Intrinsic Detective System

Phase I – Report

CS 5543 Real-Time Big Data Analytics

by

Group 6

Sri Harsha Chennavajjala (4)

Priyadarsini Nidadavolu (16)

Tej Kumar Yentrapragada (27)

Chaitanya Sai Manne (14)

Table of Contents

Project Objectives 3

Significance 3

Features 3

Project Uniqueness 4

Approach 4

Data Sources 4

Analytic Tools 4

Analytical Tasks 4

Expected Inputs/Outputs 4

Algorithms 5

Related Work 5

Open Source Projects 5

Application Specification 5

System Specification 5

Software Architecture 5

Design of Big Data Analytics Server 7

Parallelism/Distribution: Task/Data 7

Features, workflow, technologies 8

Design of Mobile Client (smartphone/web) 10

Features, GUI, technologies 10

Activity Diagram (workflow, data, task) 10

Sequence Diagram (interaction/collaboration) 11

Project Management 11

Plan & Project Timelines 11

Project Members 11

Implementation Status Report 12

Project GitHub Repository Link 12

Documentation 12

Work completed 18

Work to be completed 19

Issues/Concerns 19

Bibliography 20

Project Objectives

Significance

In recent times, the use of surveillance cameras by the organizations as well as by the individuals has been increased in order to increase the security to their properties. But most of these surveillance systems do not include the automatic decision making capabilities (machine learning) such as alerting the owners about the possible theft, suggesting items to buy to the customers etc. Our project minimizes the long tedious manual work of identifying the suspects in the surveillance video by recognizing the persons in the video. If the system finds a possible threat, it will immediately alert the security authorities. There by we can reduce the crimes in the organization.

Features

- Database of faces: To develop an application that automates the process of extraction of human faces from a stream of video and stores it in a database.
- Face Recognition: To develop the functionality that compares the human faces from a video stream with the HDFS and recognizes the persons.
- Expression Detection: To identify the expression of the detected person, so as to make a decision on his upcoming reactions/activities.
- Scenario Identification: To develop a system, that can identify the scene of the processing video stream with which we can identify the possible location.
- Notification: To develop the functionality where the application alerts the security personnel if it finds a suspect in the vicinity.
- Reports: The application logs each and every instance of security threat and generates reports like the time of the day at which the possibility of robbery is high, the suspects history of robberies etc.

Project Uniqueness

The Unique Selling Point of the project is that the project not only records the activities, it can also identify the persons in the video by comparing them with the local database and takes appropriate decisions. This process reduces the man power required and also makes it easy for the security personnel to take actions immediately before the situation goes out of hands.

Approach

Data Sources

Data can be streamed from any video capturing device like webcams, handy cams, mobiles, social networking video sources etc.

Analytic Tools

- Spark
- Storm
- Kafka

Analytical Tasks

- Spark will be used to generate the training dataset. The machine will be trained to identify about the possible image or the scene using Spark's MLLib.
- On the other side, Storm will have the testing data set with which the analysis must be done.
- Kafka acts as a producer/consumer, which collects the streaming data from the video source and sends it to the Spark and Storm.

Expected Inputs/Outputs

- **Input:** A continuous stream of video possibly with the human faces is expected as an input to our system.

- **Output:** Person recognition and an alarm notification to the mobile device would be the possible output.

Algorithms

Machine Learning algorithms for Image Classification:

- Decision Tree
- Random Forest

Related Work

Open Source Projects

- **Face recognition using eigenfaces:** This application uses the eigenvectors to map a face to the 'face space'. Later it compares the obtained faces with these eigenfaces.
- **Face recognition using Laplacianfaces:** This application works based on appearance-based face recognition method called the Laplacianface approach. This can be achieved by using locality preserving projections.

Application Specification

System Specification

The system that we develop will have the capability of handling large set of data and also it can produce near real-time analytical results. This can be achieved by using distributed and parallel data processing tools such as Apache Spark, Apache Storm and Apache Kafka.

Software Architecture

The below pictorial representation of architecture shows that there will be an Input video sourcing device which could send the video to Kafka Message Broker. There can be two kinds of video stream that will be given to the Kafka. The training data set which is to train the system about the person we would like to recognize and the testing data set in which we need to identify the person. The training data set that is sent to Kafka will be forwarded to Spark, where it uses its

MLLib (Machine Learning Library) to train the model about the person which it needs to identify. On the other side, the live video from the public areas will be streamed to Kafka, which is then sent to Storm. Storm communicates with the Spark Server and get an idea about the training data set with which it identifies the person in the video frame. When a match is found, it will store the results to database and send an alert to the mobile device.

Spark upon receiving the video, it tries to divide the video in to frames and key frames with which it segregates the video into an image files and stores into Hadoop Distributed File System(HDFS). On the other side Storm upon classifying the images and performing a person match, the final data will be sent back to the Spark to store into HDFS. Kafka being a consumer of the output, will get the final output and an alert system will be triggered. Apart from these feature, we would also like to find out the scene of the video, where it is happening based on the video that's been processed and also to identify the expression of the person whom we identified so as to make a decision on his upcoming activities. For ex: if a person is aggressive, we can predict that he is up to something, else if his expression is showing as tensed, he might have got into some trouble which might involve other culprits.

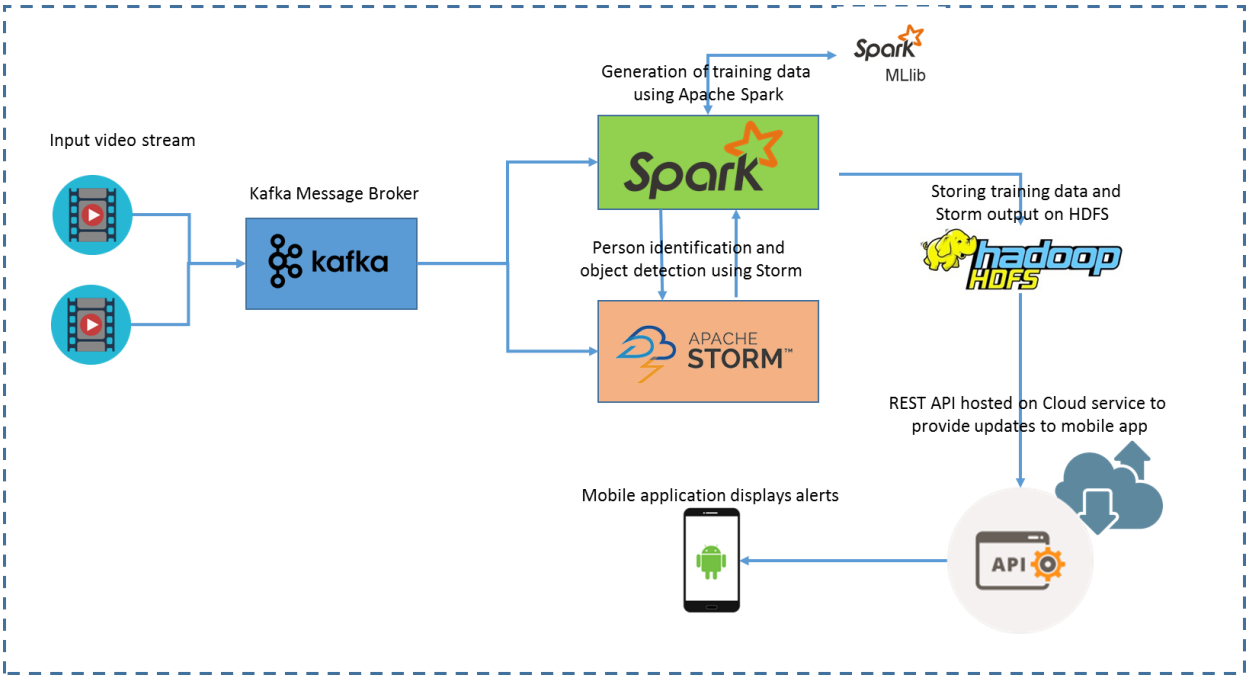

Fig. Software Architecture Diagram

Design of Big Data Analytics Server

Basically, our analytics server has been designed to detect the person, the situation of the scene and his expressions. This design is achieved by making use of the existing servers namely Spark, Storm and Kafka. The streaming video will be sent to Kafka, which acts as a producer and consumer in our system. Kafka processes the video and sends it to the Spark. Spark will build a model by using MLLib. This model is trained to know the face of a person whom we want to detect.

On the other side, Storm receives a video stream from Kafka. Storms uses this data for its testing purpose. Storm communicates with the Spark to get the trained model which is used in person detection. So, Storm perform parallel processing, because of which our model will be faster in detection. Lastly, upon detecting the person the output data will be sent to Kafka. In this case, Kafka acts as a consumer. Being a data pipeline, Kafka triggers an alert to Android mobile device.

Parallelism/Distribution: Task/Data

Models are used to classify the process to extract the human faces from a stream of video and put it in database. Machine learning algorithms are used here and can be processed on multiple nodes and send to the master node to process this.

Storm: A batch processing engine which can undergo micro-batch processing and works on task parallel computations. It has a distributed architecture where each bolt can be parallelized which in turn displays the topology in parallelism. Parallelism is done on each node and feature extraction is executed on these nodes.

Spark: Apache Spark uses the map-reduce paradigm and perform computations ina distributed manner. The inbuilt machine learning library name MLLib, has be used to train the model. Also Spark uses HDFS (Hadoop Distributed File System) as a data storage unit. SparkSQL can be used to query the data and to produce results.

Features, workflow, technologies

Spark is contributed as a UI and will be trained to identify the image from spark lib. It's been used to store the data. The Storm is used for persona and object detection. All the analyses from the videos are done in Storm. Storm converts the raw streaming data into complete data products. Kafka is used as a Pub/Sub real-time messaging system which provides durability and fault-tolerance. Kafka acts as a message broker between the Spark and Storm.

Storm: Apache Storm has the ability to process very large amount of real-time data. There will be many nodes in a cluster and each node is very fast and can process millions of data. There are three abstractions termed as spout, bolt and topology. Spout acts like an input i.e. the source of data which receives data from Twitter API or Kafka or any source which has the information. Bolt on the other hand process this incoming data from spout and produces many output streams. Finally, topology is the network of spouts and bolts, where the edges are connected to bolts. It is language independent and can support all languages.

Spark: Apache Spark is an extended map-reduce paradigm which supports more types of computations and has been designed to be fast. It supports higher-level tools and APIs in SparkSQL, Java, Scala, R etc. SparkSQL is used for structured data processing which uses SQL and Hive. It also provides API for graph operations. Spark works on batch processing which can process millions of data on thousands of nodes very quickly.

Kafka: Apache Kafka is distributed messaging system which commits log services. It is implemented in Java and Scala which has a high throughput to supports huge volume of data. It also supports real-time processing and handles system or machine failures. Kafka has three elements producer, consumer and Broker. Producers write the data to the brokers. Consumers read the data from brokers to process further. All the data is stored in the form of Topics which are splitted into partitions and each partition is replicates. It has high writes and reads. For live streaming recently Kafka introduced Zookeeper where streaming is done through Spark package.

Workflow Diagrams:

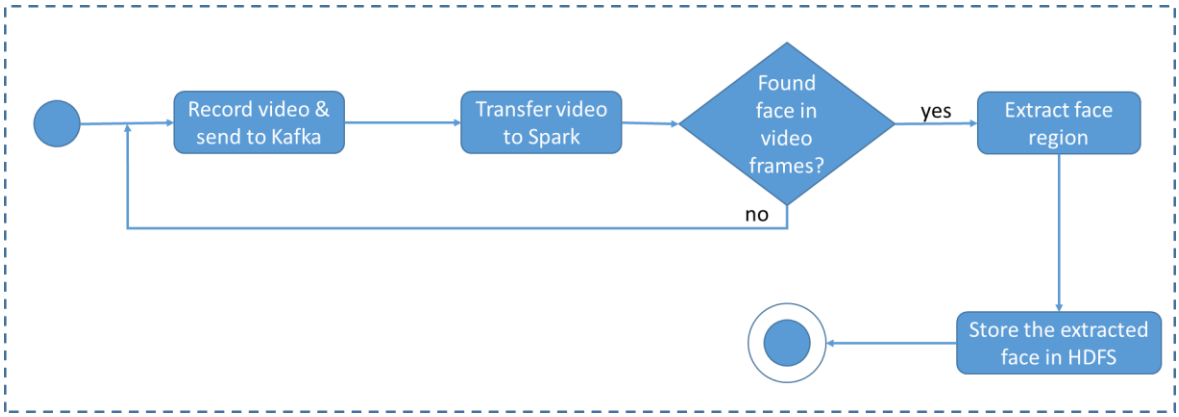

Fig. Activity diagram for collection of training dataset

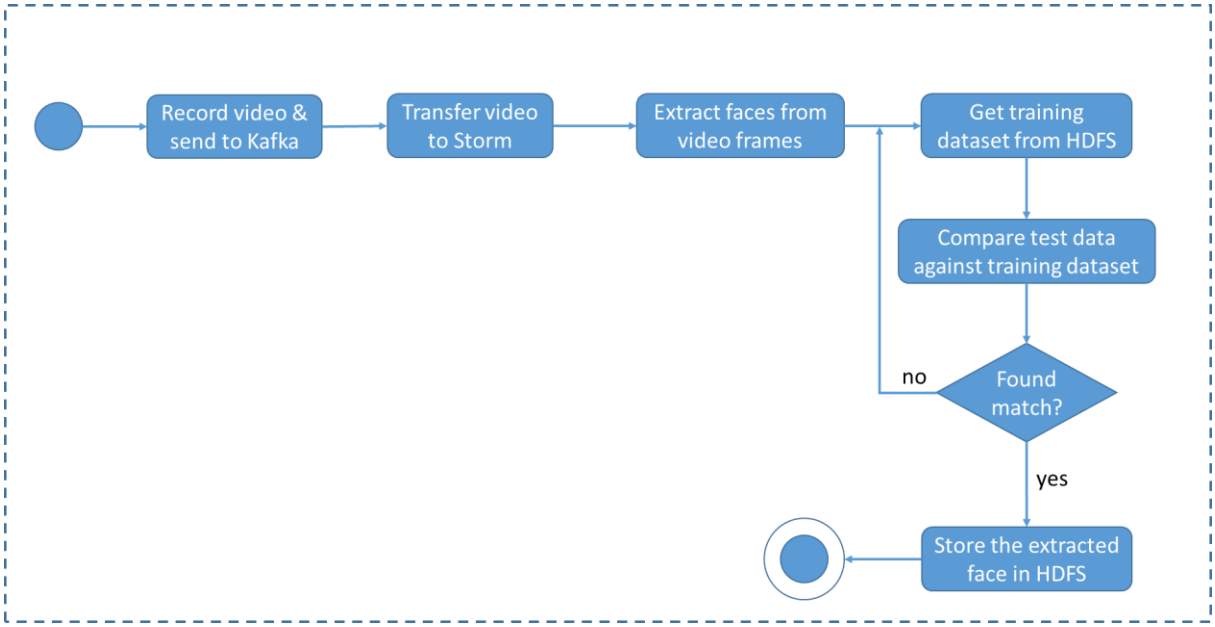

Fig. Activity diagram for person recognition

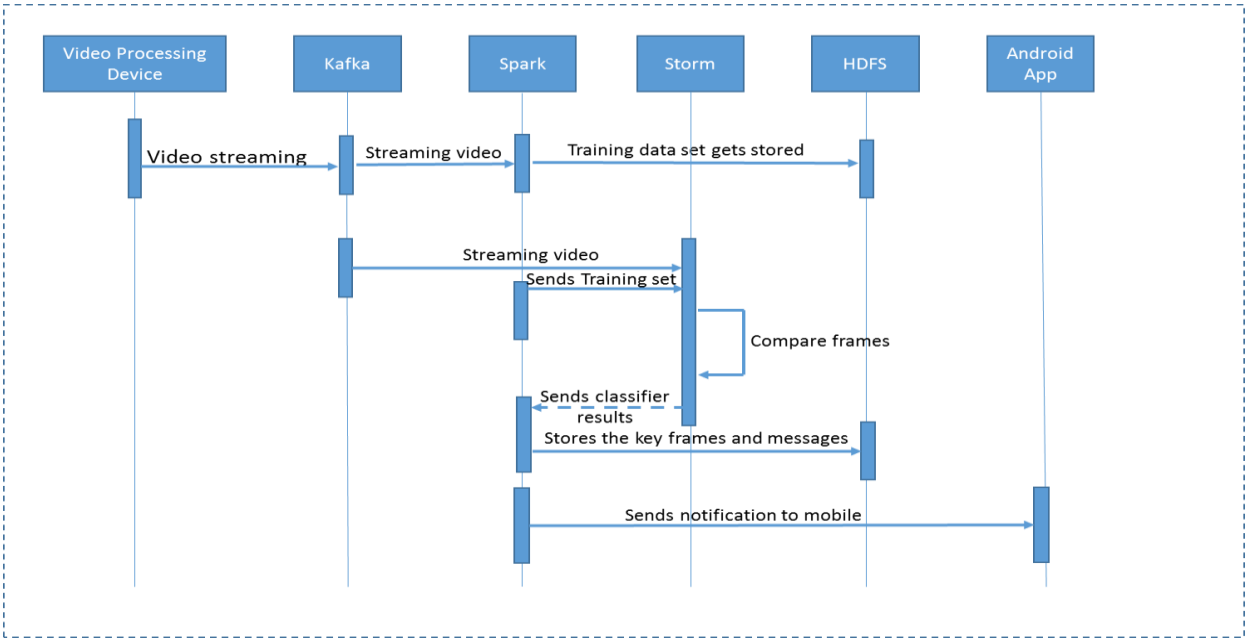

Fig. Sequence diagram for the system workflow

Design of Mobile Client (smartphone/web)

A smart phone application which is Android based will be designed in order to get an alert message. This application basically is helpful in identifying the possible threat activities.

Features, GUI, technologies

A push notification will be shown, when there is any trigger from Kafka.

Android Java will be used to design the Android Application.

Activity Diagram (workflow, data, task)

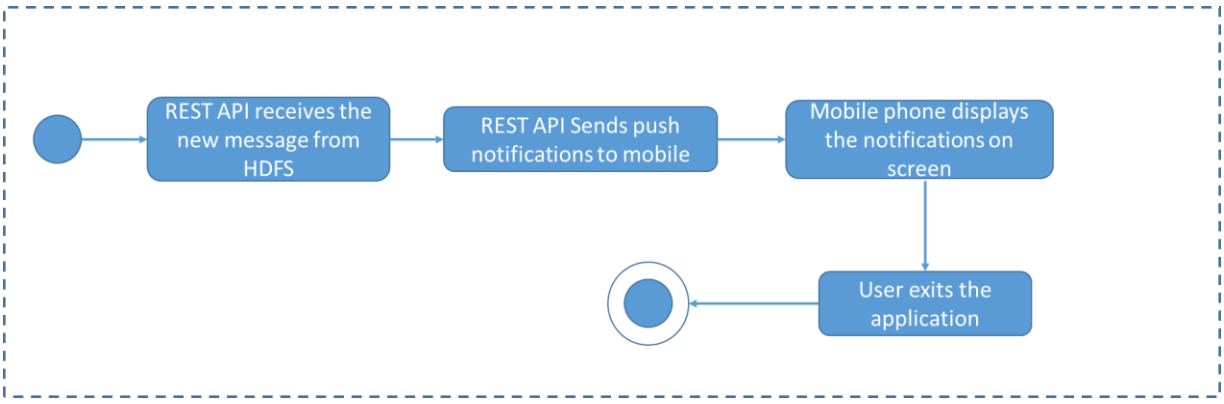

Fig. Activity diagram for alert message display on mobile

Sequence Diagram (interaction/collaboration)

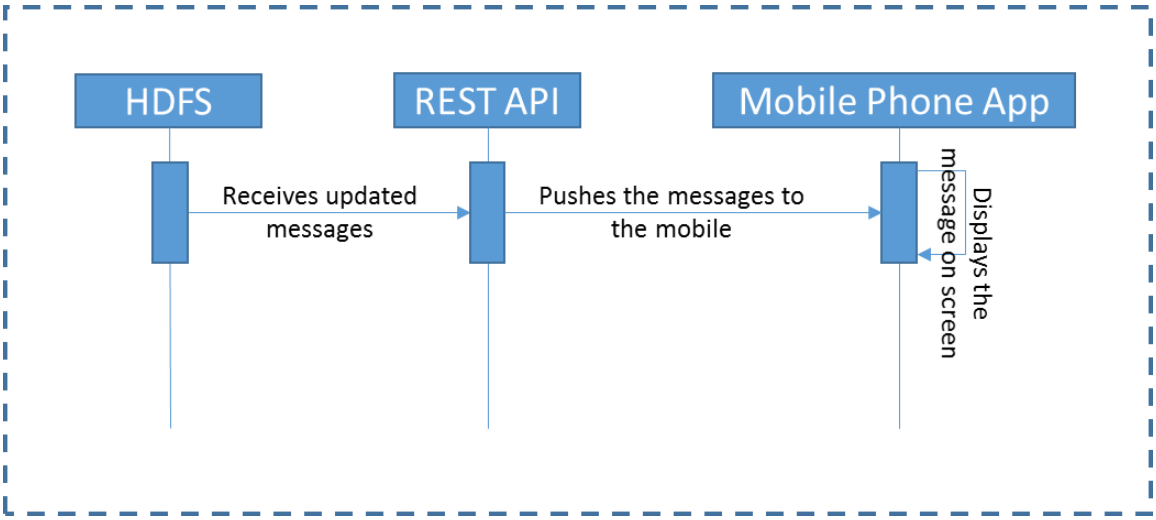

Fig. Sequence diagram for the system workflow

Project Management

Plan & Project Timelines

Increment	Deadline
Increment 1	23 September 2016
Increment 2	14 October 2016
Increment 3	18 November 2016
Increment 4	2 December 2016
Final Submission	9 December 2016

Project Members

- Sri Harsha Chennavajjala (4)
- Priyadarsini Nidadavolu (16)
- Tej Kumar Yentrapragada (27)
- Chaitanya Sai Manne (14)

Implementation Status Report

Project GitHub Repository Link

https://github.com/npdarsiniOrg/RTB-Project/tree/master/Source/Phase-1

Documentation

Key frame detection and tags generation: We used SIFT techniques to find the key frames in the video. Later we passed the key frames to Clarifai API to generate the tags for the objects in the video frames. These tags were added to each video frame and we created a short video. Output of this short video is as shown in the below figures.

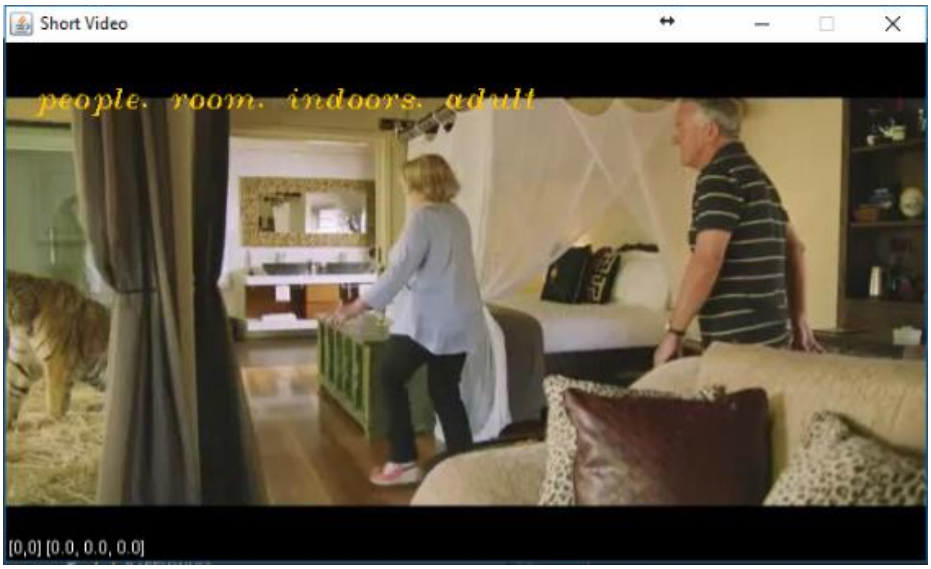

Fig. Key frame with relevant tags

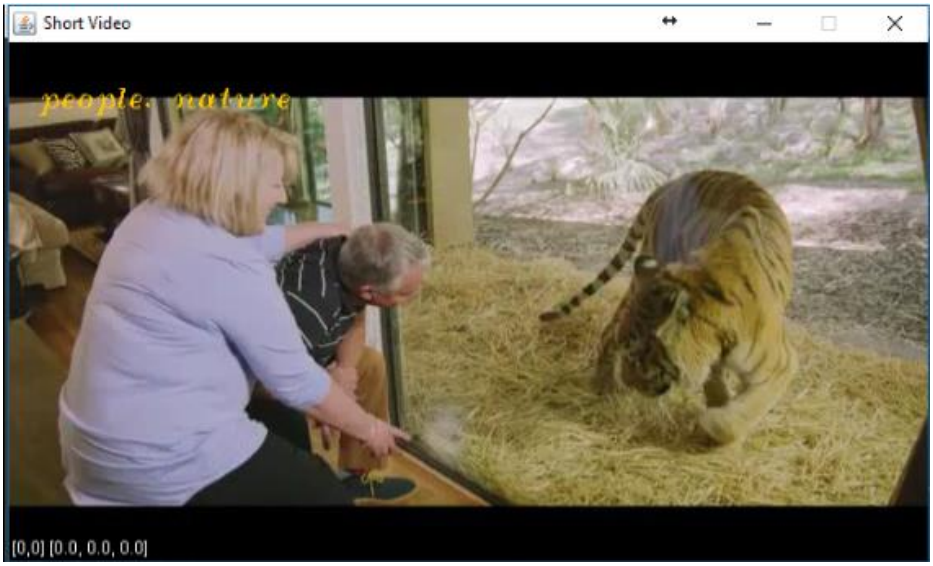

Fig. Key frame with relevant tags

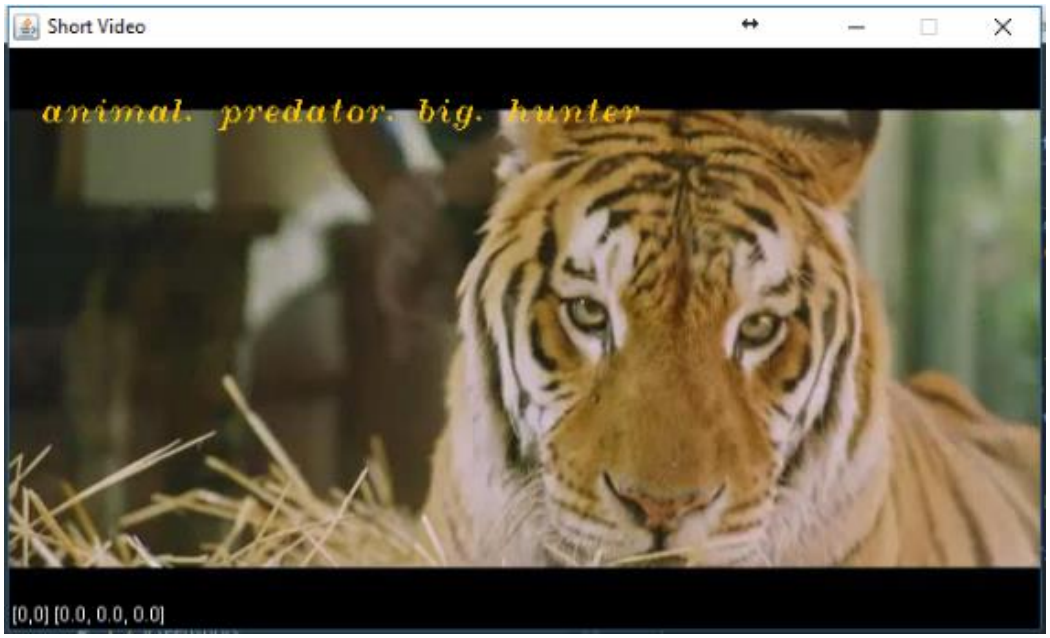

Fig. Key frame with relevant tags

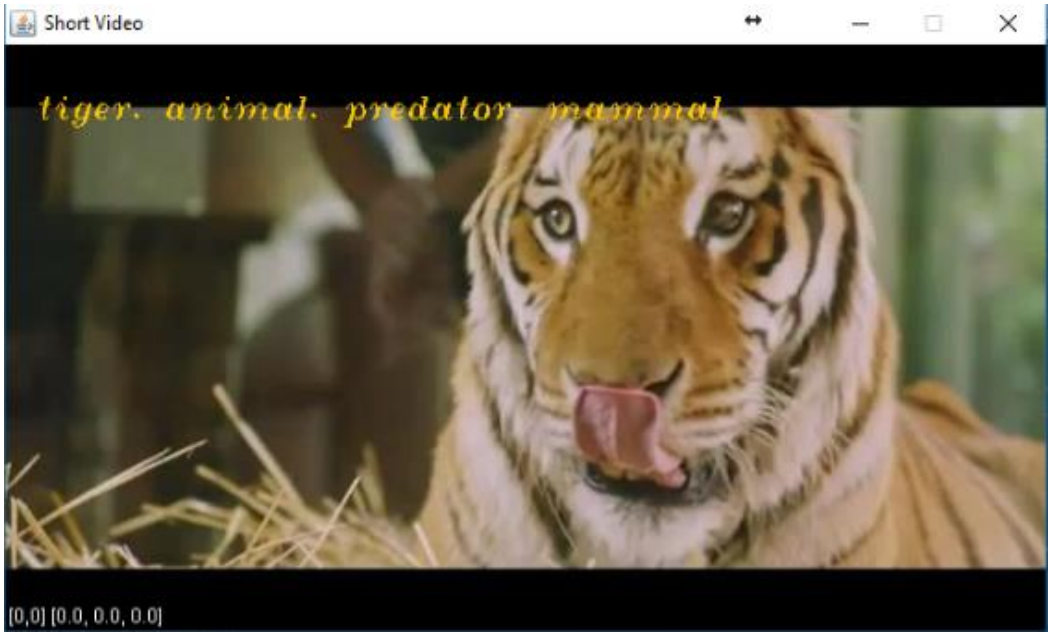

Fig. Key frame with relevant tags

Fig. Key frame with relevant tags

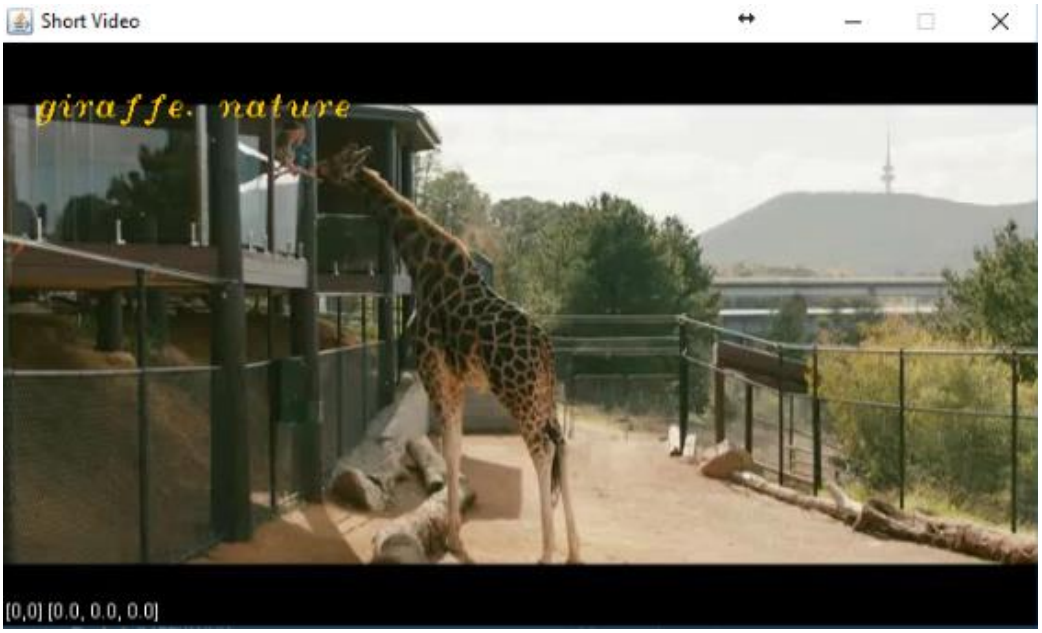

Fig. Key frame with relevant tags

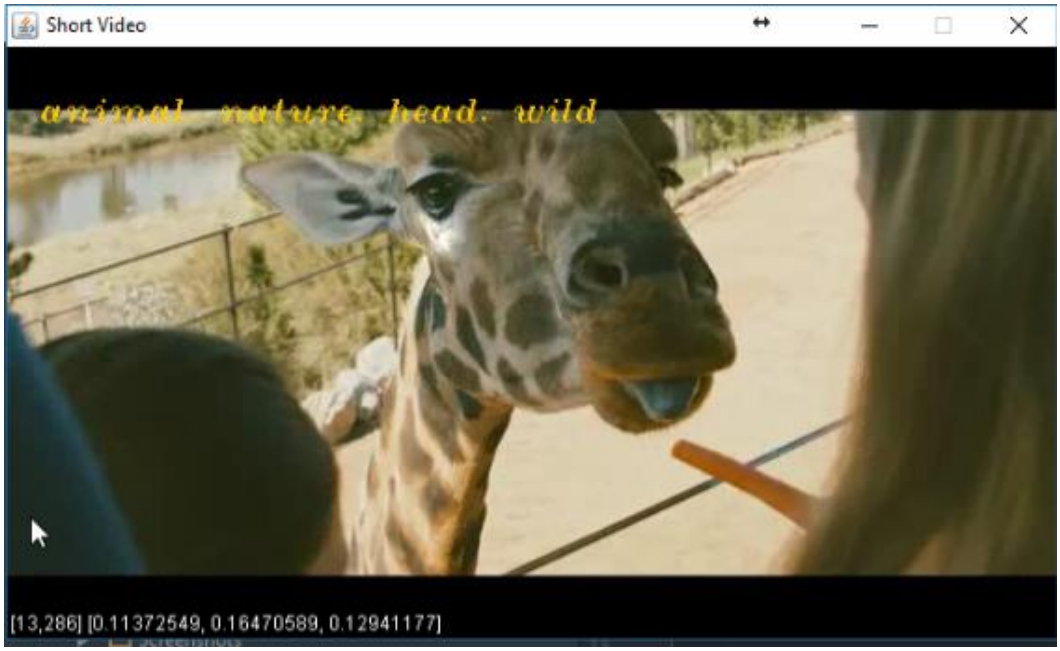

Fig. Key frame with relevant tags

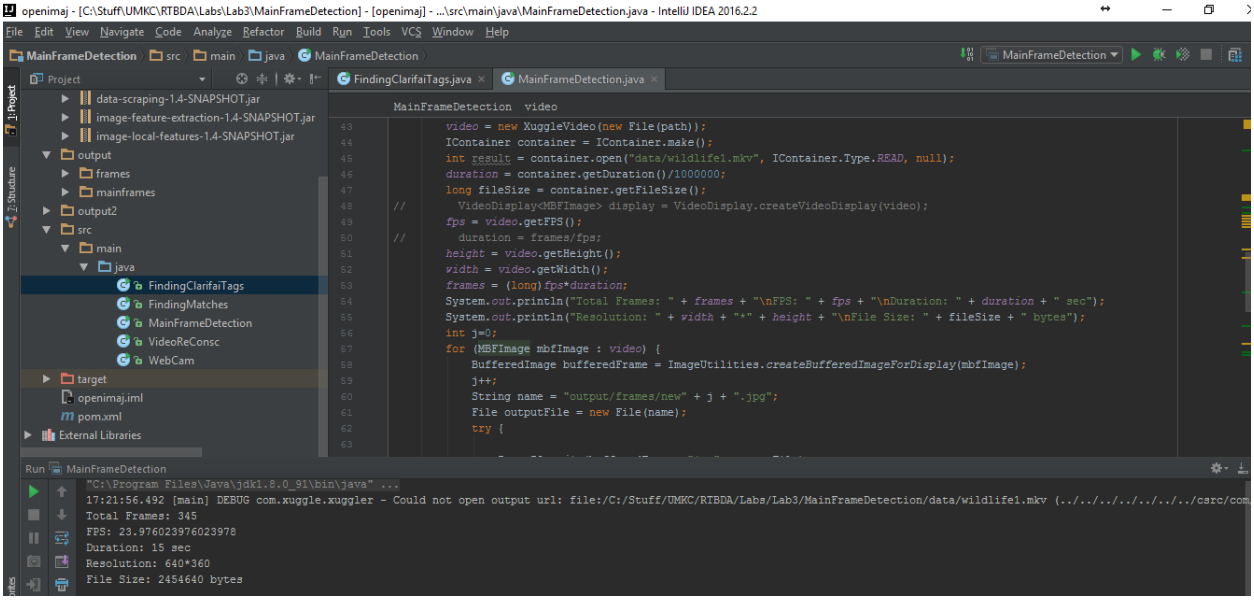

Fig. Implementation of logic in IntelliJ IDEA

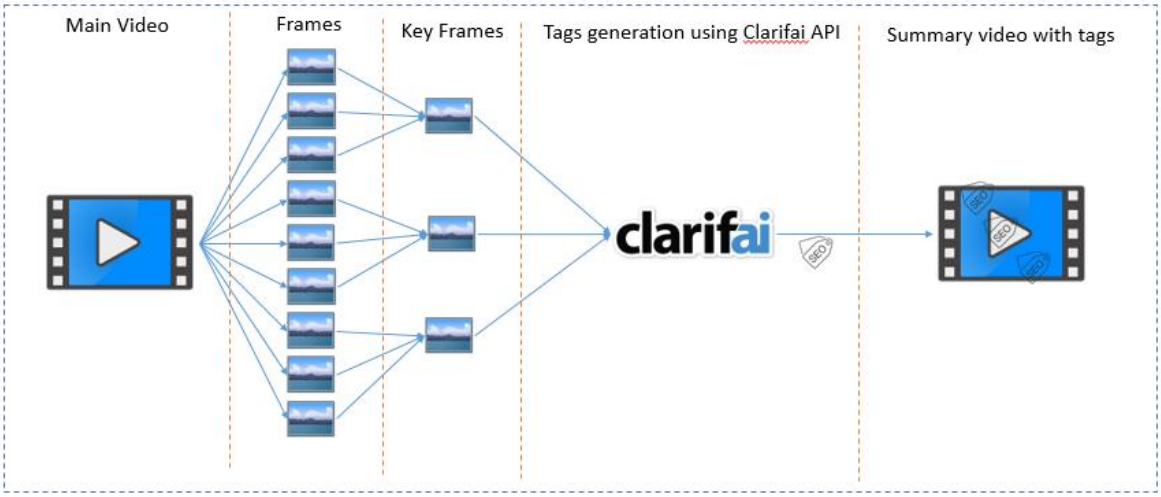

Fig. Workflow of key frame generation and adding tags to key frames using Clarifai API

Person face and full body detection in a video stream:

In this later stage, we first tried to detect the human faces in the input video stream. Later we extended this feature to detect different parts of human body (eyes, nose etc.) and in the final the full body of the person. The detected human body will be highlighted with a rectangular box around it as shown in the below screenshots.

Fig. Person body detection

Fig. Person body detection

Fig. Person body detection

Work completed

We've divided the increment 1 tasks in to three groups, first one is requirements gathering. In this, we've gone through various research papers and understand the basic steps to be followed in implementing the proposed system. We identified the advantages and performance improvements that we can achieve by using Apache Spark and Apache Storm in processing and analyzing the video frames. Each one from our team has participated actively and after several group meetings we concluded the system architecture mentioned in the earlier section.

Second one is, reading a video file and extracting the key frames from the video. Tej Kumar and Chaitanya worked on this task and they are able to successfully complete the task in less time. Later Sri Harsha and Priyadarsini used these extracted key frames in generating the tags for the objects in these frames using Clarifai API. We've also created a short video using these key frames and the generated tags. This helped us in understanding the framework and workflow of OpenIMAJ.

Third one is, detecting a person's face and the key points on his face (nose, mouth, eyes etc.) using HaarCascadeDetector class of OpenIMAJ framework. Tej Kumar and Chaitanya worked on these tasks. They are able to successfully annotate the detected faces with a rectangular box in a given video stream. Later Sri Harsha and Priyadarsini used these results to extend the functionality by detecting a person in the given video stream. We've tested various built in cascades available in OpenIMAJ (eyes, mouth, upper body, lower body, full body etc.,) and are finally able to highlight the persons with a rectangular box.

Time taken: 60 hrs/person

Contributions: Sri Harsha 25%, Priyadarsini 25%, Chaitanya 25%, Tej Kumar 25%.

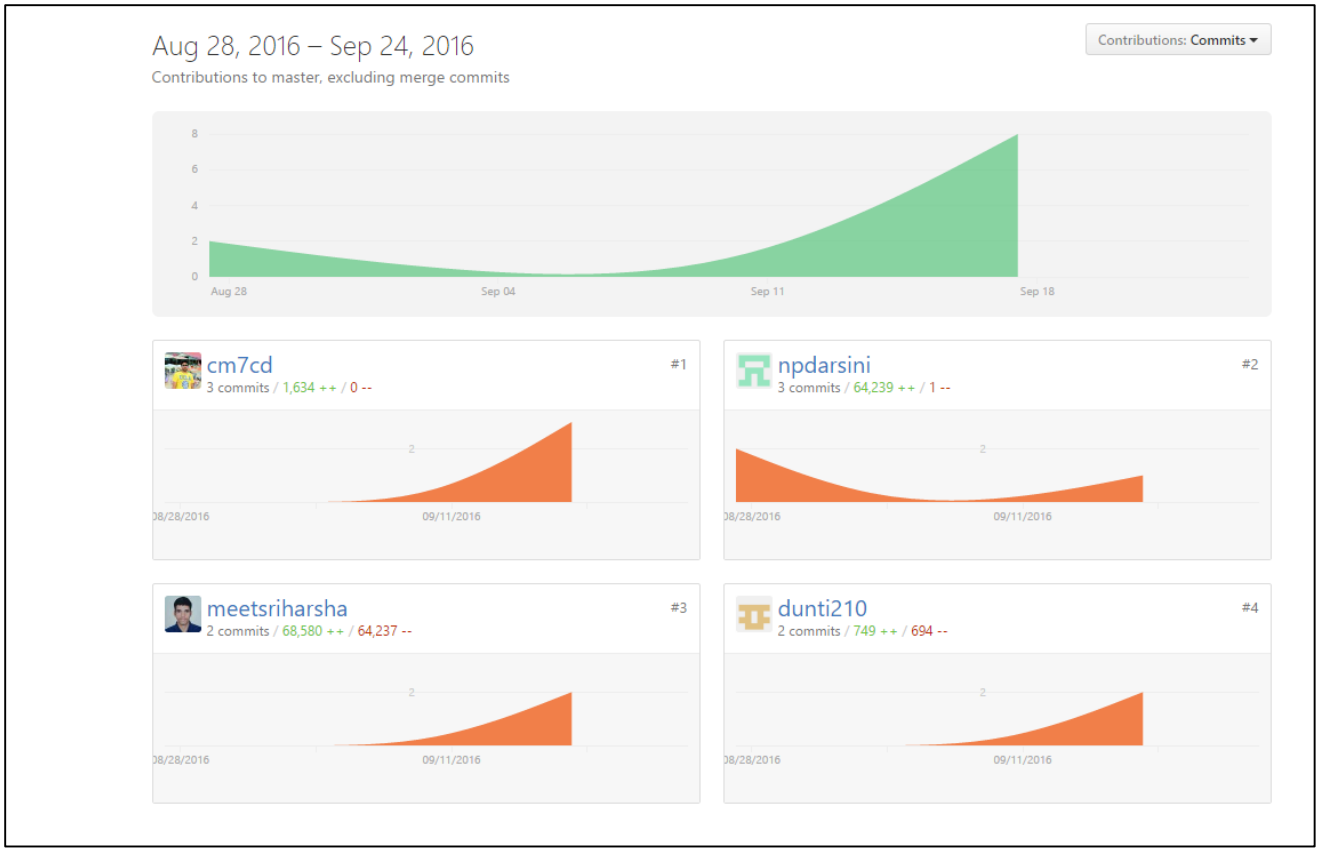

Fig. Project contributions graph in GitHub

Work to be completed

For the next increment our target is to implement an application to generate the training dataset of human faces feature vectors and store them in database. For this we need to understand and have a hands on experience of how Kafka, Spark communicates and processes the data. We will also try to implement a basic end to end Kafka-Storm application. Priyadarsini and Sri Harsha will work on generation of training dataset. Chaitanya and Tej Kumar will work on understanding and implementation of Kafka-Storm application. Time to be taken: 70 hrs/person.

Issues/Concerns

The cascades available with OpenIMAJ are not reliable in detecting a person. We got false positive results in some scenes of the input video.

Bibliography

- Face recognition using Laplacianfaces - http://ieeexplore.ieee.org/document/1388260/?arnumber=1388260
- Face recognition using eigenfaces - http://ieeexplore.ieee.org/document/139758/?arnumber=139758&tag=1
- Facial recognition techinques - https://en.wikipedia.org/wiki/Facial_recognition_system
- Computer Vision - http://opencv.org/
- OpenIMAJ Face Recognition Library - http://openimaj.org/openimaj-image/faces/dependencies.html
- Tags generation for a video frame - https://www.clarifai.com/