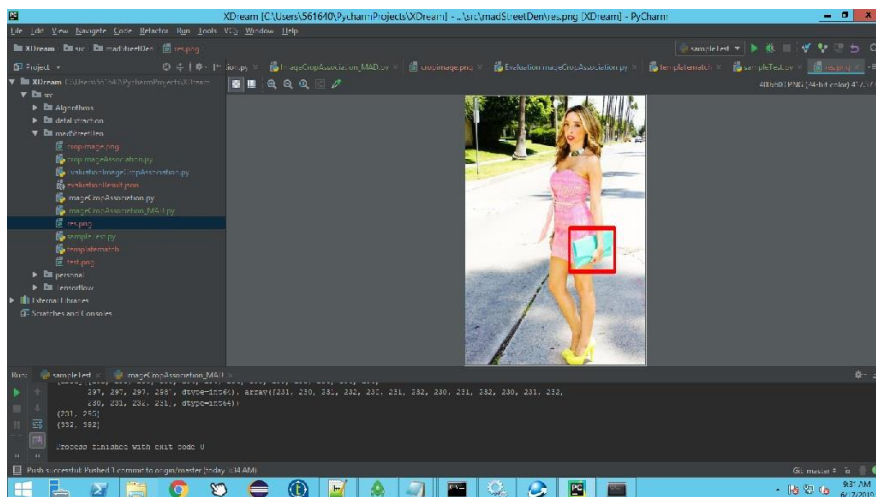
Data Structures Chosen:

1. List – To add the crop names and its co-ordinates
2. Set – To add the unique crop names, which has the associated images.

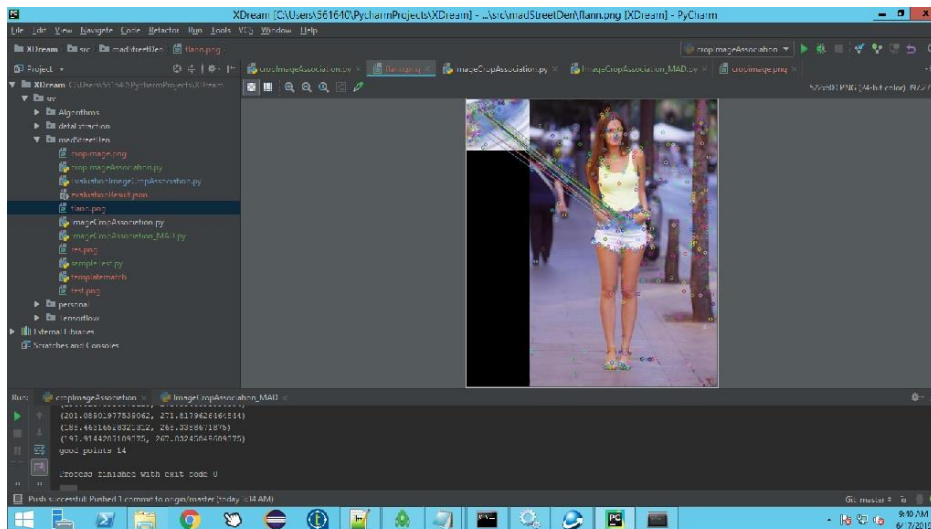
Cropped Image:



Source Image:



Visualizations by FlannBasedMatcher:



Evaluation Metrics:

The metrics chosen for the evaluations are against the “out_res.json”,

1. True Positive:

For a particular source image, if the actual crop is ‘x’ and predicted crop is also ‘x’, then this is said to be true positive count.

2. False Positive:

For a particular source image, the predicted crop is ‘x’ and it has no ‘x’ in the actual crop, then it is said to be false positive.

3. False Negative:

For a particular source image, if the actual crop is ‘x’, it has no ‘x’ in the predicted crop, then it is said to be false negative.

Example from Sample Test:

Source image: d469bcc8-d501-5b75-81b0-00eabb7a2238

Actual crops:

- d5677b81-0129-5c5d-be64-d888625c92bb
- 78a4e540-9e91-5ec1-9b6f-97be05788f1a
- 0db89e0b-5860-5025-a314-4ebc6159ca2b

Predicted crops:

- 0db89e0b-5860-5025-a314-4ebc6159ca2b
- 3269df9a-7ac9-4f84-9b5e-d1340bd00bf5
- bfab47d0-65d4-46f4-9863-04b5c2118121
- d5677b81-0129-5c5d-be64-d888625c92bb

Observed Metrics:

d469bcc8-d501-5b75-81b0-00eabb7a2238: {

```
    "TP": 2,  
    "FN": 1,  
    "FP": 2,  
    "precision": 0.5,  
    "recall": 0.67  
},
```

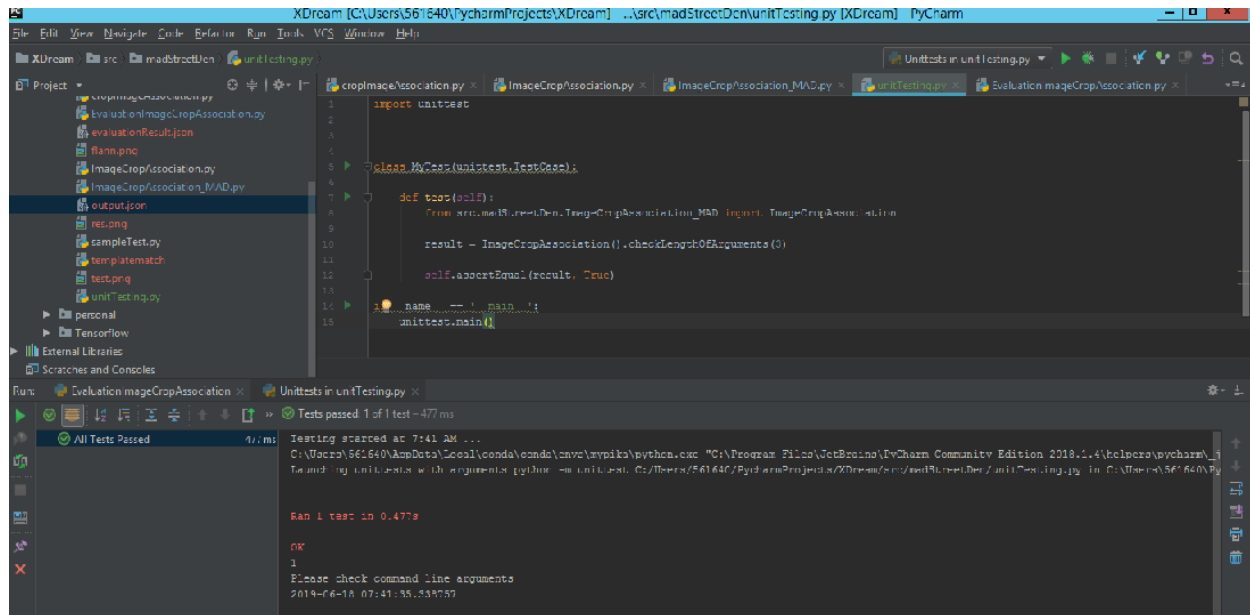
Unit Testing:

Testcase: 1

Checking for the number of command line Arguments available.

Positive Scenario :

- Evaluating whether the length of the arguments meets the criteria



The screenshot displays the PyCharm IDE interface. The top pane shows the source code of `unittesting.py`, which includes an import for `unittest`, a class `MyClass(unittest.TestCase)`, a test method `def test(self):` that creates an `ImageCropAssociation` object and checks its `arguments` attribute, and a `__name__ == '__main__':` block calling `unittest.main()`. The bottom pane shows the test results for `unittesting.py`. It indicates that all tests passed, with a total of 1 test in 477 ms. The console output shows the test execution details, including the test name `test` and the message `Please check command line arguments`.

```
import unittest

class MyClass(unittest.TestCase):

    def test(self):
        from src.main import ImageCropAssociation
        result = ImageCropAssociation().checklengthOfArguments(0)
        self.assertEqual(result, True)

if __name__ == '__main__':
    unittest.main()
```

Run: Evaluation imageCropAssociation x unittesting.py x
Tests passed: 1 of 1 test - 477 ms
All Tests Passed 477 ms
Testing started at 7:41 AM ...
C:\Users\561640\AppData\Local\conda\conda\envs\myproj\python.exe "C:\Program Files\JetBrains\PyCharm Community Edition 2018.1.4\helpers\pycharm..."
Running unittest with arguments: python -m unittest C:\Users\561640\PycharmProjects\XDream\src\main\Dev\unittesting.py in C:\Users\561640\PycharmProjects\XDream\src\main\Dev\unittesting.py
Run 1 test in 0.477s
OK
1
Please check command line arguments
2019-06-18 07:41:55.558750

Negative Scenario :

- Tweated the logic in the actual source code and checked for the scenario to be failed.

