

CS5542 Big Data Analytics and App

Lab Assignment #1

Submitted by:

Koushik Katakam – 10

Team – 5

Goals:

The goals of the Lab Assignment 1:

1. Based on the Project theme, download the dataset.
2. Perform basic NLP operations namely Tokenization and Lemmatization on the downloaded captions.
3. From the extracted data, report the image statistics.
4. Perform feature extraction using SIFT algorithm on the image data.

Technologies Used:

Pycharm

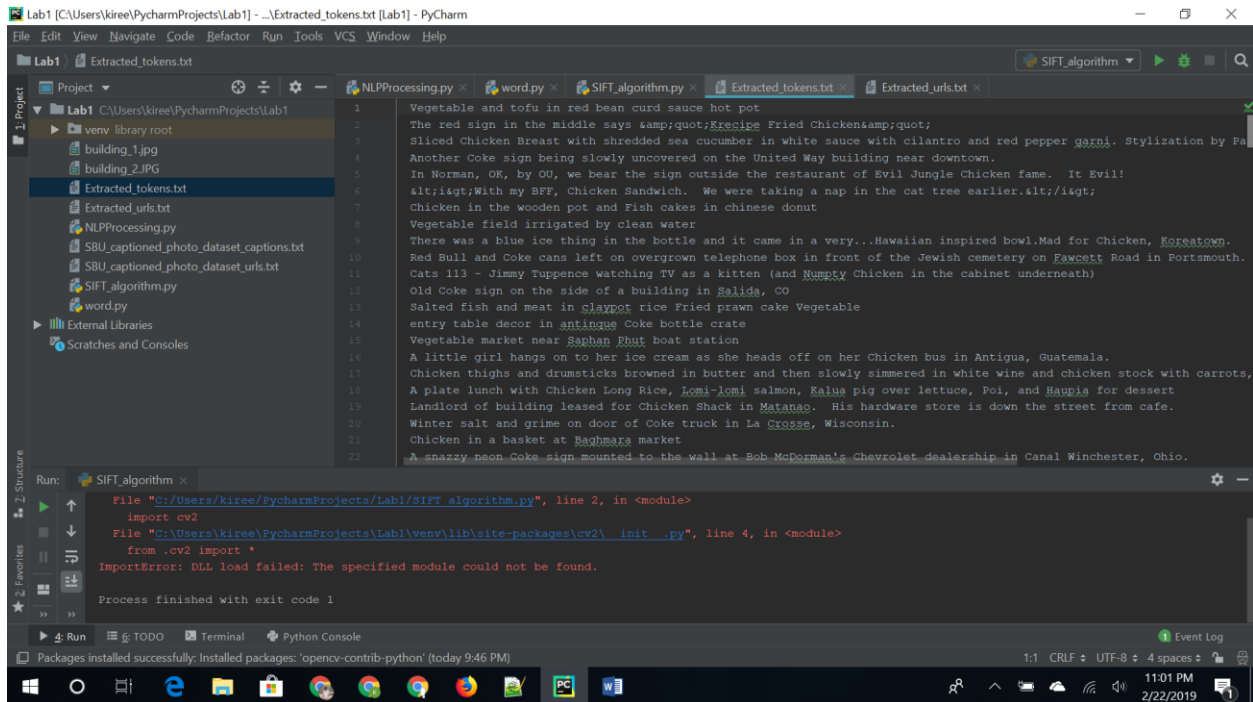
Packages Used:

- nltk
- opencv-python
- numpy
- matplotlib
- tensorflow
- linecache

Dataset:

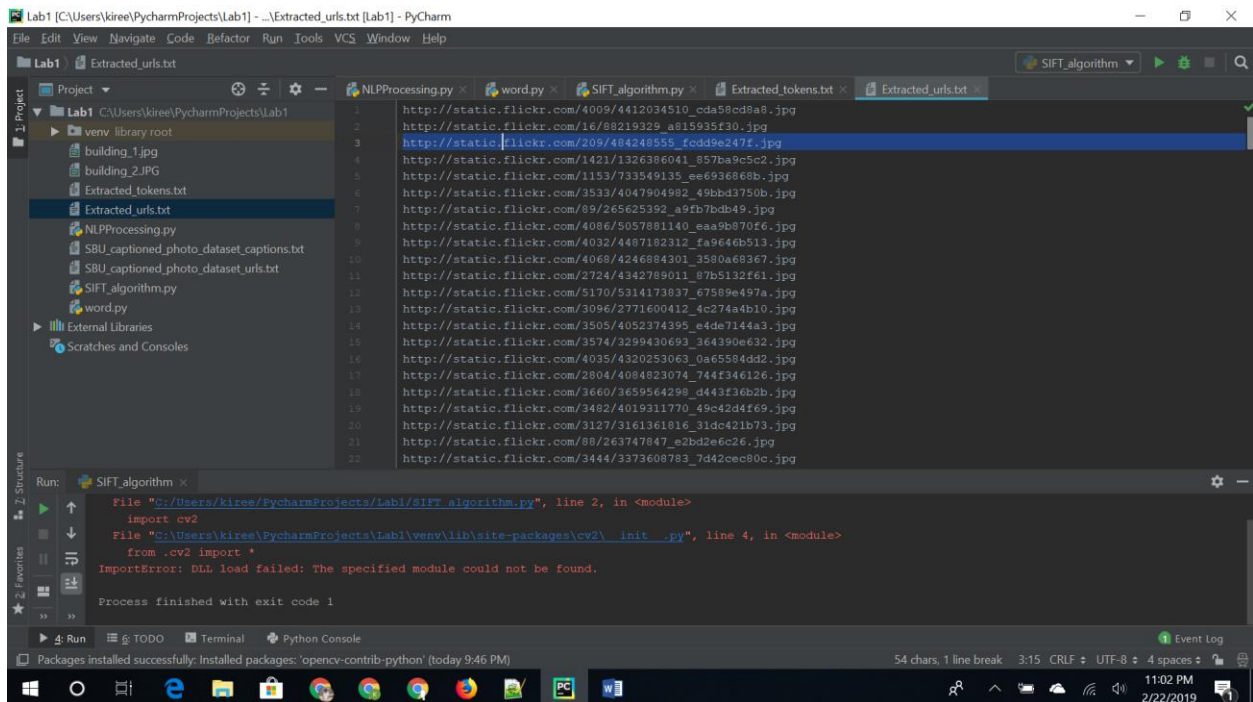
There are many datasets available online. From those, we have chosen SBU dataset in which the data is in the form of “URL’s” and “Captions”.

Screenshots of Dataset URL's and corresponding Captions:



This screenshot shows the PyCharm IDE with a project named 'Lab1'. The left sidebar displays the project structure, including files like 'building_1.jpg', 'building_2.JPG', 'Extracted_tokens.txt', 'Extracted_urls.txt', 'NLPProcessing.py', 'SBU_captioned_photo_dataset_captions.txt', 'SBU_captioned_photo_dataset_urls.txt', 'SIFT_algorithm.py', and 'word.py'. The main editor window shows the 'Extracted_urls.txt' file, which contains a list of URLs. The 'SIFT_algorithm.py' file is also visible, showing a script that imports 'cv2' and attempts to load a DLL, resulting in an 'ImportError: DLL load failed: The specified module could not be found.' error. The bottom status bar indicates that packages were installed successfully: 'opencv-contrib-python' (today 9:46 PM).

```
1 Vegetable and tofu in red bean curd sauce hot pot
2 The red sign in the middle says &quot;Recipe Fried Chicken&quot;
3 Sliced Chicken Breast with shredded sea cucumber in white sauce with cilantro and red pepper garni. Stylization by Pa
4 Another Coke sign being slowly uncovered on the United Way building near downtown.
5 In Norman, OK, by OU, we bear the sign outside the restaurant of Evil Jungle Chicken fame. It Evil!
6 <img alt="With my BFF, Chicken Sandwich. We were taking a nap in the cat tree earlier.</img>
7 Chicken in the wooden pot and Fish cakes in chinese donut
8 Vegetable field irrigated by clean water
9 There was a blue ice thing in the bottle and it came in a very...Hawaiian inspired bowl.Mad for Chicken, Koreatown.
10 Red Bull and Coke cans left on overgrown telephone box in front of the Jewish cemetery on Fawcett Road in Portsmouth.
11 Cats 113 - Jimmy Tuppence watching TV as a kitten (and Numpy Chicken in the cabinet underneath)
12 Old Coke sign on the side of a building in Salida, CO
13 Salted fish and meat in claypot rice Fried prawn cake Vegetable
14 entry table decor in antique Coke bottle crate
15 Vegetable market near Baghmati boat station
16 A little girl hangs on to her ice cream as she heads off on her Chicken bus in Antigua, Guatemala.
17 Chicken thighs and drumsticks browned in butter and then slowly simmered in white wine and chicken stock with carrots.
18 A plate lunch with Chicken Long Rice, Komi-komi salmon, Kalua pig over lettuce, Poi, and Haupia for dessert
19 Landlord of building leased for Chicken Shack in Matanago. His hardware store is down the street from cafe.
20 Winter salt and grime on door of Coke truck in La Crosse, Wisconsin.
21 Chicken in a basket at Baghmati market
22 A snazzy neon Coke sign mounted to the wall at Bob McDorman's Chevrolet dealership in Canal Winchester, Ohio.
```

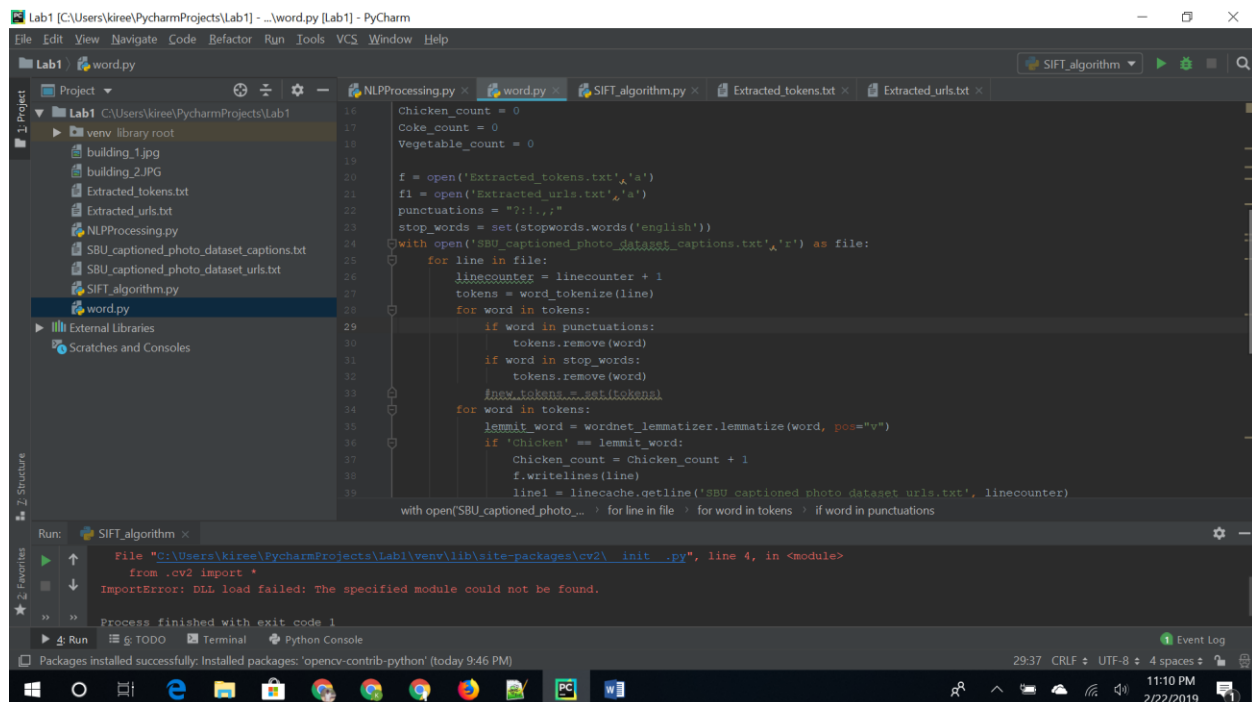


This screenshot shows the PyCharm IDE with a project named 'Lab1'. The left sidebar displays the project structure, including files like 'building_1.jpg', 'building_2.JPG', 'Extracted_tokens.txt', 'Extracted_urls.txt', 'NLPProcessing.py', 'SBU_captioned_photo_dataset_captions.txt', 'SBU_captioned_photo_dataset_urls.txt', 'SIFT_algorithm.py', and 'word.py'. The main editor window shows the 'Extracted_urls.txt' file, which contains a list of URLs. The 'SIFT_algorithm.py' file is also visible, showing a script that imports 'cv2' and attempts to load a DLL, resulting in an 'ImportError: DLL load failed: The specified module could not be found.' error. The bottom status bar indicates that packages were installed successfully: 'opencv-contrib-python' (today 9:46 PM).

```
1 http://static.flickr.com/4009/4412034510_ea58cd8a0.jpg
2 http://static.flickr.com/16/88219329_a815935f30.jpg
3 http://static.flickr.com/209/484248555_fcd9e247f.jpg
4 http://static.flickr.com/1421/1326386041_857ba9c5c2.jpg
5 http://static.flickr.com/1153/733549135_e6936868b.jpg
6 http://static.flickr.com/3533/4047904982_49bbd3750b.jpg
7 http://static.flickr.com/89/265625392_a9fb7bdb49.jpg
8 http://static.flickr.com/4086/5057881140_eaa9b870f6.jpg
9 http://static.flickr.com/4032/4487182312_fa9646b513.jpg
10 http://static.flickr.com/4068/4246884301_3580a68367.jpg
11 http://static.flickr.com/2724/4342789011_87b5132f61.jpg
12 http://static.flickr.com/5170/5314173837_67589e497a.jpg
13 http://static.flickr.com/3096/2771600412_4c274a4b10.jpg
14 http://static.flickr.com/3505/4052374395_e4de7144a3.jpg
15 http://static.flickr.com/3574/3299430693_364390e632.jpg
16 http://static.flickr.com/4035/4320253063_0a65584d2.jpg
17 http://static.flickr.com/2804/4084823074_744f346126.jpg
18 http://static.flickr.com/3660/3659564298_d443f36b2b.jpg
19 http://static.flickr.com/3482/4019311770_49c42d4f69.jpg
20 http://static.flickr.com/3127/3161361816_31dc421b73.jpg
21 http://static.flickr.com/88/263747847_e2bd2e6c26.jpg
22 http://static.flickr.com/3444/3373608783_7d42cee80c.jpg
```

Tokenization:

- Tokenization generally tokenizes the sentence or a paragraph into corresponding words and stores them in the form of list.
- There are two ways to do Tokenization namely word tokenization(word_tokenize) and sentence tokenization(sent_tokenize).
- Word Tokenization is used for extracting the required captions from the dataset.
- Using linecache package the urls are extracted after the process of tokenization and store them in the new file.



The screenshot shows the PyCharm IDE interface. The main editor window displays a Python script named `word.py`. The script performs the following steps:

- Initializes counters for `Chicken_count`, `Coke_count`, and `Vegetable_count` to 0.
- Opens `Extracted_tokens.txt` for writing and `Extracted_urls.txt` for reading.
- Defines a set of `punctuations` and a set of `stop_words` from the `stopwords.words('english')` list.
- Iterates over `SBU_captioned_photo_dataset_urls.txt` line by line.
- For each line, it increments `linecounter` and tokenizes the line into `tokens` using `word_tokenize`.
- It then filters out punctuation and stop words from the `tokens` list.
- For the remaining words, it uses `wordnet_lemmatizer.lemmatize` to lemmatize them.
- If the lemmatized word is 'Chicken', it increments `Chicken_count`.
- Finally, it writes the original line to `Extracted_urls.txt` using `linecache.getline`.

The bottom of the IDE shows a run console with an error message: `ImportError: DLL load failed: The specified module could not be found.` This is likely due to a missing system DLL on the Windows machine.

Lemmatization:

- Lemmatization generally produces the output of the word in its root form which can be either in the form of adjective, noun, verb.
- We eliminate stopwords when we do lemmatization process.

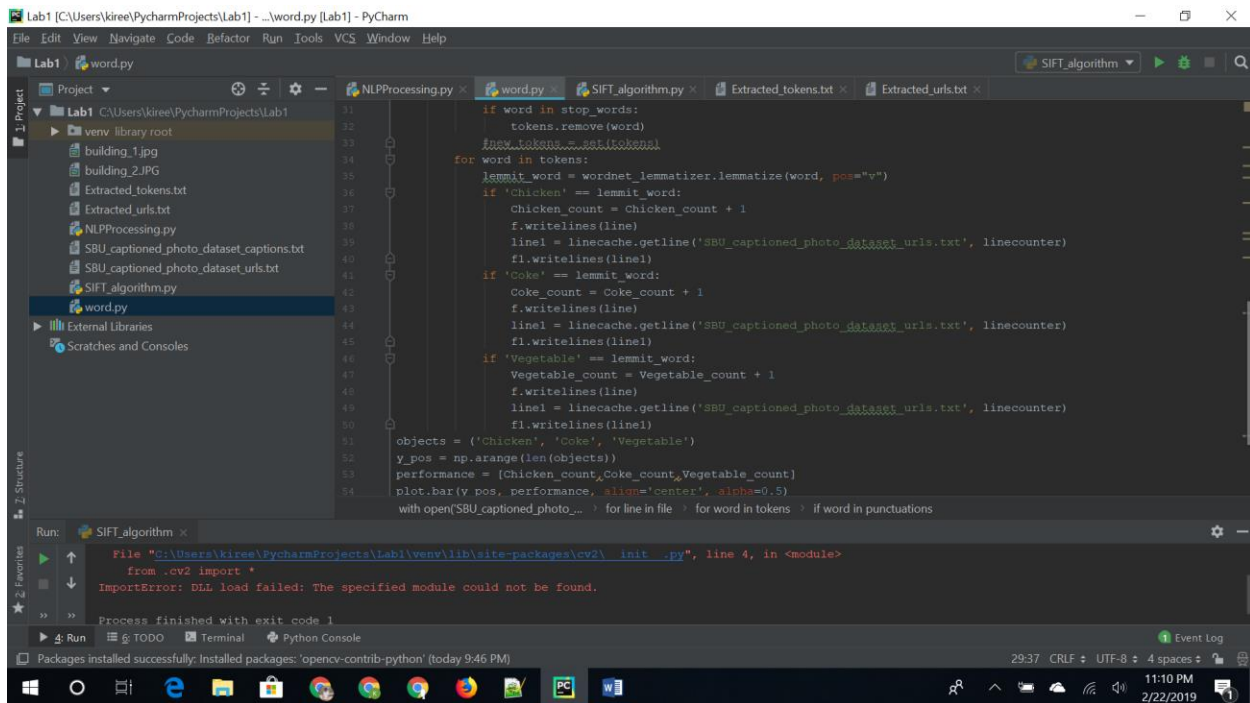
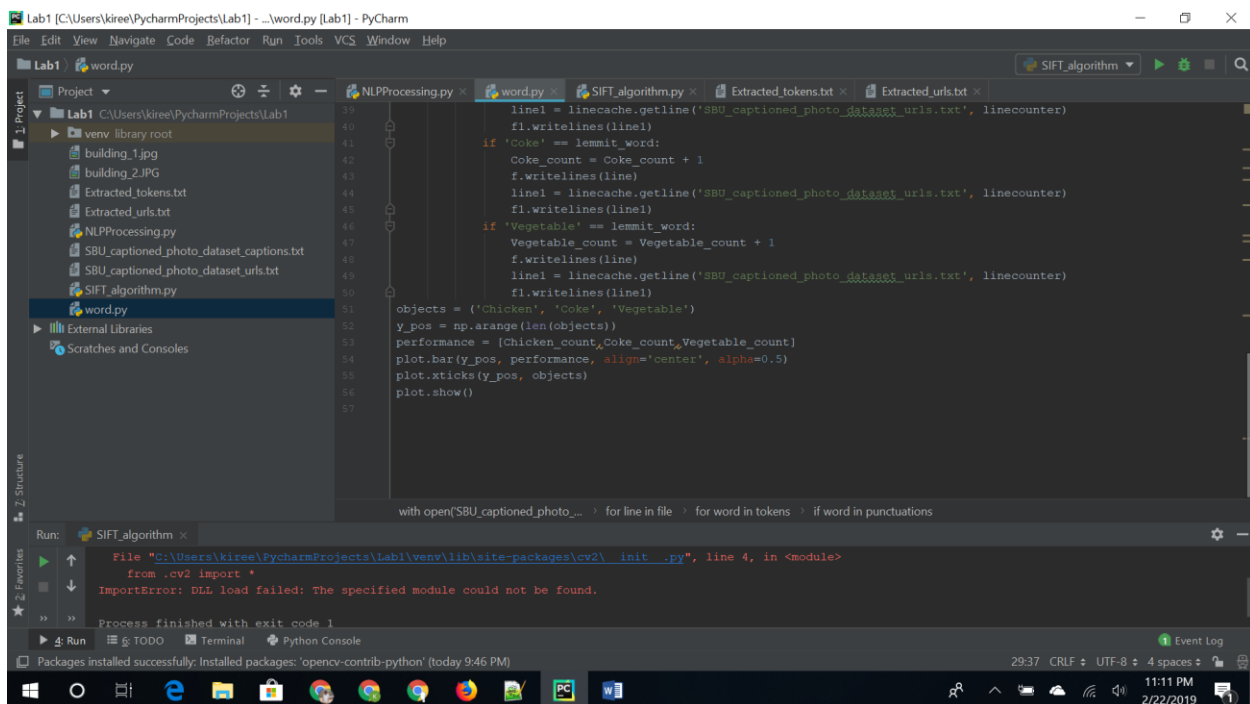
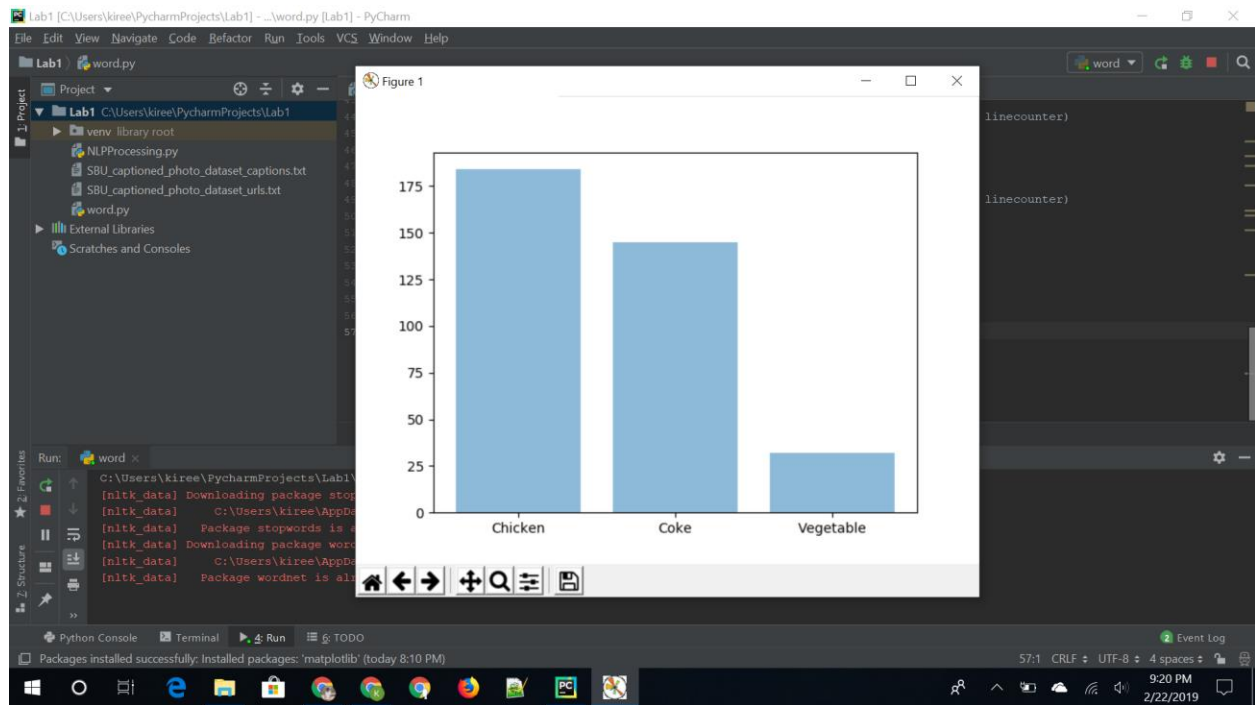
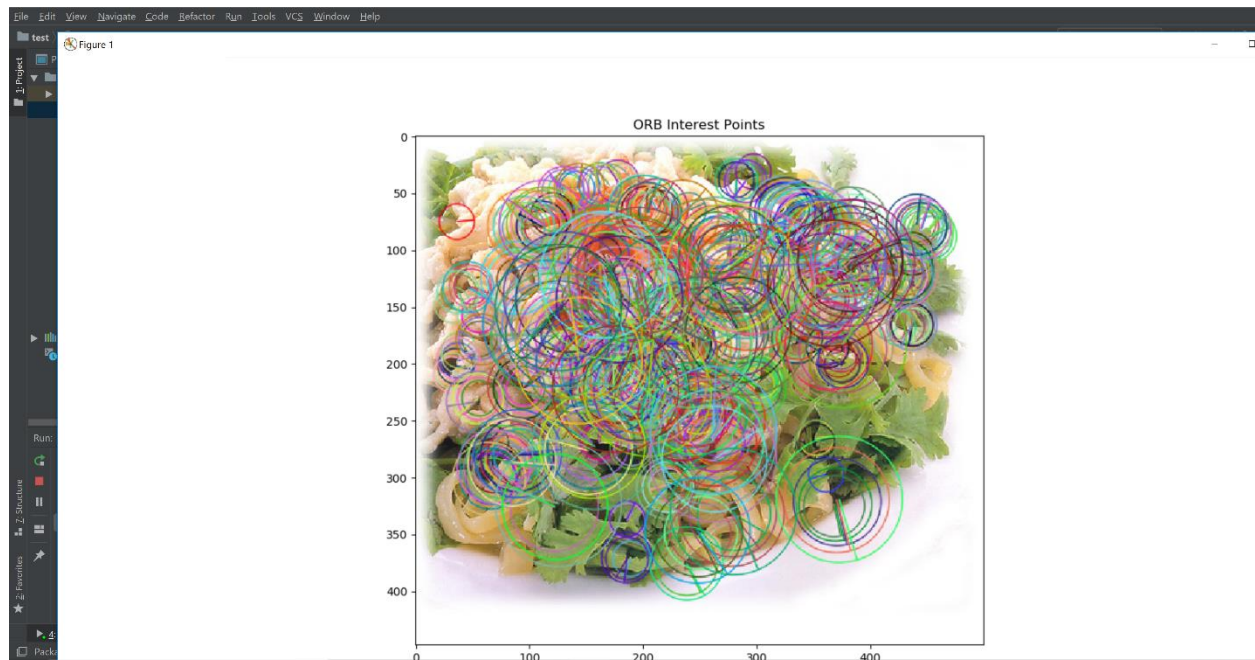


Image Statistics:





SIFT Algorithm:



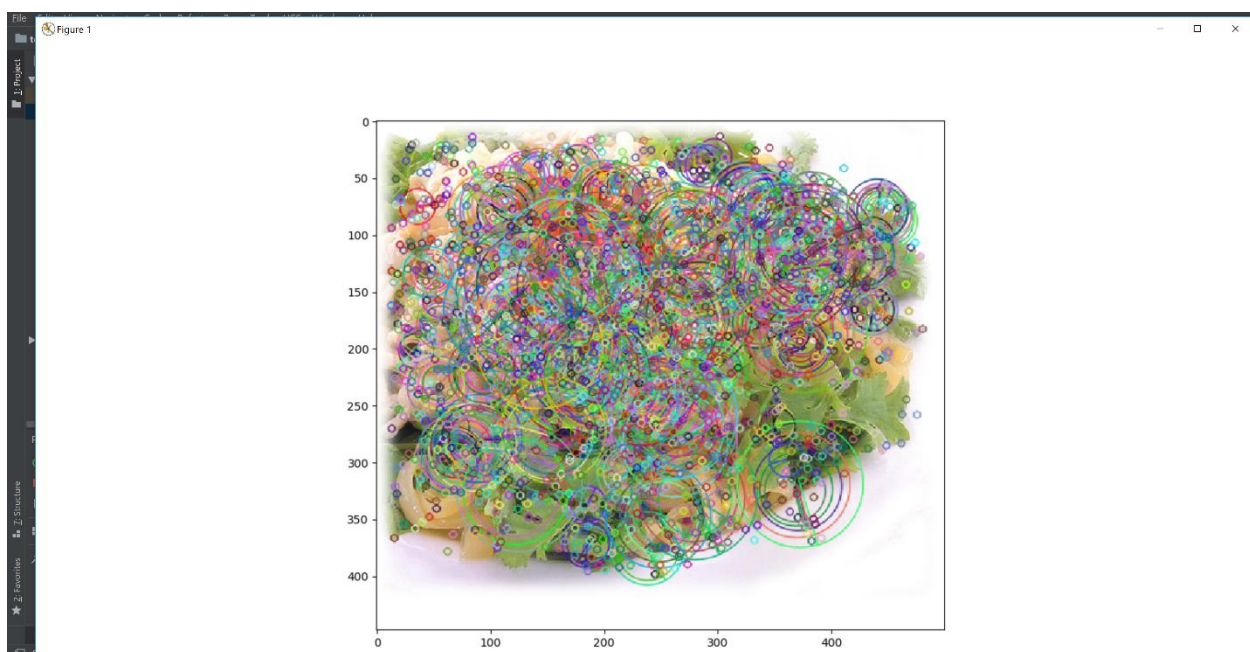
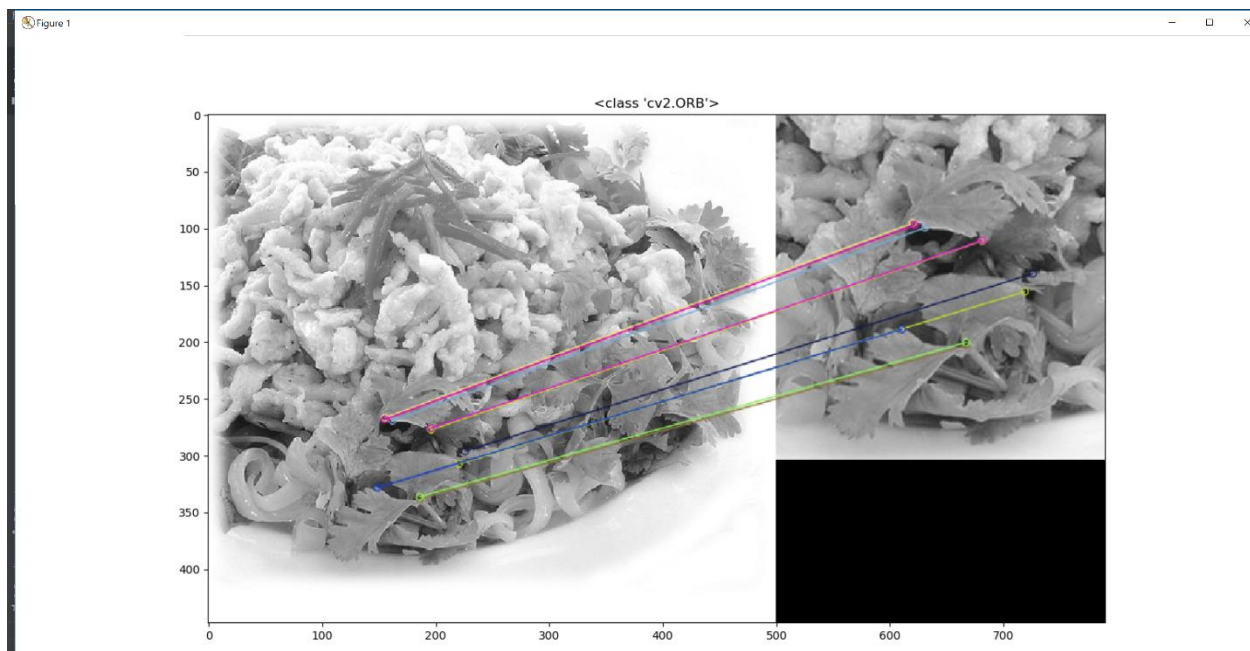
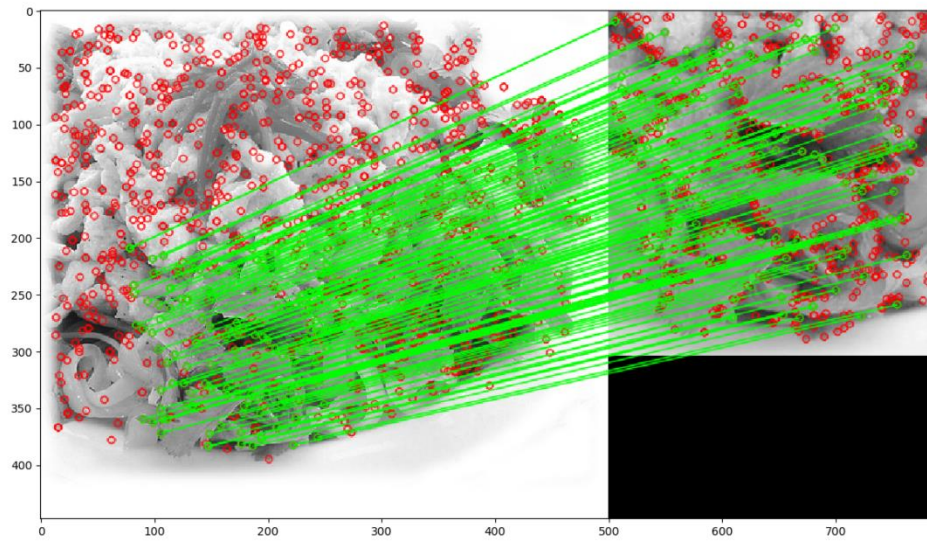


Figure 1



```
Sefactor Run Tools VCS Window Help
SIFT algorithm.py
plt.title('SIFT algorithm')
plt.imshow(img_matches)
plt.show()

orb = cv2.ORB_create()
draw_image_matches(orb, 'building_1.jpg', 'building_2.jpg')

sift = cv2.xfeatures2d.SIFT_create()
kp, des = sift.detectAndCompute(img_building, None)
img_kp = cv2.drawKeypoints(img_building, kp, img_building)

plt.figure(figsize=(15, 15))
plt.imshow(img_kp)
plt.show()

img1, kp1, des1 = image_detect_and_compute(sift, 'building_1.jpg')
img2, kp2, des2 = image_detect_and_compute(sift, 'building_2.jpg')

FLANN_INDEX_KDTREE = 1
index_params = dict(algorithm=FLANN_INDEX_KDTREE, trees=5)
search_params = dict(checks=50)

flann = cv2.FlannBasedMatcher(index_params, search_params)
matches = flann.knnMatch(des1, des2, k=2)
```