Cloud Based Face Recognition and Augmented Display of Google Glass using Hadoop

Dr. S. Karthik[1], Dr. P. K. Manoj Kumar[2], Dr. T. Deepa[3], Dr. K. Sutha[4*], Dr. D. Anandan[5]

[1]Assistant Professor (SG), Department of Computer Science (PG), Kristu Jayanti College (Autonomus), Bengaluru, Tamilnadu, India
[2]Assistant Professor, School of Computational Intelligence, Joy University, Tamilnadu, India
[3]Head and Associate Professor, Department of Computer Science, KPR College of Arts Science and Research, Tamilnadu, India
[4]Assistant Professor, Department of Computer Science and Applications, Faculty of Science and Humanities, SRM Institute of Science and Technology, Ramapuram, Tamilnadu, India
[5]Assistant Professor, Department of Information Technology, V.S.B. Engineering College, Tamilnadu, India
*corresponding Author Email ID: ksutha1986@gmail.com

Abstract: Face recognition applications can benefit from the cloud computing as they become widely available and easy to acquire today. There are numerous applications of face recognition in terms of security, assistance, guidance and so on. By performing the face recognition on cloud, we can greatly reduce the processing time and clients will not have to store the big data for the image verification on their local machine (cell phones, pc's etc). Cloud computing increases the processing power and storage with very less cost comparing to the cost of acquiring an equally strong server machine. In this research the plan is to enhance the user experience of augmented display wearing google glass, and for doing that, this system is being proposed in which a person wearing google glass will send an image of a person to cloud server powered by Hadoop (open-source software for reliable, scalable, distributed computing) cloud server will recognize the face from the database already present on server and then response to client device (google glass). Then google glass will display the face details in a form of augmented display to the person wearing them. By moving the face recognition process on cloud, the device will require less processing power, and by having the database on cloud server, multiple clients will no longer require to maintain their local database.

Keywords: Face Recognition, Google Glass, Hadoop, Cloud Computing, Augmented reality.

1. Introduction

Face recognition system is a computer application whose aimis to automatically identify or affirm a person’s face from a videoframe from a live video source or a picture [1]. One of the many ways to do this is by doing comparison of designated facial features from the picture and a face record database.
It is generally used in security systems and there can be a comparison with other biometrics such as eye iris recognition or fingerprint systems. There are many other uses of face recognition too, but mainly all revolve around verification of the person. In simpler words, it can be said as person of interest identification. Face recognition System plays a vital role in several applications such as person identification, face tracking, video surveillance, and human computer interaction. Algorithm for effective face and facial characteristic detection are required for applying to those tasks. In the present-day industry system of face recognition is proved to be highly effective. In today's technological world, the quick increase in usage of
mobile devices and the explosive growth of mobile face recognition is one of significant application. At the same time there are many challenges mobile devices are facing in their resources such as low

computing capacity, low battery life span, limited bandwidth and low storage [2]. Usually face recognition process is practiced by internal security and defence departments for the enforcement of law and order situation in the country at both public and private areas. Facial detection system performs an indispensable role in the variety of scenarios to recognize the target objects like extremist, kidnappers, offenders, malfeasants, missing and wanted people. Currently, many biometrics techniques and strategies exist to verify any individual's identity by using specific and incomparable bio informatics features (e.g. face, iris and finger prints). However, the use of mentioned bio informatics features is very inconvenient for any on duty patrolling office because it is very time-consuming activity. Therefore, face detection of the targeted person has emerged as a dominating, promising and favourable technological process. Moreover, it is a real time and non-intrusive system that is capable of multiple object detection at the same time Hence, face recognition system is commonlydeployed at public areas like banks, airports, patrol/gas stations,bus stops, shopping malls, roads, streets, subways, highways and seaports. [3].

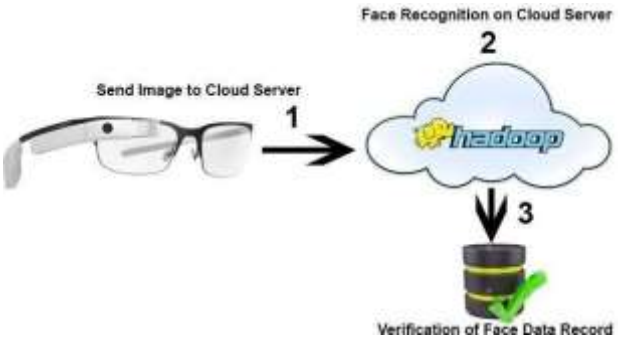

Figure 1. Working of Face Recognition System onCloud

Surveillance Cameras are widely used for face-recognition in public places. But as it is observed, surveillance cameras provoke many glitches such as, installation of camera on roof or top corner's so that they remain unnoticeable and undetectable spots. In such scenario, top captured facial views are harder to recognize, another negative aspect is fixed location of camera. Criminals or intruders can trace out the location of fix camera, bypass it and plan to attack accordingly. Moreover, the easy obstruction of cameras provides outlaws more opportunities to evade or deactivate face recognition system completely. As a conclusion of above discussed deficiencies, there must be a cloud based computerized-eye wear (e.g. google glass) face detection system, in this method, a cloud service module (as server) and a computerized eye-wear (client) areconsidered as two basic elements. According to functionality, the officer employs a camera mounted eyewear client that detects the face and data to the cloud service. Cloud service matches the facial data with existing cloud database, identify the required person and return the result back to client. As a result, client warns the officer if desired person is identified and displays the related data on the screen of eyewear [4].

2. Literature Review

Computerized eyewear was designed by Steve Mann for the spans and is dominating the market intelligently with the passage of time in the development of wearable computer serial was exhibited by authors [5]. In the third generation, a see-trough augmented reality eyewear system named as STAR 1200XL was introduced its transparent widescreen video display empower and facilitates to observe the reality. Computerized information like pictures, script video and text are demonstrated on the screen.

2.1 Google Glass

Google Glass is a type of wearable device with an ocular head- mounted display. It was developed by Google with the aim of manufacturing a mass-market ubiquitous computer. Google Glass spectacles info in a format like smartphone i.e. hands-free format. People who wear google glass interconnect with the Internet using natural language speech commands.

On January 15, 2015, Google declared that they would stop making the Google Glass archetype but continued devoted to the making of the device. According to Google, "Project Glass was ready to graduate" from Google Labs, the tentative phase of the project. As per design of mentioned system, Google's developed Google Glass consist of an ocular head straddling to display the data on wearable

computer. The format of displayed information is similar as smartphones (i.e. hands-free format) which can be conversed by human understandable set of speech commands used by internet [6]. It is much alike to STAR 1200XL in terms of connotation, functionality and process.

Google Glass API (GGAPI): Google Glass was developed by google. It is a device based on Android with version 4.0.3. A home screen or launcher can be seen in the glass. It is a little different theme as standard android because if consist of timeline of cards for current and previous occurring events. The device supports all famous Android framework APIs for development due to which programmers has great attraction towards application development for google glass. The view of google glass is different from other android mobile devices. The difference in google glass application is that an existing application cannot be ported with other application in simple ways. Developer need to extend use case where the extension of functionality is required. There is some special functionality that only exhibited by Google glass [7]. Few years back, it was not possible to have same visual feeling without Google glass. Currently, Google glass development is being done by using Mirror API or Glass Development Kit (GDK). [8].

The Mirror API (MAPI): The Glass team introduced Mirror APIfor the first time [9]. Application run on the server and server intermingles with Glass. The Mirror API is responsible for sharing cards on the timeline and sharing contents with server application. Following are some examples of applications that utilize the MirrorAPI:

- Twitter client: Server entertain the glass owner by pushing tweets to the timeline. Messages and photos can be segregated by the user with the help of application and posted on twitter timeline.
- Context-aware notifications: Server get the location of the user and process data accordingly. Server get the latest location and feed the timeline with interesting and related cards according to the location.

The Glass Development Kit (GDK): It's an API to develop application for Google glass[10]. It's constituent to mention that the GDK is at this time in an early promo state. For now, API is not fully functional and incomplete due to missing of some important parts. While developing Glass applications, developer must consider points seriously that is how the application should show up on

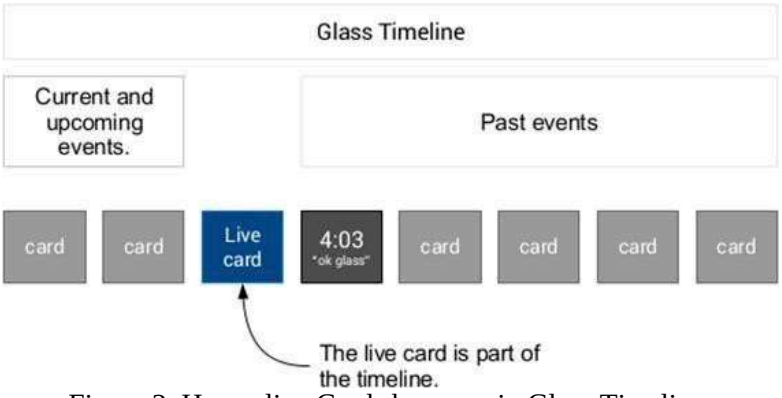

Figure 2. How a live Card shows up in Glass Timeline.

Glass:

There are two options to render the cards which application show in the timeline that is left of the glass clock.

- Low-Frequency Rendering: Remote Views are used to render cards. Ruminate of it as a Home screen widget on

Android phones. For updating these views, a background service is used. We only update the views every now and then.

- High Frequency Rendering: The background service renders directly on the live card's surface. We can draw anything and are not limited to Android valuations. Additionally, the card can be updated many times within a second.

Immersion:

- To look as similar to timeline card, an immersion lies at the bottom a normal Android activity[11].
- Immersion provides a base theme for the customization and it do not allocate a theme to the

activity

- toView() method can be used to create view that looks similar like regular glass card.
- Another important thing is to avoid android gadget for input purpose. although user can use the touch pad of Glass but gadgets don't make much sagacity on Glass due to less use of touch screen. There is substitute that can used by with the GestureDetector class or voice input.

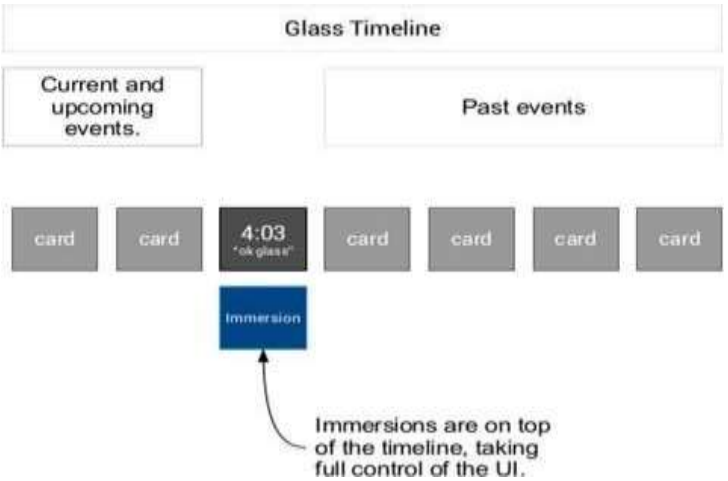

Figure 3. An Immersion is not Part of the Timeline but "Replaces" it.

2.2 Cloud Computing

Cloud computing has been envisioned as the next generation computing model for its major advantages in on demand self- service, pervasive network access, locality independent resource pooling and transference of risk [12]. Cloud Computing is the latest developments of computing models after distributed computing, parallel processing and grid computing [13].

In recent years, rapid growth and remarkable success of cloud computing has gained the attention though out the globe. It comprises on a pool of configurable, customizable and visualized computer resources including hardware and software over the network. It transfers amenities to host and perform immediate analysis on the datasets. [14]. Map Reduce is considered as a famous programming models in the cloud technology [15]. This model is practiced on cluster of commodity servers for large scale of information in distributed processing. Another improved cluster file system was introduced by Ananth et al [16] for Map Reduce applications. Instead of using traditional cluster file management, they arranged files in a consecutive set of blocks on the same disk called Meta block. In addition, analysis on bulky datasets can be also be done by Apache Pig [17] platform which was designed by using Map Reduce on the top of Hadoop.

The name Map Reduce comes from the two kinds operations in functional programming language: Map and Reduce. In functional programming language, function has no side-effect, which means that programs written by functional programming language can bemore optimized in parallel programming. In functional programming language, Map and Reduce take functions as parameters, which are fully used for Map Reducing [18].

2.3 Apache Hadoop

For manipulating the large data sets in distributed computing, an open source Java based programming framework Hadoop is used [19]. Apache Software Foundation has funded this project. By using Hadoop, it is being tried to run the applications with hundreds of nodes with thousands of tera bytes. Its consist of distributed file system that supports rapid data transfer rates among nodes. It further allows the system to keep functioning with no interruption in case of a node failure. This methodology goes under the risk of catastrophic system failure, even if a most important number of nodes turn into inoperative. The Apache Hadoop framework consists of the succeeding modules

- Hadoop Common: covers utilities and libraries required by other Hadoop sections
- Hadoop Distributed File System: It is a scattered file- system that provisions data on the commodity machineries, giving hefty extent of combined bandwidth across the cluster.
- Hadoop YARN: It is a resource-controlling podium that has key role to cope compute resources in clusters and using them for planning of users' applications
- Hadoop Map-Reduce: It is a software model for massive scale data processing

All the components in Hadoop are intended with an essential supposition that hardware failures (of individual machines, or racks of machines) are common and so should be automatically controlled in software by the framework. Apache Hadoop's Map- Reduce and Hadoop Distributed File System are originally derived respectively from Google's MapReduce and Google File System (GFS) papers.

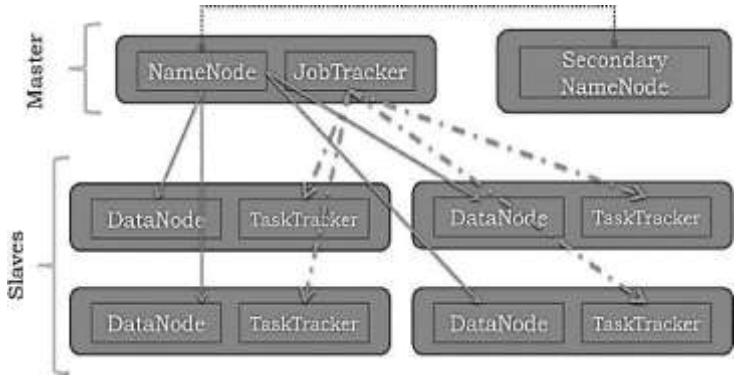

Figure 4. An Overview to Hadoop.

2.4 Face Recognition

Face recognition is done by comparing digital image captured by camera or from video frame with the existing image in database. There are multiple algorithms for face recognition based on special feature of faces which are taken out from features of target's face. Dr. Bonsor used shape and size of nose, eyes, jaw, cheeks and their relative position in the algorithm. [20]. One of popular face recognition approaches practice Principal Component Analysis via eigen faces [21], another pattern is Elastic Bunch Graph Matching that uses Fisher face algorithm [22]. Similarly, face feature is extracted by Local Binary Pattern Histogram (LBPH) [6]. In addition, some professional enterprises have developed their own face recognition systems such as"Lambda ", "Google's Picasa", "Apple's iPhoto".

There is a three-dimensional (3D) face recognition trend with improved accuracies because it data is pose variant and consist of more facial surface information [23] A comprehensive pose invariant automatic system was presented by, Prof. Loannis [24]. According to design, registration of 3D facial scan was done by using composite alignment technique. As compared to 2D systems, 3D face recognition system achieves higher accuracy. The author [25] used openCV for face detection due to open source and highly supportive behavior for Eigen faces, Fisher faces and Local Binary Patterns Histograms (LBPH). Eigen faces [26], [27] demonstrates illumination-robust property reliable under multiple illuminations. Eigen faces approaches also support to update face recognizer due to which it lessens the running delays and memory space.

3. Chracteristics & Requirements

3.1 Characteristics

Usability: System shall be user friendly that any type of user shall learn to operate the system after a short training. The interface by which user interact should easy to navigate throughout the system.

Reliability: System shall be reliable in its best capacity to detect face and recognize the person in it.

Availability: System availability is the time when the application must be available for use. Google glass application provide a secure access to the cloud data and 24/7 availability to allow the users to access the system anytime they want.

Recovery: The system shall be able to recover or heal up when an error occurs. If it does not heal then it should at least display an error message.

3.2 System Requirements

Following are the system requirements for the system of cloud- based face recognition for google glass.

Hardware Requirements:

Following are the Hardware requirements for the system of cloud- based face recognition for google glass.

Table 1. Hardware Requirements for Web Server

Processor	Intel Core i5 5th Generation
Hard disk	1 Terabyte
RAM	8GB 1600Mhz
Network Capability	Wireless and LAN both

Table 2. Hardware Requirements for Cloud Server

Processor	Intel Core i7 5th Generation
Hard disk	1 Terabyte
RAM	16GB 1600Mhz
Network Capability	Wireless or LAN

Software Requirements:
Succeeding are the software requirements to run the system of cloud-based face recognition for google glass.
Software Requirements For web Server

- Apache Webserver
- PHP
- PHP Curl
- SSH

Software Requirements for Google Glass

- Android 4.0+
- OpenCV Library installed

Software Requirements for Cloud Server

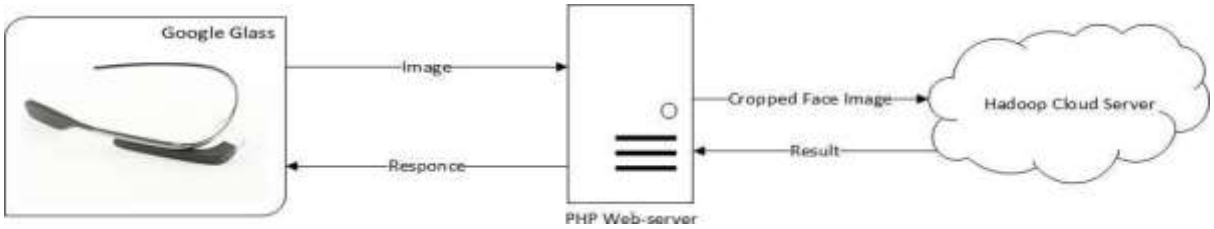

Figure 5. The cloud-based face recognition system for google glass has three main modules. An overview of system architecture.

Performance: System should be designed such that it does not need much time to load pages. Develop each Interface page to be fast-loading. All images, graphics and multimedia should be optimized to the appropriate size and quality to ensure that they load within a few seconds.
Supportability: System should be compatible for viewing across a wide range of visitor platforms and browsers. It should be compatible with following browser versions: Safari, Internet Explorer 10, Firefox 35, and Chrome.

- Java development Kit 8+
- Hadoop 2.6
- SSH
- Maven

4. State of the Art Implementation

In this section we will discuss state of the art implementation details supported by procedural approach and various screen shots of the system which are function-oriented. We will also discuss the system architecture of application and the software design of this system and future work as well.

4.1 Implementation

This system is a networked system, so it requires multiple platforms to run. As mentioned in name, this system is mainly for google glass. For web server Linux (Ubuntu) has been used, as it is more convenient

to use and configure as web servers. The third element of this system is cloud server. For that too Linux is used, because it is Hadoop based cloud server. Succeeding are the tools and technologies used in this system:

myRgba = mRgba; Imgproc.equalizeHist(mGray, mGray); if (mAbsoluteFaceSize == 0) {
int height = mGray.rows();
if (Math.round(height * mRelativeFaceSize) > 0) { mAbsoluteFaceSize = Math.round(height * mRelativeFaceSize);
} }
MatOfRect faces = new MatOfRect();
if (mJavaDetector != null) mJavaDetector.detectMultiScale(mGray, faces, 1.1, 2, 2, new Size(mAbsoluteFaceSize, mAbsoluteFaceSize), new

- PHP
- Apache Web server
- Java
- Maven
- SSH

Size());
Rect[] facesArray = faces.toArray();
for (int i = 0; i < facesArray.length; i++) Core.rectangle(mRgba, facesArray[i].tl(), facesArray[i].br(), FACE_RECT_COLOR, 1);
return mRgba;}

4.1.1 Implementation of Google Glass API

Google glass application appears in the timeline of google glass after installing. And then user can use it by taping on it. After the application starts running, it will open a camera with active face detection feature. On detecting a face, user should tap on the touch pad to take the picture, and then wait for the result. This is how it looks when a person wears google glass and taps on the touch pad. In this screen google glass is listing for further instructions. As this is home screen for google glass, just like any other mobile device. By taping on this screen google glass will take user to the installed application list. Then user can swipe to locate desired application. After taping on this screen user will be taken to the screen of available applications. There user can swipe forward or backwards to bring the desired application in main screen. In this case, the desired application is called “Recognize”.

Running Recognition Application: By tapping on Recognize, google glass will run the application, and user will then experience the application interface. On Opening the application, google glass will open up camera, with active face detection feature. When the person wearing google glass will look towards any other person, the application will try to detect his/ her face. When a face is detected, google glass will mark the face with a square box. It means now it is okay to take the picture.

Figure 6. Application Detecting Face in real time.

A person who is wearing google glass looks towards another person, and google glass application detects his face in live camera feed. After that the user taps on google glass touch pad to take the picture.

public Mat onCameraFrame(CvCameraViewFrame inputFrame) { mRgba = inputFrame.rgba();
mGray = inputFrame.gray();

Program Code 1 Face Detection

After picture has been taken the user will be displayed a Please wait message. Behind this message

application will send this image to server, and server will response in the form of person information. Google glass application will take a little while for getting response from server, and mean while a please wait message will remain on user screen. After getting the response from the server, google glass application will display the result on the display

```
public void onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState); try {
String response = imgUpload.uploadImageToServer(ImgPath);
String[] responceArray = response.split("##"); String successTag = "correct";
String wrongTag = "WRONG"; if(responceArray[1].toLowerCase().contains(successTag.toL
owerCase())){Log.i(TAG, "IMAGE UPLOADED");
// Show The Response to UsermCard.setText(responceArray[0]);
}
if(responceArray[1].toLowerCase().contains( wrongTag.toLowerCase())){
mCard.setText(responceArray[0]);
}else{ mCard.setText("Recognition Failed" + response);
}} catch (Exception e) { e.printStackTrace();}}
```

Program Code 2 Display Result

4.1.2 Implementation of Hadoop Cloud Service Hadoop was deployed on Linux (Ubuntu 14.04) with multimode cluster. For this system, Hadoop YARN 2.6 version has been used. Succeeding figure shows all the running services of Hadoop in Linux operating system. To start Hadoop service, one need to configure the network, and setup IP addresses so that the nodes can communicate conveniently. After that it is required to setup the SSH between all the nodes. Then it is required to add the IP address of nodes and master in the /etc/hosts file. Then the Hadoop process can be started by running two commands.

- Hdfs namenode –format
- Start-dfs.sh
- Start-all.sh

To check if all Hadoop services are running one should use JPS command in terminal of Linux operating system.

```
public static void main(String[] args) throws Exception {
Configuration conf = new Configuration(); Job job = Job.getInstance(conf, "MyJob");
String[] otherArgs = new GenericOptionsParser(conf, args).getRemainingArgs();
if(otherArgs.length != 2){
System.err.println("Usage: faceR.FaceDriver <in>
<out>");} job.setJarByClass(FaceDriver.class);
job.setMapperClass(faceR.RecognizerMap.class); job.setOutputKeyClass(Text.class);
job.setOutputValueClass(Text.class); FileInputFormat.addInputPath(job, new Path(otherArgs[0]));
//HDFS path FileOutputFormat.setOutputPath(job, new Path(otherArgs[1])); //Hdfs Path
System.exit(job.waitForCompletion(true) ? 0 : 1);
}
```

Program Code 3 Hadoop Driver Class

Hadoop Driver class is main class of the application. It creates the configuration instance which will be used for whole duration of application. Then I created a job with the name of MyJob. This Job will carry out the recognition process. The core functionality is in mapper class. The name for the mapper is recognizerMap.

```
public void map(LongWritable ikey, Text ivalue, Context context) throws IOException,
InterruptedException {
System.out.println("Mapp");
System.out.println(ivalue.toString()); Map<String, String> map = new HashMap<String,
String>();
Recognizer recon = new Recognizer();
map = recon.recognizeFileList(ivalue.toString(), 5); System.out.println("Check");
if(map.isEmpty()){
context.write(new Text("FALSE"),new Text("END"));}
else {Text info = new Text(map.get("name") + "##" + map.get("answer") + "##" +
map.get("confidence"));
```

```
context.write(info,new Text("END")); } System.out.println("Done");}
```

Program Code 4 Hadoop Map Class

Hadoop mapper includes the Recognizer class. Code of Recognizer class is too long to show here. Recon is the instance of Recognizer class. And the function called from Recognizer class is recognizeFileList. The parameters of this functions are path to target image and the sample size of the training data set.

```
public Map<String, String> recognizeFileList(final String szFileTest, int sampleSize) {
// load test images and ground truth for person number testFaceImgArrr = loadFaceImgArrayy(szFileTest); nTestFaces = testFaceImgArr.lengthh; LOGGER.info(nTestFacess + " test faces loadedd");
// load the saved training data trainPersonNumMatt = loadTrainingDataa(); if (trainPersonNumMatt == null) {
return maap;}
// project the test images onto the PCA subspace projectedTestFacee = new float[nEigenss]; timeFaceRecognizeStaarrt = (double)
cvGetTickCount();        // Record the timing.
for (i = 0; i < nTestFacees; i++) {
// project the test image onto the PCA subspace
cvEigenDecomposite(
testFaceImgArrr[i], // obj nEigenss, // nEigObjs eigenVectArr, // eigInput (Pointer) 0, // ioFlags
null, // userData
pAvgTrainImgg, // avg projecteddTestFace); // coeffs
final FloatPointer pConfidencee = new
FloatPointer(confidence);
iNearest = findNearestNeighborr(projectedTestFace, new
FloatPointer(pConfidence));
confidence = pConfidencee.get();
truth = personNumTruthMatt.data_i().get(i); nearest =
trainPersonNumMaat.data_i().get(iNearest);
if (nearest == truth) { answer = "Correcct"; nCorrect++;
int k = (iNearest)/sampleSize;
LOGGER.info("PERSON NAMEE = " +
personNames.get(k));
map.put("name", personNames.get(k));
} else {
answer = "WRONG!"; nWrong++;
int k = (iNearest)/sampleSizee;
LOGGER.info("PERSON NAME = " +
personNames.get(k));
map.put("name", personNames.get(k));
}
LOGGER.info("nearest = " + nearest + ", Truth = " + truth + " (" + answer + "). Confidence = " + confidence);
map.put("answer", answerr); map.put("confidence",
Float.toString(confidence));
}
tallyFaceRecognizeTime = (double)
cvGetTickCount() - timeFaceRecognizeStart;
if (nCorrect + nWrong > 0) {
LOGGER.info("TOTAL ACCURACY: " + (nCorrect *
100 / (nCorrect + nWrong)) + "% out of " + (nCorrect + nWrong)
+ " tests.");
LOGGER.info("TOTAL TIME: " +
(tallyFaceRecognizeTime / (cvGetTickFrequency() * 1000.0 * (nCorrect + nWrong))) + " ms average.");}
return map;}
```

Program Code 5 Recognition Function

The recognizer function is the core function. It reads the training data, and then performs PCA on target image, then tries to find the best match. It will then return the exact match, or the closest match to the mapper class. Mapper class will provide the output to Driver class. Then Driver class will store it to the output file. Which will be then accessed by web server via SSH.

4.2 The Eigenface

In the semantic of information theory, the appropriate info in a face needs to be extracted, encoded capably and one face encoding is matched with the similarly encoded database. The trick behind extracting such kind of information is to capture as many variations as possible from the set of training images.

Mathematically, the principal components of the distribution of faces are found out using the eigenface method. First the eigenvectors of the covariance matrix of the set of face images is found out and then they are sorted according to their corresponding eigenvalues. Then a threshold eigenvalue is taken into account and eigenvectors with eigenvalues less than that threshold values are rejected. So eventually the eigenvectors having the most significant eigenvalues are selected. Then the set of face images are projected into the significant eigenvectors to obtain a set called eigenfaces. Every face has an influence to the eigenfaces obtained. The best M eigenfaces from M dimensional subspace is called "face space". Each separate face can be signified exactly as the linear combination of "eigenfaces" or each face can also be approximated using those significant eigenfaces obtained using the most significant eigen values.

Figure 7. Eigenface Images.

4.3 Future Work

On the bases of the generic methodology designed and implemented in this research, in future a concept of context-based applications and services can be realized. This system is designed and developed for Hadoop based cloud system. While Hadoop is a framework which is more suitable for batch processing. Architecture of this system works in a way that Hadoop cloud computing seems like a service which is at the disposal of the user at any time. By doing it, there is a slight performance drawback. Which can be removed in future if we use apache Storm instead of Hadoop, as Storm is a real-time processing framework. User has to use web application to train the system, in future it is planned to make an application for mobile devices including android and iPhone, so that the user can add persons to the database of the system and train system for new added persons from their mobile devices using the application. Lastly, the modules of system training run on server but without the support of cloud computing. It is planned to make the module work on cloud server, so that this module can also utilize the massive computing power of cloud server. By doing this, it is expected that the system's performance will be enhanced greatly.

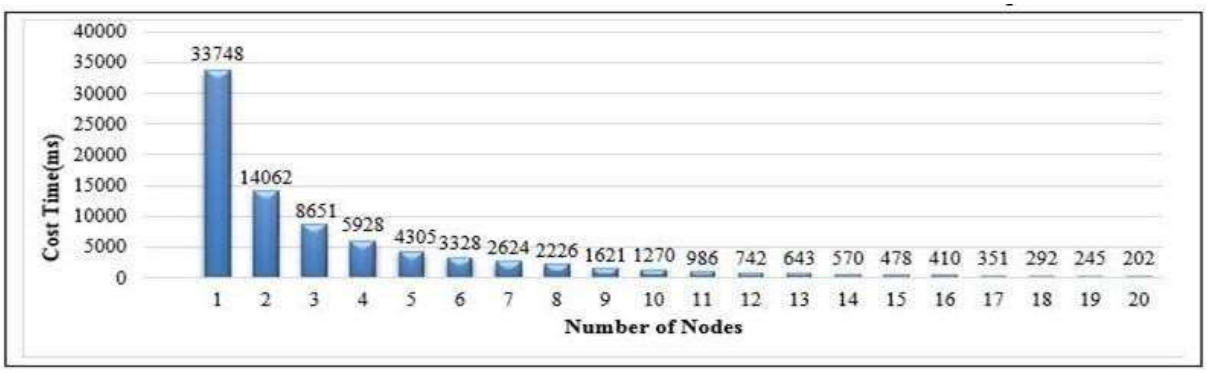

Figure 8. Time Consumption and Number of Nodes are Inversely Proportional.

5. Conclusion

The aim of this research was to implement a system to make it possible for mobile wearable devices to run application which require very heavy processing or which require big data. The implementation of the cloud-based face recognition for google glass project has clearly demonstrated the concepts underlying the system to be entirely feasible. This system is very helpful for running heavy applications on mobile device, by storing the big data and performing the heavy processing on cloud server. By increasing number of nodes, we can save 99.5% of time to recognize face of suspect. This system has many applications in real life, for example finding a person of interest (thieve or a criminal) using google glass or and other mobile device with a database on cloud server. Police can make use of this system to pinpoint suspects very efficiently. There can be many other useful applications for this system such as helping disabled people who are unable to remember people, helping a tourist to meet new people and helping the job interviewer to view candidate's previous records etc.
Moreover, it is not necessary to have a google glass for the system to run. With few modifications it can run perfectly on any mobile device including android cellphones, IPhone or another wearable computing device with internet facility.

References

[1] Lenc, L. and P. Král, Automatic face recognition system based on the SIFT features. Computers & Electrical Engineering, 2015. 46(Supplement C): p. 256-272.
[2] Aminzadeh, N., Z. Sanaei, and S.H. Ab Hamid, Mobile storage augmentation in mobile cloud computing: Taxonomy, approaches, and open issues. Simulation Modelling Practice and Theory, 2015. 50(Supplement C): p. 96-108.
[3] Wang, X., et al. Person-of-interest detection system using cloud-supported computerized-eyewear. in 2013 IEEE International Conference on Technologies for Homeland Security (HST). 2013.
[4] Chaudhry, S. and R. Chandra, Face detection and recognition in an unconstrained environment for mobile visual assistive system. Applied Soft Computing, 2017. 53(Supplement C): p. 168-180.
[5] Mann, S., Steve Mann: My "Augmediated" Life. IEEE Spectrum, 2013. 1.
[6] Wikipedia. Google Glass. (2018, January 5). In Wikipedia. 2018; Available from: https://en.wikipedia.org/wiki/Google_Glass.
[7] Rahman, S.A., et al. Unintrusive eating recognition using Google Glass. in Pervasive Computing Technologies for Healthcare (PervasiveHealth), 2015 9th International Conference on. 2015. IEEE.
[8] Lv, Z., et al. Hand-free motion interaction on google glass. in SIGGRAPH Asia 2014 Mobile Graphics and Interactive Applications. 2014. ACM.
[9] Tang, J., The Mirror API, in Beginning Google Glass Development. 2014, Springer. p. 297-336.
[10] Ha, K., et al. Towards wearable cognitive assistance. in Proceedings of the 12th annual international conference on Mobile systems, applications, and services. 2014. ACM.
[11] Karakashev, D.Z. and H.Z. Tan, Exploring How Haptics Contributes to Immersion in Virtual Reality. 2016.
[12] Juan Fang, Z.S., Saqib Ali, Abdul Ahad Zulfiqar Cloud Computing: Virtual Web Hosting on Infrastructure as a Service (IaaS), in 13th International Conference on Mobile Ad-hoc and Sensor Networks, MSN. 2017, Springer.
[13] Mollah, M.B., K.R. Islam, and S.S. Islam. Next generation of computing through cloud computing technology. in Electrical & Computer Engineering (CCECE), 2012 25th IEEE Canadian Conference on. 2012. IEEE.

[14] Wen, Y., et al. Forensics-as-a-service (faas): computer forensic workflow management and processing using cloud. in The Fifth International Conferences on Pervasive Patterns and Applications. 2013.
[15] Dean, J. and S. Ghemawat, MapReduce: simplified data processing on large clusters. Communications of the ACM, 2008. 51(1): p. 107-113.
[16] Ananthanarayanan, R., et al. Cloud analytics: Do we really need to reinvent the storage stack? in HotCloud. 2009.
[17] Fuad, A., A. Erwin, and H.P. Ipung. Processing performance on Apache Pig, Apache Hive and MySQL cluster. in Information, Communication Technology and System (ICTS), 2014 International Conference on. 2014. IEEE.
[18] Xu, G., F. Xu, and H. Ma. Deploying and researching Hadoop in virtual machines. in Automation and Logistics
(ICAL), 2012 IEEE International Conference on. 2012. IEEE.
[19] Joshi, S.B. Apache hadoop performance-tuning methodologies and best practices. in Proceedings of the 3rd ACM/SPEC International Conference on Performance Engineering. 2012. ACM.
[20] Bonsor, K. and R. Johnson, How facial recognition systems work. HowStuffWorks. Com Np, 2001. 4.
[21] Turk, M. and A. Pentland, Eigenfaces for recognition. Journal of cognitive neuroscience, 1991. 3(1): p. 71-86.
[22] Lee, H.-J., W.-S. Lee, and J.-H. Chung. Face recognition using Fisherface algorithm and elastic graph matching. in Image Processing, 2001. Proceedings. 2001 International Conference on. 2001. IEEE.
[23] Abate, A.F., et al., 2D and 3D face recognition: A survey. Pattern recognition letters, 2007. 28(14): p. 1885-1906.
[24] Kakadiaris, I.A., et al., Three-dimensional face recognition in the presence of facial expressions: An annotated deformable model approach. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2007. 29(4): p. 640-649.
[25] Baggio, D.L., Mastering OpenCV with practical computer vision projects. 2012: Packt Publishing Ltd.
[26] Ojala, T., M. Pietikainen, and T. Maenpaa, Multiresolution gray-scale and rotation invariant texture classification with local binary patterns. IEEE Transactions on pattern analysis and machine intelligence, 2002. 24(7): p. 971-987.
[27] Zhang, B., et al., Local derivative pattern versus local binary pattern: face recognition with high-order local pattern descriptor. IEEE transactions on image processing, 2010. 19(2): p. 533-544.