**CSCI 6368: Computer Vision**

Spring 2021

Project 4: ORB (Oriented FAST and Rotated BRIEF)

**Name:** Md Rysul Kabir

**Submission date:** March 9, 2021

**Introduction:**

There are many features (descriptor) we can extract from images to discriminate one from another. These features can include calculation using fourier transform on the image pixel as well as the colors of the images. What is more, the extracted histogram of the gradients (HOG) of the image can significantly help us differentiate among multiple images. Using the notion of of HOG and image pyramids we can acquire even more distinctive features from images. ORB is one of them.

This project consists of extracting features using Oriented FAST and Rotated BRIEF (ORB) method. There are other different methods to extract features from images like Scale Invariant Feature Transform (SIFT) and Speed Up Robust Features (SURF). ORB is said to be a better alternative to SIFT and SURF in terms of computational cost and matching performance.

ORB has two components: i) Features from Accelerate and Segments Tests (FAST) and ii) Binary Robust Independent Elementary Feature (BRIEF). FAST provides information about the possible location of the important edges in an image. For a given pixel in a circular window, it compares its brightness with the surrounding 16 pixels of the window. Finally, depending on the pixel values, they are sorted into three categories (darker, brighter, equal to the pixel in concern). If 8 pixels are darker or brighter than the pixel, the pixel is decided to be a key point. As FAST does not have any scale variant feature, ORB algorithm incorporates multiscale image pyramids where the FAST algorithm provides the key points of every image level. After locating the key points, ORB assigns an orientation using intensity centroids.

On the other hand, BRIEF takes in all the key points found by the FAST algorithm to convert them into finery features where each feature vector can be describes using 128-512 bits string. BRIEF starts off smoothing the image using a Gaussian Kernel after which it selects a random pair of pixels in the neighborhood of the key point in concern. The first pixel is drawn from a normal distribution of one standard deviation whereas the other pixel is drawn from a normal distribution of twice of that standard deviation. If the first pixel is brighter than the second one, it is assigned with 1 otherwise it is assigned with 0. BRIEF keeps assigning such values until all the key points are exhausted.

Timeline

Description automatically generated

**Figure 1:** Flow diagram of the overall process of this project.

**Results:**

**Tables of objects used and their corresponding ORB features:**

I had to use a calculator instead of a remote.

|  |  |  |
| --- | --- | --- |
| **Object Name** | **Object Image** | **Object ORB Features** |
| Key |  | A picture containing text, green  Description automatically generated |
| Watch |  | A picture containing text, green  Description automatically generated |
| Calculator | A picture containing indoor, keyboard, electronics, cellphone  Description automatically generated | A picture containing electronics, cellphone  Description automatically generated |
| Phone | A picture containing text, indoor, different  Description automatically generated | A picture containing text, indoor, different, plastic  Description automatically generated |

Below I am adding some queries using the GUI.

**Queries:**

Graphical user interface

Description automatically generated

**Figure 2**: Query 1

Graphical user interface

Description automatically generated

**Figure 2:** Query 2

**Discussions:**

Overall, the project was quite interesting. One of the best essences of computer vision is object detection. Nowadays, most of the scholars are using complex neural networks to perform such tasks. However, with the implementation of the project I came to know that, though not as versatile as computer vision, feature extraction from images using image processing concepts can be used to detect different objects in an efficient way.

The most challenging part of this project is to implement the GUI using Python’s Tkinter framework. While not familiar with this framework at all, I was able to successfully implement the GUI part for the object detection.

Moreover, to calculate the overall accuracy of this object detection system, I have created an image dataset which were labelled manually by me. The non-GUI script was used to detect overall accuracy of the system I built for this project. Interestingly, I acquired to have a 100% accuracy on my dataset. However, the most significant constraint of this system is using a brute force matcher which uses the Hamming Distance to calculate the dissimilarity of the descriptor where the lowest distance means they are more similar. As a result, we can raise a question about cases where there will be objects in an image which are not among the 4 objects used for this project. At that situation, this object detection will provide erroneous detection results. What is more, because of the matching procedure it will be able to successfully detect one of them, if an image has more than one of these 4 objects.

**Conclusion:**

The most significant enhancement that can be done on this project is using a better matcher where we can use a metric to certain tell if a certain object is there or not. What is more, if that matcher can provide detection capability of multiple objects at the same scene, that will be a significant improvement as well.

Besides the improvement regarding the matcher, the user interface for this system can be more enhanced with better knowledge of TKinter.