Face Detection and Face Recognition in Android Mobile Applications

Octavian DOSPINESCU[1], Iulian POPA[2]
[1] Faculty of Economics and Business Administration, AL.I.Cuza University, Iasi, Romania
[2] Pentalog, Iasi, Romania
doctav@uaic.ro, iulyanpopa@gmail.com

The quality of the smartphone's camera enables us to capture high quality pictures at a high resolution, so we can perform different types of recognition on these images. Face detection is one of these types of recognition that is very common in our society. We use it every day on Facebook to tag friends in our pictures. It is also used in video games alongside Kinect concept, or in security to allow the access to private places only to authorized persons. These are just some examples of using facial recognition, because in modern society, detection and facial recognition tend to surround us everywhere. The aim of this article is to create an application for smartphones that can recognize human faces. The main goal of this application is to grant access to certain areas or rooms only to certain authorized persons. For example, we can speak here of hospitals or educational institutions where there are rooms where only certain employees can enter. Of course, this type of application can cover a wide range of uses, such as helping people suffering from Alzheimer's to recognize the people they loved, to fill gaps persons who can't remember the names of their relatives or for example to automatically capture the face of our own children when they smile.

Keywords: *Face Recognition, Face Detection, Mobile Applications And Face Detection, Mobile Applications And Face Detection*

1 Introduction

People have always had the ability to recognize and distinguish two different faces, while computers have begun to show the same ability not very long ago. According to [1] by the mid-1960s, researchers began the work on the recognition of human faces using the computer.

Lately robust facial recognition systems are becoming more useful, being helpful in various fields such as terrorism and crime fighting and different user authentication in real or virtual spaces for better security. But despite the amazing innovations and developments in which all take part, the extent of a person from a photograph by comparing it with other images previously saved in a database it is still problematic and a very complex topic. This is mainly due to the variety of human faces depending on the area and the different conditions in which they can be, such as camera performance, facial expressions, lighting, makeup, glasses and more.

Most times, a very simple variation of these conditions affect the accuracy of the systems through which the facial recognition is performed, mainly when using highly populated databases. This is the main cause for facial recognition systems are not yet widely used.

Facial recognition is usually used in security systems and can be compared to other biometric techniques such as fingerprint or iris recognition systems [2]. Also, facial and form recognition could help marketers in having a better impact on their clients because the visual aspect is one of the classic three channels: visual, auditory and kinaesthetic [3]. Some facial recognition algorithms identify faces by extracting landmarks or features of an image with the subject's face, which is nothing more than a set of pixels ordered by a particular pattern [4].

Each human face has many distinctive landmarks, different deck peaks that make up facial features. Each face has approximately 80 nodal points of which the following are used for facial recognition [5]:

- Distance between the eyes;
- Width of the nose;
- Depth of the eye sockets;

DOI: 10.12948/issn14531305/20.1.2016.02

- The shape of the cheekbones;
- The length of the jaw line.

Before the actual process of recognition is necessary to create image galleries. From the perspective of facial recognition, the gallery is a set of models serving as a reference for biometric matching process [6].

The following are the stages of facial recognition [7]:

1. *The image capture* is performed using a video or photo camera. The camera that is used to capture the image is more powerful so the recognition will be much more accurate.
2. *The face detection* is usually a fairly complex process because an image has almost always a background or other faces. Thus, the system will try the standardization of the photo so it has the same characteristics as previously stored images in the gallery.
3. *The extraction of features and the generation of a model* is a mathematical representation named biometric reference that will be the recognition substance.
4. *Comparing the models* is the process by which the biometric reference is compared to the other models of familiar faces in the gallery.
5. *Declaring identity* is establishing the kinship between two references, but which is often carried out by the human factor.

2 Mathematical Theories and Techniques for Facial Recognition

To achieve facial recognition there are some complex mathematical, geometry and photometry algorithms that support the processing and calculations that are made on a photo. First algorithms that were used to process facial recognition were based only on geometry, with which we only identified similarities between the main features as the position of the head, eyes, nose or mouth.

A Cartesian plan is a plan made by perpendicular lines that are called lines. If we have two points with coordinates A (xA, yA) and B with coordinates (xB, yB) the calculation of the distance between these two points is.

$$\mathbf{AB = rad((xA - xB)^2 + (yA - yB)^2)}$$

In any triangle the law of cosines says:

$$\mathbf{c^2 = a^2 + b^2 - 2ab\cos(\gamma)}$$

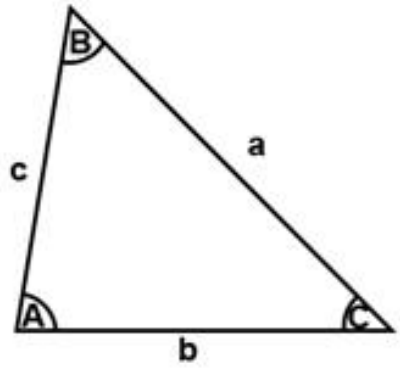

Fig. 1. A triangle example

Which means that we also can get from the previous equation the angle:

$$\mathbf{\gamma = \arccos([a^2 + b^2 - c^2] / 2ab)}$$

These equations will be very useful for the three distances from one face [8]: distance between two eyes, distance from the mouth to the left eye and the distance from the mouth to the right eye. Then we can calculate the three angles formed by the three lines of the triangle based on the previously calculated distances.

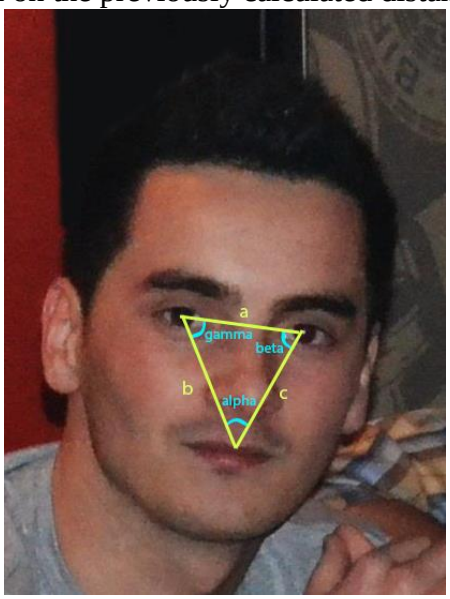

Fig. 2. The law of cosines on a face

The Principal Component Analysis (PCA) is one of the oldest and most used of the techniques of multivariate analysis. In general this algorithm extracts the relevant information from an image and encode them as efficient as possible by studying variations in values so that it can find to build correlations with other variables [9]. Mathematically the algorithm

DOI: 10.12948/issn14531305/20.1.2016.02

calculates the eigenvectors of the covariance matrix of an image containing a face. Each picture in a set of images contributes to a vector, characterizing the variation of these images. Representing these vectors is called set of faces "eigen" (eigenfaces), each face being represented by a linear combination of "eigen" faces.

The algorithm can be simplified into the following simple steps:

- Purchasing a database of faces, calculating the combinations of "eigen" faces required for subsequent recognition, a step which is also called the training algorithm;
- When a new image is found, it is calculated its data set;
- Determining whether the image is a face or not;
- Finally, determine if the new face corresponds to a known one or not.

"Eigenfaces" has a somewhat holistic approach to make facial recognition. Eigenfaces approach maximizes total dispersion, which can lead to problems if the variance is generated by an external source, because the components with a maximum variation across all classes are not necessarily useful for classification.

Local Binary Patterns Histogram Method takes a different approach than the method Eigenfaces. In LBPH each image is analysed independently, while "eigenfaces" method looks at the data set as a whole [10].

LBPH method is somewhat simpler in the sense that we characterize each image data set locally and when provided a new unknown picture, we perform the same analysis on it comparing the results for each of the images in the data set.

3 Android Resources Used In Face Detection and Face Recognition

Nowadays, every component and feature tend to be integrated in a complex software architecture [11]. Following the main researches on the subject, we found several libraries dealing with this complex process of facial recognition, which was yet another indication that this issue is becoming increasingly popular.

The following are just a few from the list of libraries found throughout the investigations:

1. **OpenCV Face Recognizer** - OpenCV (Open Source Computer Vision Library) is a free library under a BSD license which includes hundreds of computer vision algorithms.
2. **Rekognition** - Alternative to Face.com, Rekognition can make facial detection, crawling, facial recognition and scene understanding. It can be automatically trained using images and tags just like on Facebook [12].
3. **CERT** (Computer Expression Recognition Toolbox) - is a fully automated system for recognizing facial expression that operate in real time.
4. **FaceRect** - is a powerful and free API for face detection. It finds faces (both front and profile) on the image specified by a URL or uploaded as a file and is able to find multiple faces in a single shot, the result being provided in JSON format with a bounding box for each face which is found [12].
5. **Face++** - uses the latest technology of computer vision and data mining to provide three basic services (detection, recognition and analysis). This API provides detection and Landmark analysis.
6. **FaceReader** - is a tool that is able to automatically analyse facial expressions, giving users an objective assessment of a person's emotions.

Finally after an analysis of the advantages and disadvantages of each library we decided that the best choice for this work is OpenCV library. This library is free although it is very complex and offers various facial recognition algorithms.

To implement this application we faced the following problems:

- against all expectations, the Android API was not enough to make this application;
- the faces detection involves finding ways to make a difference between them and the background.

Open Computer Vision Library (OpenCV) is a free graphics library developed by Intel in C language and distributed under a BSD license and has a huge community of developers and

DOI: 10.12948/issn14531305/20.1.2016.02

users because this library is used either individually or by companies.
It allows creating very complex calculations on pictures and video streaming, providing multiple features [13]:

- image processing;
- video streaming processing;
- real-time graphics processing;
- pattern recognition;
- image analysis.

The library offers three methods of facial recognition: Eigenfaces, Fisherfaces and Local Binary Patterns Histogram (LBPH) [14].
All of these methods make the face recognition by comparing a face to be recognized with a trained set of faces previously processed and stored in memory. We opted for using this algorithm in our application that is called FaceIdentity.

4 FaceIdentity – A Model of Implementation for Face Detection and Face Recognition in Android

The purpose of this application is to implement detection and facial recognition on the Android platform. The library was imported into the Android project and then we created a new class (CameraView) that extends the base class (JavaCameraView) of the OpenCV library. The most important method in the CameraView class is takePicture () which saves picture in the phone memory.

```
public void takePicture(final String fileName) {
    PictureCallback callback = new PictureCallback() {
        private String mPictureFileName = fileName;
        @Override
        public void onPictureTaken(byte[] data, Camera camera) {
            Log.i(FaceApplication.LOG_TAG, "Saving a image to file");
            Bitmap picture = BitmapFactory.decodeByteArray(data, 0, data.length);
            try {
                FileOutputStream out = new FileOutputStream(mPictureFileName);
                picture.compress(Bitmap.CompressFormat.JPEG, 90, out);
                picture.recycle();
                mCamera.startPreview();
            } catch (Exception e) {
                e.printStackTrace();
            }
        }
    };
    mCamera.takePicture(null, null, callback);
}
```

Listing 1. The implementation of takePicture() method in CamerView class

Another important function that we implemented by using OpenCV library functions was **onCameraFrame()**. This method uses **MatOfRect**, **Utils** and **Mat** classes from the OpenCV library, with which the camera detects the faces on the camera frames and using the **rectangle()** function from the **Core** class, it borders the face with a rectangle.

DOI: 10.12948/issn14531305/20.1.2016.02

```
public Mat onCameraFrame(CvCameraViewFrame inputFrame) {
        mRgba = inputFrame.rgba();
        mGray = inputFrame.gray();
        if (mAbsoluteFaceSize == 0) {
                int height = mGray.rows();
                if (Math.round(height * mRelativeFaceSize) > 0) {
                        mAbsoluteFaceSize = Math.round(height * mRelativeFaceSize);
                }
        }
        MatOfRect faces = new MatOfRect();
        Rect[] facesArray = faces.toArray();

        if ((facesArray.length == 1) && (faceState == TRAINING)
                        && (countImages < MAXIMG)
                        && (!text.getText().toString().isEmpty())) {
                Mat m = new Mat();
                Rect r = facesArray[0];

                m = mRgba.submat(r);
                mBitmap = Bitmap.createBitmap(m.width(),m.height(),
                                Bitmap.Config.ARGB_8888);
                Utils.matToBitmap(m, mBitmap);

        } else if ((facesArray.length > 0) && (faceState == SEARCHING)) {
                Mat m = new Mat();
                m = mGray.submat(facesArray[0]);
                mBitmap = Bitmap.createBitmap(m.width(), m.height(),
                                Bitmap.Config.ARGB_8888);

                Utils.matToBitmap(m, mBitmap);
                }
        for (int i = 0; i < facesArray.length; i++)
                Core.rectangle(mRgba, facesArray[i].tl(), facesArray[i].br(),
                                FACE_RECT_COLOR, 3);
        return mRgba;
}
```

Listing 2. The implementation of onCameraFrame() method

Training the application with a set of faces that could be used for subsequent recognition method is made by **train()** method that reads all faces previously saved and pass them to the class **FaceRecognizer** from OpenCV library as follows:

```
public boolean train() {
        File path = new File(mPath);
        File[] imageFiles = path.listFiles(pngFilter);
        for (File image : imageFiles) {
                String p = image.getAbsolutePath();
                 img = cvLoadImage(p);
                        grayImg = IplImage.create(img.width(), img.height(), IPL_DEPTH_8U, 1);
                        cvCvtColor(img, grayImg, CV_BGR2GRAY);
                 images.put(counter, grayImg);
                labels[counter] = label;
                        counter++;
        }
         faceRecognizer.train(images, labels);
```

Listing 3. The train() method

DOI: 10.12948/issn14531305/20.1.2016.02

A face recognition is performed using the **predict()** method from the class **FaceRecognizer** belonging to OpenCV library.

```
public String predict(Mat m) {
        if (!canPredict()) {
                return "";
        }
        int n[] = new int[1];
        double p[] = new double[1];
        IplImage ipl = MatToIplImage(m,WIDTH, HEIGHT);
        faceRecognizer.predict(ipl, n, p);
        if (n[0]!=-1) {
                mProb=(int)p[0];
        } else {
                mProb=-1;
        if (n[0] != -1) {
                return labelsFile.get(n[0]);
        } else {
                return "Unkown";
        }
}
```

Listing 4. The implementation of image prediction

5 The results of FaceIdentity Implementation

We tested the application on different types of images that contain from 1 to 100 faces, testing the face detection and face recognition functions.

The results of face recognition depends on the light conditions and also on the face position on the camera.

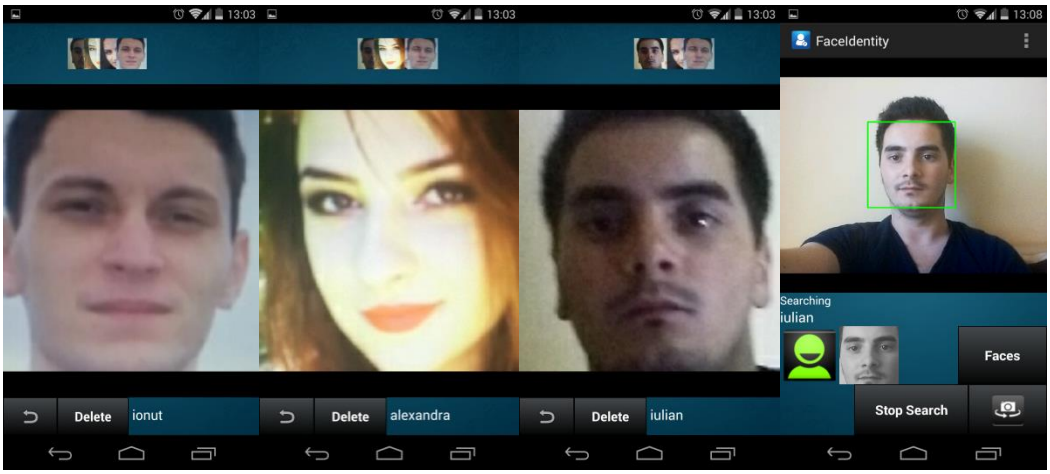

Fig. 3. The face recognition process in FaceIdentity application (author's face)

The face detection process works very quickly and we made tests on images with groups having almost 100 frontal faces on the camera, as it is presented in the following figure.

DOI: 10.12948/issn14531305/20.1.2016.02

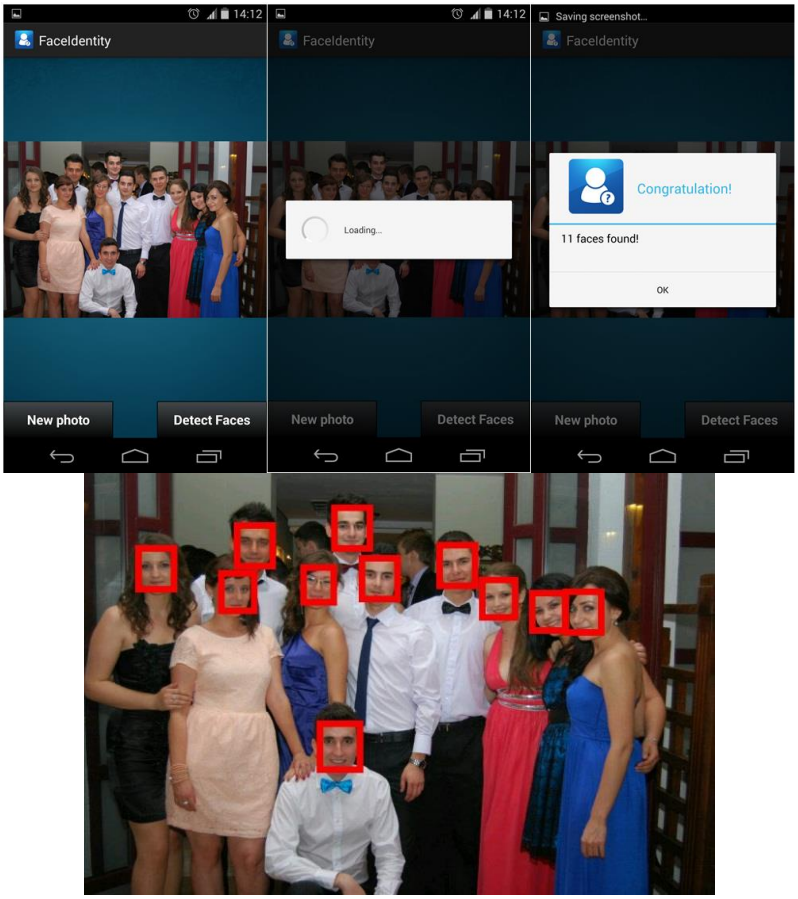

Fig. 4. The face detection process in FaceIdentity application (a small group: the authors and their colleagues)

With more faces on the camera, the face detection process works much better than facial recognition.

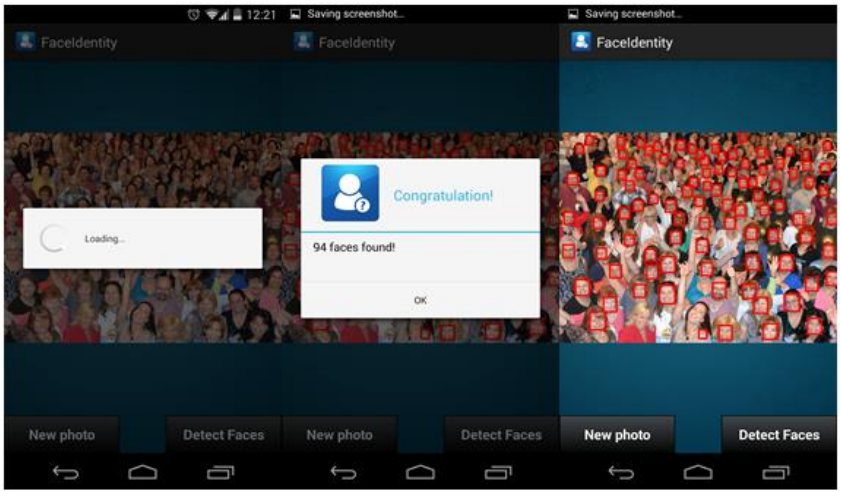

Fig. 4. The face detection process in FaceIdentity application (a large group with almost 100 faces)

6 Conclusions and Future Directions

Today the cameras and microphones are very small, lightweight and have been successfully integrated into wearable systems. Audio and video recognition systems have the advantage that they use critical ways that people use them for recognition.

Face recognition systems used today works

DOI: 10.12948/issn14531305/20.1.2016.02

very well in limited circumstances, although they work well with frontal images and constant illumination. We can say that the majority of current face recognition algorithms fails in different conditions in which people need to use the idea of this technology. The next generation of facial recognition systems should recognize people in real time and in circumstances far less limited.

The potential developments of this subject are countless; the technology for facial recognition can be used in the future in almost every industry, from education, medicine, marketing, international security, stadiums, government offices, transport, airports, borders etc.

A practical example where this technology could revolutionize the world and change the way we interact not only among ourselves but with the objects around us could be counting the time spent by a person in front of a billboard, so relying on real data, advertisers can make decisions more accurate and favorable oriented exactly on people interested in their product.

Another example of the use but on another level could be the identification of the persons present in the examination room, where just by photographing the exam room, the teachers will know exactly who is present or absent. Examples may continue to identify unwanted persons on a list of suspects previously entered into the database or recognizing companies' employees or clients with which they interact.

Acknowledgments:

„This work was supported by a grant of the Romanian National Authority for Scientific Research and Innovation, CNCS – UEFISCDI, project number PN-II-RU-TE-2014-4-0748".

References

[1] M. Rouse, "Facial Recognition," February 2012. [Online]. Available: http://whatis.techtarget.com/definition/facial-recognition.

[2] O. Dospinescu and I. Lîsîi, "The Recognition of Fingerprints on Mobile Applications–an Android Case Study," *Journal of Eastern Europe Research in Business and Economics*, vol. 2016, pp. 1-11, 2016.

[3] N. Dospinescu, "The Public Relations Events in Promoting Brand Identity of the City," *Economics and Applied Informatics*, pp. 39-46, 2014.

[4] R. Jafri and H. Arabnia, "A Survey of Face Recognition Techniques," *Journal of Information Processing Systems*, vol. 5, no. 2, pp. 41-68, 2009.

[5] "Face Recognition - Technology Overview," Ex-Sight, 2009. [Online]. Available: http://www.ex-sight.com/technology.htm. [Accessed 2016].

[6] S. Li and A. Jain, Handbook of Face Recognition, London: Springer-Verlag, 2011.

[7] L. Introna and H. Nissenbaum, "Facial Recognition Technology. A Survey of Policy and Implementation Issues," New York University, New York, 2009.

[8] R. Chellappa, C. Wilson and S. Sirohey, "Human and machine recognition of faces: A survey," *Proceedings of the IEEE*, vol. 83, no. 5, pp. 705-741, 1995.

[9] M. Turk and A. Pentland, "Face recognition using eigenfaces," *Computer Vision and Pattern Recognition, 1991. Proceedings CVPR '91,* pp. 586-591, 1991.

[10] E. Arubas, "Face Detection and Recognition (Theory and Practice)," 6 April 2013. [Online]. Available: http://eyalarubas.com/face-detection-and-recognition.html. [Accessed 16 January 2016].

[11] C. Strîmbei, O. Dospinescu, R.-M. Strainu and A. Nistor, "Software Architectures – Present and Visions," *Informatica Economica Journal,* vol. 19, no. 4, pp. 13-27, 2015.

[12] C. Ismael, "List of 10+ Face Detection / Recognition APIs, libraries, and

DOI: 10.12948/issn14531305/20.1.2016.02

software," 19 June 2013. [Online]. Available: http://blog.mashape.com/list-of-10-face-detection-recognition-apis/. [Accessed December 2015].

[13] IntelCorp, "Open Source Computer Vision," 25 January 2016. [Online]. Available: http://opencv.org/. [Accessed January 2016].

[14] D. L. Baggio, S. Emami, D. M. Escriva, K. Ievgen, N. Mahmood, J. Saragih and R. Shilkrot, Mastering OpenCV with Practical Computer Vision Projects, Packt Publishing, 2012.

Octavian DOSPINESCU graduated the Faculty of Economics and Business Administration in 2000 and the Faculty of Informatics in 2001. He achieved the PhD in 2009 and he has published as author or co-author over 30 articles. He is author and co-author of 10 books and teaches as an associate professor in the Department of Information Systems of the Faculty of Economics and Business Administration, University Alexandru Ioan Cuza, Iasi. Since 2010 he has been a Microsoft Certified Professional, Dynamics Navision, Trade&Inventory Module. In 2014 he successfully graduated the course "Programming Mobile Applications for Android Handheld Systems" authorized by Maryland University. He is interested in mobile devices software, computer programming and decision support systems.

Iulian POPA graduated the Faculty of Economics and Business Administration, Alexandru Ioan Cuza University of Iasi. He has a large background in developing business applications and his "get things done on time without excuses" attitude is what defines him. Currently he works at Pentalog where is implementing and integrating new automated systems for big european companies.

DOI: 10.12948/issn14531305/20.1.2016.02