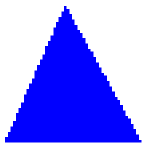
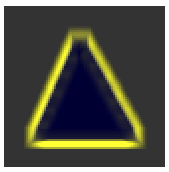
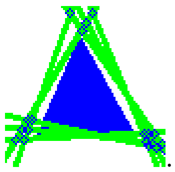
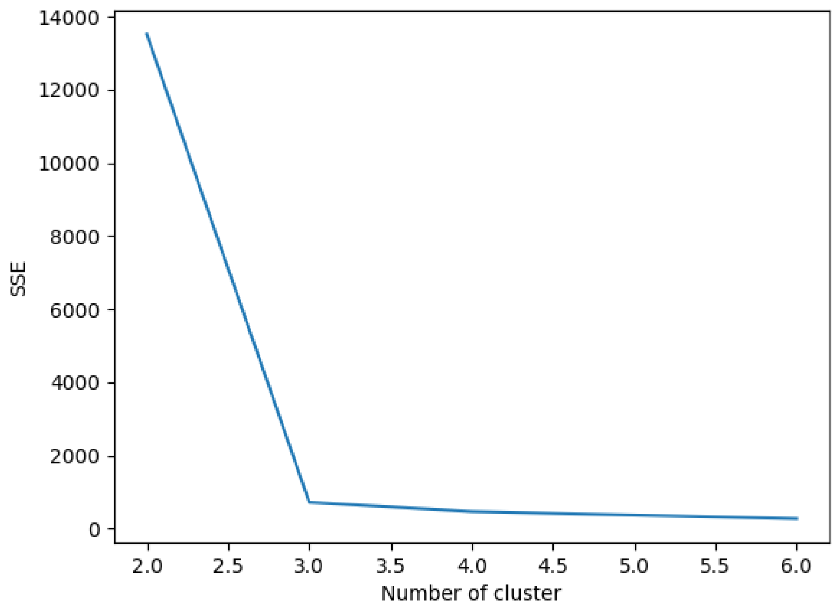
CS786 Assignment 2

Navya Rawat, 160434
Mayank Sharma, 160392

March, 8, 2019

1 Detect the geometry of a shape

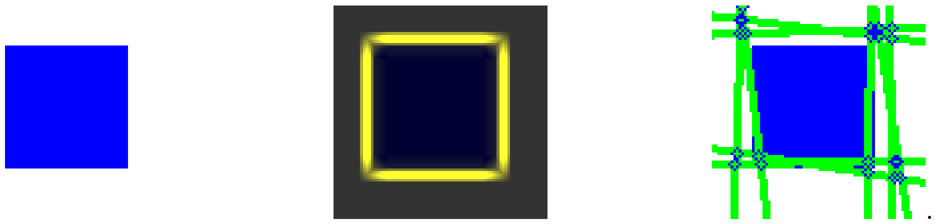


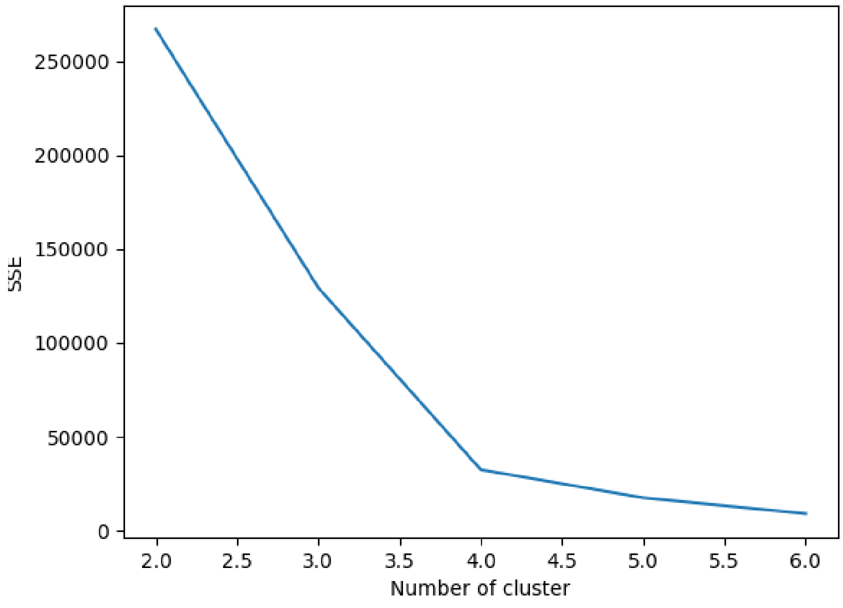
Triangle shape and color used:

1. The first image is the original triangle which is being used throughout this assignment.
2. This is the Gabor Filter applied image.
3. The HoughLines applied Image. The blue dots at corners are the possible vertices of the triangle.

Once we obtain the clusters of image, we use K-Means clustering algorithm with k values from 2 to 6 (inclusive). Finally, we use the elbow method to get the optimum value of k, which in our case is the number of vertices.

Similarly, for square, we get the following.





```
root@msharma:/cv/A2 \$ python q1.py --help
usage: q1.py [-h] [--shape SHAPE]

optional arguments:
  -h, --help       show this help message and exit
  --shape SHAPE    1 for Square, 0 for triangle
```

Listing 1: Output of ‘python q1.py –help‘

2 Generate the images to perform feature/conjunction search

```
root@msharma:/cv/A2 \$ python q2.py --help
usage: q2.py [-h] [-n NUM_OBJECTS] [--conjunction] [--feature FEATURE]
             [--hide_image]

optional arguments:
  -h, --help            show this help message and exit
  -n NUM_OBJECTS, --num_objects NUM_OBJECTS
                        The number of objects to generate
  --conjunction         Supply this flag to simulate conjunction search
  --feature FEATURE     0 for color different, 1 for shape different
  --hide_image          Hide the image generated, useful for q3.py
```

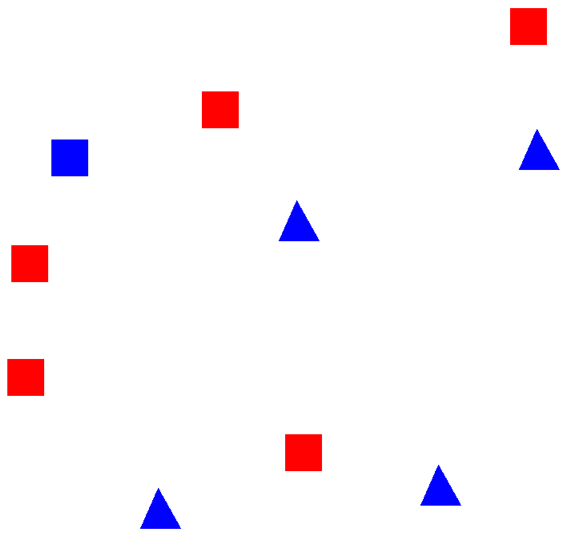
Listing 2: Output of ‘python q2.py –help‘

The file q2.py is used to generate images containing stimuli for feature/conunction search. The parameters ”–function” and ”–conjunction” can be used one by one to get the corresponding stimuli.

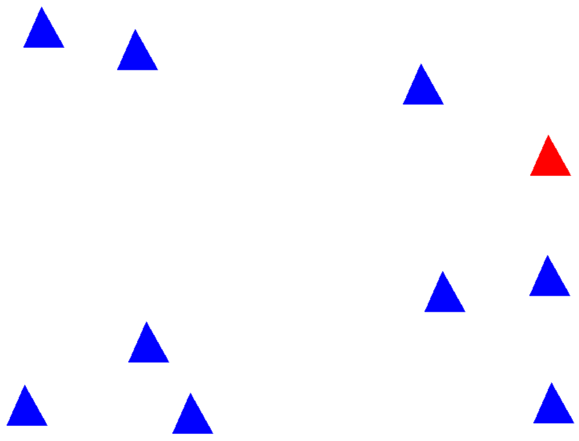
Note: The code has been written to generate random images each time, and occassionally (very rarely) it results into some shapes very close or overlapping. Also, each run of program will give different output and the ration between number of squares to trianges will vary between 0.25 to 4. This also is random.

For ”NUM_OBJECTS = 10”, we get the following results for each of the cases:

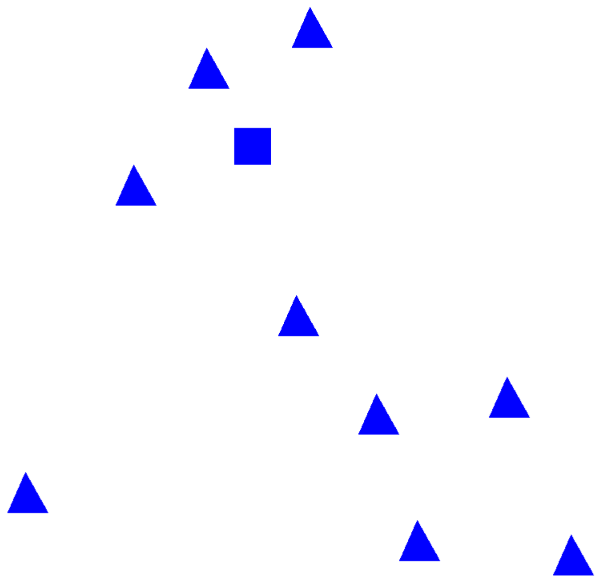
- **Conjunction Search with 1 square of blue color, while remaining of red color**



- **Feature search with ”–feature 0” for a single different color**



- **Feature search with ”–feature 1” for a single different shape**



q2.py is used to generate two files – ”q3_input_image.png” and ”q3_input_locations.csv”. The image contains the stimuli which will be used for visual search. The csv file contains the locations of the triangles and squares generated.

3 Perform visual search, and benchmark

```
root@msharma:/cv/A2# usage: q3.py [-h] [--benchmark]

optional arguments:
  -h, --help    show this help message and exit
  --benchmark   Perform timing benchmarks using num_object values [10, 20,
    30,
                40, 50]
```

Listing 3: Output of ‘python q3.py –help‘

The file q3.py is used to benchmark (i.e. get output for various num_objects values) and also time the feature/conjunction search. The code firstly uses the "q3_input_image.png" and "q3_input_locations.csv" files generated using "q2.py". The csv file contains the locations of the shapes in the image. For each shape, we run the predict() function in "q1.py".

Note: The only type of data passed between q2.py and q3.py is the image file and the csv file (and obviously global variables of q2.py). This ensures that no "cheating" occurs.

For the 1st stimuli generated in the above section, we get the following output from q3.py.

```
{'Red Triangles': 0, 'Blue Squares': 1, 'Blue Triangles': 4, 'Red Squares':
    5}
The search type used = Conjunction
The total time elapsed in search = 1.539934 seconds
```

Listing 4: Output of ‘python q3.py ‘

For the 2nd stimuli generated in the above section, we get the following output from q3.py.

```
{'Red Triangles': 1, 'Blue Squares': 0, 'Blue Triangles': 9, 'Red Squares':
    0}
The search type used = Feature
The total time elapsed in search = 1.024634 seconds
```

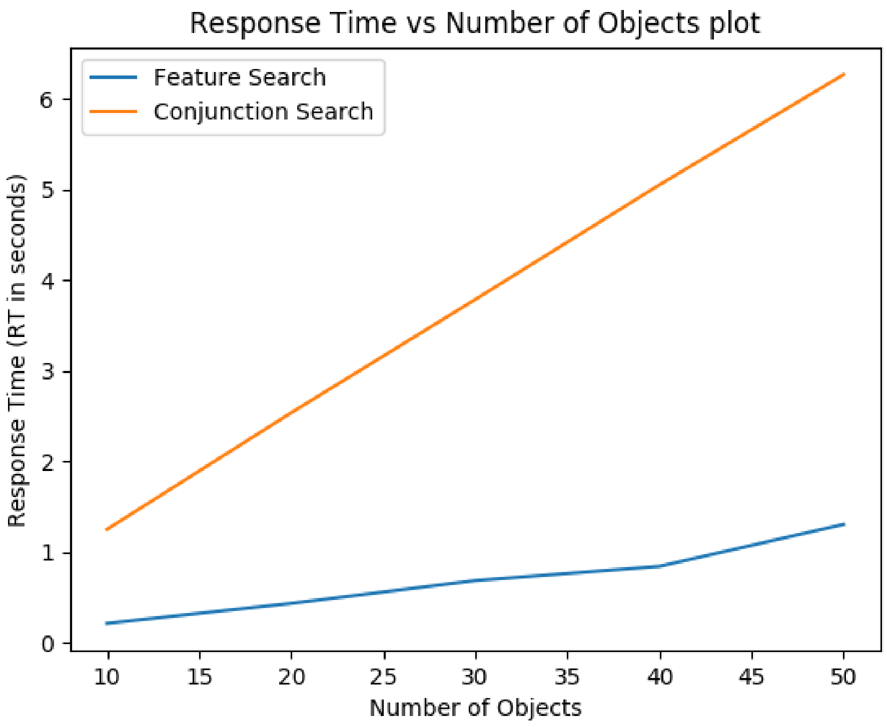
Listing 5: Output of ‘python q3.py ‘

For the 3rd stimuli generated in the above section, we get the following output from q3.py.

```
{'Red Triangles': 0, 'Blue Squares': 1, 'Blue Triangles': 9, 'Red Squares':
    0}
The search type used = Feature
The total time elapsed in search = 1.055122 seconds
```

Listing 6: Output of ‘python q3.py ‘

When q3.py is run in benchmark mode ("–benchmark" flag supplied), we get the following output as graph:



The full commandline output produced by q3.py ran in benchmark mode is as follows:

```
---------- FEATURE SEARCH ----------
q3_input_locations.csv and q3_input_image.png have been generated.
The total time elapsed in search = 0.213872 seconds

---------- FEATURE SEARCH ----------
q3_input_locations.csv and q3_input_image.png have been generated.
The total time elapsed in search = 0.433549 seconds

---------- FEATURE SEARCH ----------
q3_input_locations.csv and q3_input_image.png have been generated.
The total time elapsed in search = 0.685330 seconds

---------- FEATURE SEARCH ----------
q3_input_locations.csv and q3_input_image.png have been generated.
The total time elapsed in search = 0.842324 seconds

---------- FEATURE SEARCH ----------
q3_input_locations.csv and q3_input_image.png have been generated.
The total time elapsed in search = 1.305423 seconds

---------- CONJUNCTION SEARCH ----------
Number of Triangles (Randomly chosen) = 5
Number of Squares (Randomly chosen) = 5
q3_input_locations.csv and q3_input_image.png have been generated.
The total time elapsed in search = 1.250235 seconds

---------- CONJUNCTION SEARCH ----------
Number of Triangles (Randomly chosen) = 9
Number of Squares (Randomly chosen) = 11
q3_input_locations.csv and q3_input_image.png have been generated.
The total time elapsed in search = 2.536259 seconds

```

```
————— CONJUNCTION SEARCH —————
Number of Triangles (Randomly chosen) = 19
Number of Squares (Randomly chosen) = 11
q3_input_locations.csv and q3_input_image.png have been generated.
The total time elapsed in search = 3.786642 seconds

————— CONJUNCTION SEARCH —————
Number of Triangles (Randomly chosen) = 27
Number of Squares (Randomly chosen) = 13
q3_input_locations.csv and q3_input_image.png have been generated.
The total time elapsed in search = 5.051007 seconds

————— CONJUNCTION SEARCH —————
Number of Triangles (Randomly chosen) = 33
Number of Squares (Randomly chosen) = 17
q3_input_locations.csv and q3_input_image.png have been generated.
The total time elapsed in search = 6.267516 seconds
{'Conjunction': [1.2502350807189941, 2.5362589359283447, 3.786642074584961,
    5.05100679397583, 6.2675158977508545], 'Feature': [0.21387195587158203,
    0.4335489273071289, 0.6853299140930176, 0.8423240184783936,
    1.3054230213165283]}
```

Listing 7: Output of ‘python q3.py ‘

Conclusion

From the graph of RT vs Num objects, we can see that the time to perform conjunction search increases linearly with number of objects. The line has ¿1 slope, whereas the one of feature search has much smaller slope. In theory, feature search should give us nearly constant time, but owing to the cache misses (and other aspects), we get this little time.