Name: Ashutosh Upadhye ID:1001581542

Aditya Karve ID:1001550601

1)**Introduction:**

Our primary goal was to convert the Feature Vector Image-Processing system from mat-lab platform to OpenCV platform for Java binding. OpenCV has several open-source native implementations of algorithms that could be used to extend this system and serve as a small extension to the MavVStream.

2)**OpenCV Libraries**:

Currently we are used OpenCV 2.4.9 with Java JDK 1.8.0\_161 which is a proper and working combination. OpenCV 2.x version have free usable versions of SIFT and SURF algorithms. The OpenCV 3.x versions don’t use C++ wrappers of SIFT and SURF algorithms. There are several work-arounds available to use native C++ wrappers of feature detection algorithms in OpenCV 3.x system but they need to be custom compiled and interfaced with JNI system.

3)**Implementation**:

The main implementation of the system has been divided into three parts:

1)Bounding Box generation and Frame subtraction

2)Feature Detection using HIST and SIFT algorithms

3)Implementation of heuristics to compare group objects.

**Bounding Box and Image Subtraction**

* Currently the system reads the file Config.txt in the input folder in the project directory. It plays the video and reads the frame one by one. The displayed frame gets persisted to “frame storage directory” folder.
* In the file “read\_gray.java”, we read consecutive three frames from the frame storage directory, and we convert them to grayscale frames.
* We subtract “frame l”,”frame m”, “frame n” and select the minimum of both frames. Thus based on a given threshold we obtain proper subtracted images in the “background subtract” method.
* The main goal of image subtraction is to obtain moving entities from the given frames.
* We can use OpenCV API to detect contours in the images and draw a rectangle over the contour extremities.
* This rectangle coordinates are then superimposed over the colour frames to display bounding boxes.
* The threshold for each video might be variable.
* Once we plot the bounding boxes over moving objects. In “process\_objects.java” we crop those bounding box objects and store them in “object storage directory”.
* We create an object frame map which determines the total number of objects cropped from the moving frames.

Currently there are better background detection algorithms available in OpenCV.

**HIST and SIFT implementation**

* Once we have cropped images, we can actually read the object frames from “object storage directory“. We can use “CalcHist()” function of OpenCV to determine histogram pixel values of an image. Then we can split the arraylist of matrices to different R G and B spectrum values.
* Once that is done we get 256x1 matrix for each channel. Combining all values we might get a total of 768 values in a matrix arraylist. Please check Hist\_Feature.java file for the code.
* Similarly for two different images we can use the above process and take two image feature vectors and compare the histograms using CompareHist() and detect similarities between two images.
* Similarly for SIFT we can use SIFT feature Detector and compute feature vector values. In “Sift\_Feature.java” we normally use descriptor match to compute descriptors of a sift feature vector.
* Similarly in “SIFT\_MATCH.java” we can compare two image feature vectors and descriptors using descriptor matcher function and output a similarity index.

**Heuristic Implementation**

* The group frame map normally contains all groups ids, count comparisons and image paths.
* We can compare the object being generated in a current frame with the last frame.
* We compute it using “Hist\_match.java” and “Sift\_match.java”.
* For heuristic 1 we compare only the similarity output of two frames and if they are below a specific threshold we generate a new group id. Please view “heuristic 1.java” for code
* For heuristic 2 we normally use the bounding boxes of objects generated in the current and previous frame. If x-coordinate and y coordinate difference of both bounding boxes is greater than threshold. Then generate a new group id. Please view   
  “heuristic 2.java” for the code.
* For heuristic 3 or exhaustive heuristic, we are matching the currently processed frame with all the other processed frames that are in the group frame map. One more improvement over the exhaustive heuristic is wiping most of the entries from the group frame map so that the comparisons for each window are reduced.
* The code for heuristic 3 is in Exhaustive.java and the code for improvement of the same is in

WipeGroupFrameMap.java

The JSON Object is generated for each tuple and printed to a file. The tuple generated is printed to relation.txt in the output directory and objects are printed to relation-objects.txt

The ascii tuple output is being written to relation\_ascii.txt

The seed tuples are printed to tuple\_seed.txt and its ascii output is printed to seed\_tuple\_ascii.txt

The binary file for objects and seeds are being printed to Relation\_object\_serialized.bin

The OpenCV 2.x setup has been documented in OpenCV-2.4.9.txt file in project root directory.

Setup for OpenCV 3.x has been documented in readme.txt file in project root directory

The object and output files are in the output-opencv folder.

Steps to run the system:

1)Please read the readme.txt and opencv-2.4.9.txt to setup OpenCV in your java root project directory.

2)After OpenCV is setup then please open config.txt in input directory and configure the video that is to be played.

3)Once proper path is given in input.txt, run read\_gray.java in OpenCV-test1/src in the project root directory to run the system.

4)You can configure the threshold for each video from the input.txt itself.

5)If you want to use second order histogram algorithm for HIST, then uncomment the code in Hist\_match.java in project root directory.

6)You can reconfigure the tuple output in get\_group.java log\_to\_output function.

7)You can reconfigure the object output in get\_group.java json object code.

8)The JSON object contains the following metric:

* Similarity\_method
* Signature\_method
* Object\_id
* Group\_id
* Bounding Box
* Feature Vector
* Feature\_Vector Algorithm
* Threshold for image subtraction
* BoundBox threshold

9)The tuple format in the output is as follows:

* Frame\_id
* Object\_id
* Group\_id
* Previous\_Object\_id
* Bounding Box
* Feature Vectors