Evaluating Computation Offloading Trade-offs in Mobile Cloud Computing: A Sample Application

Jorge Luzuriaga, Juan Carlos Cano, Carlos Calafate, Pietro Manzoni
Universitat Politècnica de València
Dept. of Computer Engineering
Valencia, SPAIN
Emails: jorlu@posgrado.upv.es, jucano@disca.upv.es, calafate@disca.upv.es, pmanzoni@disca.upv.es

***Abstract*—Mobile cloud computing is generally referred to as the infrastructure where both the data storage and the data processing happen outside of the mobile device. The nature of the connection between the cloud servers and the mobile host are anyway much less reliable than in classical cloud computing with static devices. A compromise must be found between local versus remote computation so to cope with the reduced performance of the data connection and with the characteristics of the mobile device, basically its power availability limitations. In this paper, we evaluate the tradeoffs of computation offloading using as a case study a facial recognition application for smartphones where recognition is a service in the cloud. We present a specifically designed application for mobile devices developed as a component of the proposed evaluation system. The intensive calculus needed for the image manipulation is compared in terms of speed and accuracy both when we delegate it to cloud computing and when we perform it locally on the mobile device. These two alternatives and the intermediate options are compared to determine the optimal settings to take better advantage of integrating these two technologies.**

***Keywords*—Cloud Computing; Facial Recognition; Mobile devices Applications; Process Outsourcing; Mobile Cloud Computing.**

I. INTRODUCTION

The latest advances in mobile communication networks and the increasing penetration of smartphones and other mobile devices, like tablets and portable computers, are transforming the mobile Internet and allowing the users to improve their mobile experience. However, the limited computing and information/energy storage capabilities of mobile devices are delaying their abilities to support increasingly sophisticated applications demanded by users. The emerging cloud computing technology offers a natural solution to extend the limited capabilities of mobile devices. The resulting new paradigm of mobile cloud computing is being adopted by researchers as a powerful new way to extend the capabilities of mobile devices and mobile platforms, which has the potential of a deep impact on the business environment and people's daily life.

The decision of where to place the execution (local or remote mode) should be anyway made based on the quantity of computation and communication that is required by the application. A little amount of communication combined with a large amount of computation should be performed preferably in remote mode, while a large amount of communication combined with a little amount of computation should be performed preferably in local mode.

In this work, we chose face recognition as a sample application to evaluate the tradeoff of offloading computation with the intuitive idea of the required intensive calculus puts in commitment the hardware features of the mobile device. Whereas that, if the same calculus are executed by other systems with better hardware features, these processes are realized with less effort and in much less time.

We analyze the intensive calculus dividing it in sub processes that are distributed between the mobile device and the cloud infrastructure using a cascade of classifiers based on the Adaboost algorithm [19] to detect the presence of faces in an image and the Eigenfaces algorithm [11] to make the training and recognition of these faces.

Finally, we emulate the wireless channel between the mobile device and the cloud server to view how the end-to-end response time can affect at application. And also this emulation allow us to find limitations where we can get advantage with the use of this technique.

The rest of the paper is organized as follows: Section II presents the works related to the topic. Section III describes the proposed system overview, Section IV shows a sample case study: a facial recognition application. In Section V, we present a testbed to evaluate this proposal and in Section VI, we show the results obtained in the tests. The article finalizes with the conclusions in Section VII.

II. RELATED WORK

The computation off-loading from mobile devices to computational cloud infrastructure is based on deciding which methods should be remotely executed, so that benefits can be achieved in terms of both time and use of resources that ends up in saving energy. In the literature, we found a wide set of proposals with frameworks that decide dynamically whether a part of application will be executed locally or remotely [5][8][12][13][14]. Other proposals utilize nearby computers, also known as surrogates. These surrogates are resource-rich computers connected to the Internet and available to use with nearby mobile devices without incurring in WAN delays and jitter. Their objective is similar to proxy servers [15]. More aggressive proposals in which the entire user mobile devices are cloned on a remote server operating in a cloud where the execution of processes would be faster [3][4]. It enables moving an entire operative system and all its applications to selectively execute some processes on the clones, and integrating the results back into the mobile device.

 ISBN: 978-1-61208-271-4

Also, we find in [10] a cost/benefit analysis focused on energy saving for off-load computation to a server, taking into account: the processor speed, the instruction number, the bytes of data exchanged among server and mobile system, the network bandwidth, the energy consumption in different states, reaching the conclusion that energy-saved with off-load computation is greater that the energy spent on communicating with the cloud. So, if the computation is offloaded with relatively low cost, the processing in remote mode may be energy-efficient to the mobile device.

Finally, commercial proposals with applications implemented in client-server model, where the data to make the computation are transferred from a smartphone as a light weight client to the remote heavy weight server hosted in the cloud. These services typically have to interact with very large databases and require maintenance costs. An example of this type of applications is the popular Google goggles that uses pictures to search the web [6].

However, these proposals can only be used when the user is permanently connected to the Internet. Moreover, when we delegate the processing to external entities in many cases these entities follow the pay per use philosophy and their services are offered in monthly plans with a limit number of queries or charge for each query. These services provide a better performance and request priority with technical support in problematic situations. From the point of view of the user, the condition to "pay per use" of any application adds to the cost of the data line, making the remote processing economically speaking more expensive compared with local processing. Finally, it is necessary to consider the security issues in the communication process. When the information is transferred from the mobile device to the server in cloud environments, the data can be sniffed from a person/machine that listens the communication channel, then the data transfers to remote processing is less reliable compared to process the data locally whereas not necessary any transmission by the wireless network.

III. Facial Recognition

Face recognition refers to the automatically identification of a person from a digital image. The process involves: (i) face detection, (ii) feature extraction, (iii) creation of the database with known faces, and the (iv) matching with the new face.

We use the Eigenfaces algorithm that applies data dimension reduction with the minimum information lost by PCA Principal Components Analysis [8] to get the coefficients values and is able to make the matching based on the minimum distance.

The Eigenfaces process chooses the factors with high correlation because with the redundant information that exists among them it is possible to select the coefficients that contain the maximum variability and in this form get the dimension reduction of the data. Once selected, the principal coefficients are representing in matrix form.

The process of automatic recognition can be clearly separated into two stages: the training and the recognition stage. The training stage is required to learn using a classifier [19]. In our case, the learning consists in transforming the features of human faces into the form of coefficients and to store them in a matrix. This matrix represents a database of facial features of known faces. In the recognition stage, the classifier is used again to classify the data of the test image and to get values of correlation coefficients that represent the face found in the image; this stage realizes the features extraction. Finally, the values obtained are compared with the matrix values of the training stage. The minimal difference among these comparisons is the result of recognition process.

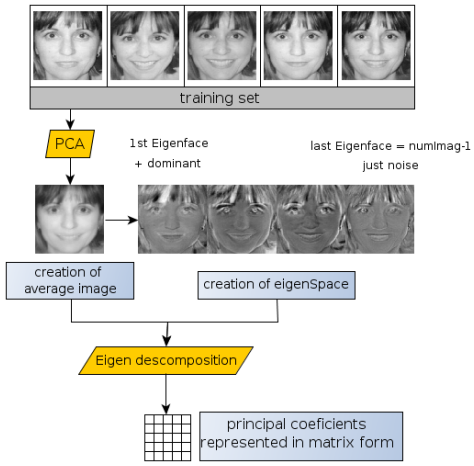


Fig. 1: The general process of training and recognition.

In the upper part of Figure 1, we have a training set of 5 images of the same person. To work with a standard input, these images have previously passed various sub processes: (i) compression, (ii) grayscale, (iii) crop, (iv) resize, and (v) equalization. The next step is to apply the process called Principal Components Analysis (PCA) where the principal features of each image are extracted and represented as another set of images with ghostly aspect. These images contain only relevant information to recognition process. The complete set of these images is called Eigenspace. Also, the PCA creates a common image that represents all images in the Eigenspace. In the Eigenspace, the first image is the most dominant and contains the representative features of all faces and the last images are the weakest that contain common features that can be found in any face. According to the specified threshold the weakest images cannot be taken into account because they are considered as noise. The last step is to get the numeric coefficient of each image in the Eigenspace through Eigen decomposing and represent in matrix form. So, each column of the matrix represents one image in the Eigenspace. A complete explanation of Eigenfaces process can be found in [11].

IV. Evaluation approach

When we delegate the processing to external entities we benefits in terms of hardware resources use and inclusive in terms of energy saving, but the latency in communication with the external entity is a big overhead especially when large data are involved. To reduce this problem a preprocessing of the information is required so that the accuracy results in posterior processes are not affected.

 ISBN: 978-1-61208-271-4

The cloud-based platform hosts the application in a centralized form and provides a software delivery model known as Software as a Service or SaaS in which the customers or users can access remotely using a thin client over the internet [9]. The main advantages of Software as a Service is the possibility to offer better services in a form totally scalable with the demand.

The first step is to evaluate the baseline performance of the application when run locally in the mobile device. Then, we identify the application dependencies among the executed processes identifying the parts of the code that can be executed in the cloud. Finally, we estimate the response time of the application in a cloud-based platform before of its deployment.

We will therefore implement and deploy applications according to different local versus remote mix to compare their execution performance. The different ways, modes or scenarios describe the place where the intensive calculus are realized. We define a first scenario where the mobile device has the capability to process images and to make recognition locally, called local mode. A second scenario where the recognition process is offloaded to a cloud computing infrastructure, called remote mode. And a third and last scenario where the recognition is performed in a distributed environment, mixing features of the two previous scenarios, called mix mode.

To know how the end-to-end response time can be affected, we emulate the communication channel in different conditions, to this end we used the Linux kernel tools of routing, filter and classification of packets to guarantee performances, bandwidths and lower latencies respectively by the utilities collection grouped in iproute2. Between the principal tools and utilities we used arpd, cstat, ifcfg, ip and tc. The last two utilities are known as LARTC of Linux Advanced Routing and Traffic Control [7] to manage the traffic traversing a network interface.

To add scenarios with delay and packet loss, we made use of NetEm that is a network emulator. This network emulator permits to convert the local area network in a slow and heavy network as can be an extended area network. NetEm is perfect to evaluate the behavior of protocols, applications and final systems, which have to be used on distributed environments. Originally NetEm behaves as a FIFO queue without delays or packet loss. To modify its discipline and its parameters, we can do it by the tc command in a Linux shell. To modify the parameters, we consider that the network delay is a variably value that depends of the amount of traffic that fluid by the same network. Generally, the delay is represented as a normal distribution with a medium value more/less the standard deviation value. The next parameter that we specify to modify the network behavior is the packet loss, here NetEm deletes randomly as packets as needed to fit to parameter.

V. EXPERIMENTS

As a basic tool for our evaluation, we developed a mobile client application for Android devices. A snapshot of this application is shown in Figure 2, where the green box indicates the face detected. In the bottom part of the screen, the characters indicate the name of the recognized person. And if appear a number next to the name, it indicates the age estimation of the person as an extra detail only available in the remote mode.

Fig. 2: Snapshot of the face recognition mobile application.

This application first captures pictures of people faces in local mode without using any communication with the external server, as if the cloud server was unavailable, the application makes the image processing necessary in the learning stage and the application is ready to receive any face image of people to make the recognition stage. In the case in which the network and the cloud server are available, the application automatically sent the data to remote processing. And the last function of the application is the visualization of the results as a front-end terminal.

The evaluation of the application has been performed on Samsung Galaxy Ace Smartphone running Android OS version 2.2 connected to internet by a WiFi and 3G networks. The testbed consists of execution of the application with the objective of getting the average execution time of each sub process with the same workload used in different scenarios, as shown in Figure 3. To get normalized values each test was repeated 10 times. The face recognition process is made with different values of people from 5 up to 20. And to each person different face images, from 1 up to 5, getting training sets with number of images multiples of 5.

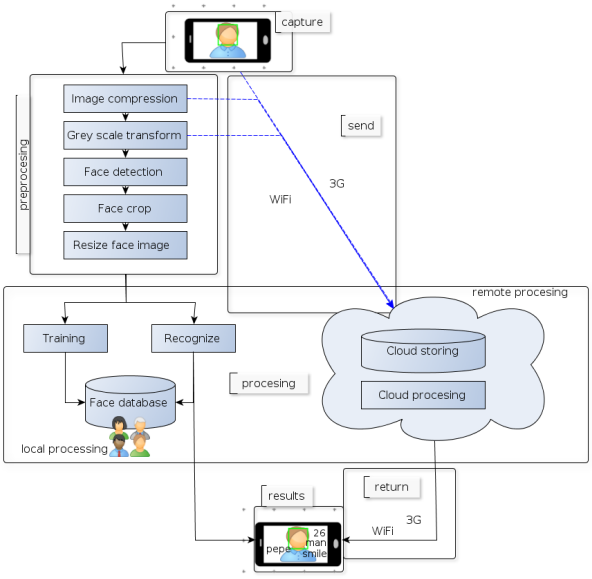


Fig. 3: Proposed system architecture.

Regarding network emulation, we first estimated the real values of latencies and packet lost experienced in a communication with a server in a cloud infrastructure, Then we used these values as guidelines to modify the network conditions when we make a POST request from the mobile device to servers in cloud infrastructure. The network conditions such as network delay, bandwidth rate, and packet loss were modified through the NetEm parameters. These tests was realized with virtual machines: we had a virtual machine executing a web server as a server on a cloud infrastructure. And the request were realized from another virtual machine, representing a mobile phone. Here, we varied the delay parameter using the values 2, 10, 50, 100 and 500 milliseconds and with each one of these values we used a different value for the packet loss parameter from 0% (emulating a perfect channel) passing for 1, 5, 10 until 20% (as a noisy channel). Each test was repeated for 10 occasions, too.

VI. RESULTS

In all scenarios it was necessary to process the captured image with an average cost in time of 480 ms. Then, the average times of each of the other pre-processes are shown in the table 1 and are graphically represented in the Figure 4. The processes that consume more time are: convert color images to grayscale (595 ms) and the detection of faces in an image (667 ms).

TABLE I: Time consumed for each sub-process to pre-processing images

Pre-process	Average Time in milliseconds
compression	180
gray scale	595
face detection	667
crop face	12
re-dimension	30
equalize	10

Now, please remember that processing is made of two stages: training and recognizing. In the training stage the overall system time is related directly to the number of images in the training set (while more images for each person more accurate are the recognition results). This creates a first trade-off between the required time to make the training and the

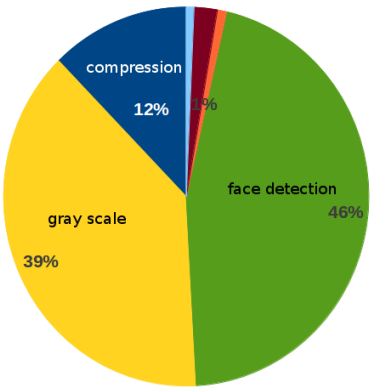


Fig. 4: Proportion of time consumed in pre-processing an image.

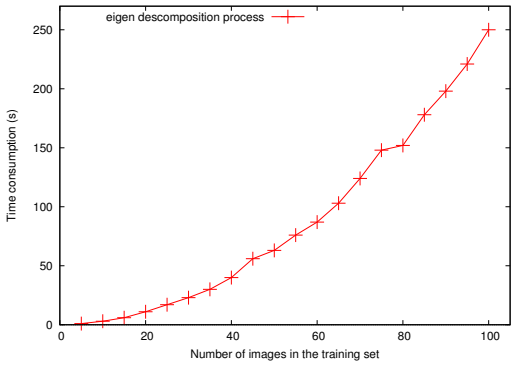


Fig. 5: Time required in the learning stage of local mode according to the number of images in the training set.

number of samples to train. As we can see in the Figure 5, a training set of 5 images needs 1,5 seconds to train while the case of a training set of 100 images needs 250 seconds. These values have an quadratic growth.

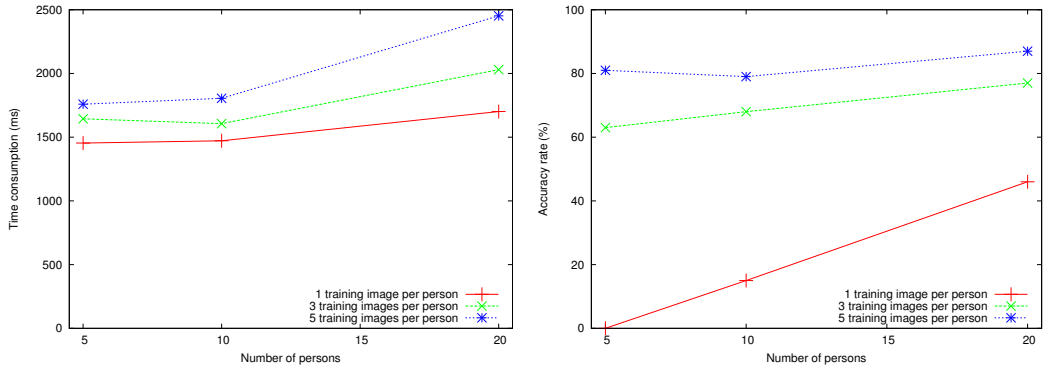
Considering the time required for the application to do the training stage can affect the user's patience. A reasonable size of training set can be of 50 images that requires 63 seconds of waiting. The recognition stage need less time when compared with the training stage, to any person whose images have been previously trained, the facial recognition is realized in a range of 1,5 to 2,5 seconds according the number of images per person, as we see in Figure 6a. Here, we confirm that the fact of using Eingenfaces converts the recognition process in a quick process and permits to operate with wide sets of faces in very short times [11].

The recognition accuracy rate when we use only one image per person is unreliable because it doesn't arrive to 50%. With 3 images per person is over 60%, but continues being unreliable. We reach close to 80% accuracy, when we use 5 images per person. As we can see in Figure 6b.

In remote processing, the images were sent in first place via a WiFi network and then via a 3G network. When we send images product of a strong pre-processing, the results are obtained in 939 ms with WiFi and in 3908 ms with 3G. When we send images with a lighter pre-processing we obtain the results in 2045 ms with WiFi and in 9790 ms with 3G. As we can see in Figure 7a.

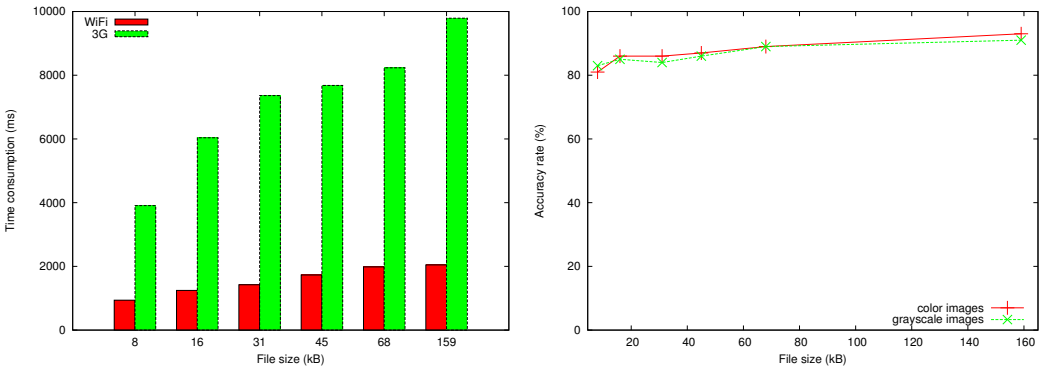
In remote mode, the accuracy rate (Figure 7b) with images whose size is in the range of 8 kB to 102 kB is over the 80%. With images of 160 kB the accuracy is 91%. Namely better results with images without compression or in general without apply the pre-processing steps. But, if we avoid the pre-processing steps, the communication is affected by performance loss. This scale the problem size with higher latency and occupancy of bandwidth. To overcome these limitations, the scenarios with mix mode, we consider that the pre-processes of gray scale and face detection with 595 ms and 667 ms respectively are very expensive in terms of time consumption, then we decide not to use them. Simply the images captured

 ISBN: 978-1-61208-271-4



(a) time consumption using different sizes of training vectors (b) face-recognition accuracy

Fig. 6: Results obtained in local mode



(a) time consumption using different wireless networks (b) face-recognition accuracy

Fig. 7: Results obtained in remote mode

are compressed with JPG standard at 85%, to get a reduced version of these images with a size of near 16 kB. Then these images are sent to remote processing in the cloud infrastructure preferably using the WiFi network, to obtain results in 1712 ms with an 86% accuracy. It is the best combination and it can be considered as the optimum mode that we recommend to use.

If these preprocessed data are sent via WiFi network the recognition results are timeless that when are realized in local mode inclusive including the latency and round-trip delay time communicating with remote mode. The latter method requires network connectivity from the mobile device to Cloud environments. In cloud side due hardware potential and the complex of the algorithms, this scenario can provide more accurate results.

In the emulation of network conditions, with a 3G channel, the time required to make the request and get the response with an ideal channel (0% of packet loss) is 6 seconds with the minimum delay (2 ms), and with the maximum delay (500 ms) the response is obtained in 22 seconds. In Figure 8a, is displayed linear growth of the time necessary to receive a reply, under the differing amounts of packet loss for some link latencies.

Finally, we modify the latency values in WiFi channel from 2 ms up to 500 ms, the emulation deliver values from 800 ms to 1800 ms respectively, as we can seen in the Figure 8b.

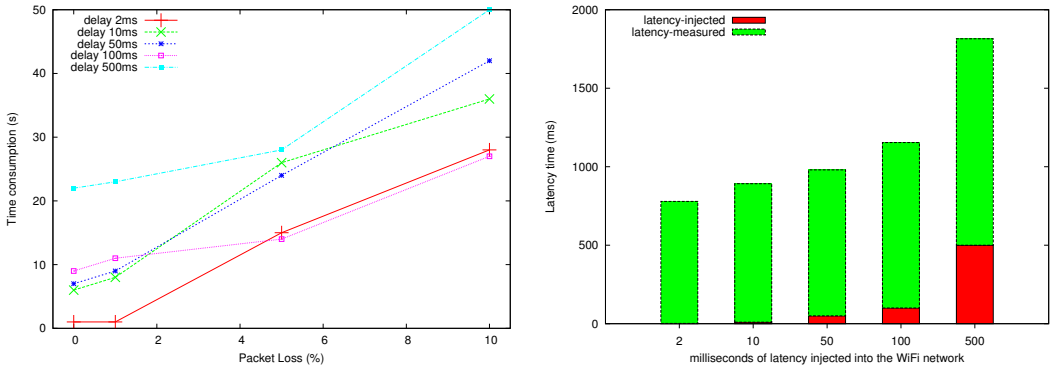
VII. CONCLUSIONS

The growth of complex applications to mobile devices with support of cloud computing infrastructure demands better understanding of the effects of latency and packet loss. The communication client-server in wireless environments might suffer more latency and are more prone to packet loss. This communication is also affected by the Internet latency.

For this reason in this paper, we presented an application designed to allow the isolation of each process involved in a recognition of a face, integrated in a testbed that allowed the control of network conditions, such as latency and packet loss.

From the obtained results, we consider that offloading computation from mobile devices to cloud computing infrastructure can be done safely only if we have a guaranteed availability of a stable channel. In fact, with a broadband access of a WiFi network, we have low aggregate latencies close to 1 second. And if we use the 3G network, we have aggregate latencies near to 4 seconds. Both options with a packet loss level under

 ISBN: 978-1-61208-271-4



(a) with changes in percentage packet loss and link delays

(b) modifying the latency values in WiFi channel

Fig. 8: The times obtained in the channel emulation using NetEm

3%. These are suitable for a good performance. When the packet loss level is over 3%, latencies are highly affected and this can be annoying to the user's experience. For these cases is preferable to use the local calculus with order to keep low latencies.

With the continuous evolution of mobiles devices and the communication networks, it is possible to design, develop and use applications that combine the two operational modes in better efforts. For example, using these operational modes in applications where we will get the results in less of one second in autonomous mode or we will automatically use the remote mode sending queries to remote servers and get results in 2 seconds in normal cases or in 5 seconds in the worst case. With recognition training vectors previously charged to both options. If these results are not correct or are unreliable is possible to aggregate new registers manually, to future queries in a crowd sourcing style. As the people's identity is a delicate theme. We can use this architectural proposal and the image processing in other aims, following with the visual content that can be found in an image, that requires recognition and identification.

ACKNOWLEDGMENTS

This work was partially supported by the *Ministerio de Ciencia e Innovación*, Spain, under Grant TIN2011-27543-C03-01.

REFERENCES

[1] D.-Y. Chen and J.-T. Tsai, "Resource-limited intelligent photo management on mobile platforms," in Machine Learning and Cybernetics (ICMLC), 2011 International Conference on, Jul 2011, pp. 627–630.

[2] P. Angin, B. Bhargava, and S. Helal, "A mobile-cloud collaborative traffic lights detector for blind navigation," in Proceedings of the 2010 Eleventh International Conference on Mobile Data Management, ser. MDM '10. Washington, DC, USA: IEEE Computer Society, 2010, pp. 396–401.

[3] B.-G. Chun, S. Ihm, P. Maniatis, M. Naik, and A. Patti, "Clonecloud: elastic execution between mobile device and cloud," in Proceedings of the sixth conference on Computer systems, ser. EuroSys '11. New York, NY, USA: ACM, 2011, pp. 301–314.

[4] B.-G. Chun and P. Maniatis, "Augmented smartphone applications through clone cloud execution," in Proceedings of the 12th conference on Hot topics in operating systems, ser. HotOS'09. Berkeley, CA, USA: USENIX Association, 2009, pp. 8–8.

[5] E. Cuervo, A. Balasubramanian, D. Cho, A. Wolman, S. Saroiu, R. Chandra, and P. Bahl, "Maui: making smartphones last longer with code offload," in Proceedings of the 8th international conference on Mobile systems, applications, and services, ser. MobiSys '10. New York, NY, USA: ACM, 2010, pp. 49–62.

[6] Google, "Google goggles," URL: http://www.google.com/mobile/goggles, [retrieved: 03, 2013].

[7] T. Graf, G. Maxwell, R. van Mook, M. van Oosterhout, P. B. Schroeder, J. Spaans, and P. Larroy. Linux Advanced Routing & Traffic Control HowTo. URL: http://www.lartc.org/. [retrieved: 03, 2013].

[8] Y. Guo, L. Zhang, J. Kong, J. Sun, T. Feng, and X. Chen, "Jupiter: transparent augmentation of smartphone capabilities through cloud computing," in Workshop on Networking, Systems, and Applications on Mobile Handhelds, ser. MobiHeld '11. New York, NY, USA: ACM, 2011, pp. 2:1–2:6.

[9] H. T. Dinh, C. Lee, D. Niyato, and P. Wang, "A survey of mobile cloud computing: architecture, applications, and approaches," in Wireless Communications and Mobile Computing. Wiley Online Library, 2011.

[10] K. Kumar and Y.-H. Lu, "Cloud computing for mobile users: Can offloading computation save energy?" in Computer, vol. 43. IEEE Computer Society, 2010, pp. 51–56.

[11] L. Lorente Giménez, "Representación de caras mediante eigenfaces," in Buran, vol. núm. 11, 1998, pp. 13–20.

[12] E. Marinelli, "Hyrax: Cloud computing on mobile devices using mapreduce," Master's thesis, Carnegie Mellon University, 2009.

[13] J. S. Rellermeyer, O. Riva, and G. Alonso, "Alfredo: an architecture for flexible interaction with electronic devices," in Proceedings of the 9th ACM/IFIP/USENIX International Conference on Middleware, ser. Middleware '08. New York, NY, USA: Springer-Verlag New York, Inc., 2008, pp. 22–41.

[14] R. Kemp, N. Palmer, T. Kielmann, and H. Bal, "The smartphone and the cloud: Power to the user," in MobiCloud, vol. 28, October 2010.

[15] M. Satyanarayanan, P. Bahl, R. Caceres, and N. Davies, "The case for vm-based cloudlets in mobile computing," in Pervasive Computing IEEE, vol. 8, No. 4, 2009.

[16] Y. Taigman and L. Wolf, "Leveraging billions of faces to overcome performance barriers in unconstrained face recognition," *CoRR*, vol. abs/1108.1122, 2011.

[17] X. Zhang, A. Kunjithapatham, S. Jeong, and S. Gibbs, "Towards an elastic application model for augmenting the computing capabilities of mobile devices with cloud computing," in Mobile Networks and Applications, vol. 16, Jun. 2011, pp. 270–284.

[18] W. Zhao, R. Chellappa, P. J. Phillips, and A. Rosenfeld, "Face recognition: A literature survey," in ACM Computing Surveys (CSUR), vol. 35, Dec. 2003, pp. 399–458.

[19] OpenCV, "Opencv v2.4.3 documentation," URL: http://docs.opencv.org/modules/ml/doc/boosting.html, [retrieved: 03, 2013].

 ISBN: 978-1-61208-271-4