MoBeTrack: A Toolkit to Analyze User Experience of Mobile Apps in the Wild

A. Generosi[a], A. Altieri[a], S. Ceccacci[a], G. Foresi[b], A. Talipu[a], G. Turri[a], M. Mengoni[a] L. Giraldi [a]
[a]Dipartimento di Ingegneria Industriale e Scienze Matematiche,
[b]Dipartimento di Ingegneria dell'Informazione,
Università Politecnica delle Marche, Ancona, Italy
Email: {a.generosi, a.altieri, g.foresi, t.abudukaiyoumu, l.giraldi}@pm.univpm.it ,
{s.ceccacci, g.turri, m.mengoni}@univpm.it

***Abstract*--MoBeTrack (Mobile Behaviour Tracking) is a toolkit for automated collection of data necessary to support User Experience (UX) assessment of mobile applications. In contrast to existing frameworks, it is able to collect user demographic information (i.e., age and gender), trace any user interaction and recognize user's emotions during the use of an application. An SDK for iOS allows to easily embedding the toolkit in every mobile application in a flexible and scalable way.**

I. Introduction

Nowadays, to compete successfully in the ultra-competitive market of mobile devices applications, understanding UX in the wild is of paramount interest. In fact, mobile applications have to face a wide variety of external stimuli and environmental conditions that are difficult to simulate in a laboratory (e.g., the user might not be sitting in front on the screen; it could perform a number of other general tasks while using an application). Consequently, although formal tests with users are important, they are not sufficient to understand the actual application performances [1].

Currently available systems do not enable to carry out UX evaluation in a proper way. Commercial frameworks such as Flurry, Google Analytics or Localytics are only designed to collect usage analytics (e.g., user demographics) so that they do not allow to collect users' performance data or users' behavioral information. Several frameworks and toolkit have been proposed in literature to support usability analysis (e.g., EvaHelper Framework [2], RobotME [3]) and enable unsupervised usability evaluation [4]. However, they only provide navigational data and user input, while they do not allow the collection of objective data related to user's behavior (e.g., user's gaze and emotions).

To fully support UX assessment of mobile apps in the wild, MoBeTrack provides users' demographic data (e.g. age and gender), performance data (e.g., time to navigate a screen) and usage data (e.g., scrolling, tapping). Moreover, it exploits eye tracking and emotion recognition systems to allow the collection of behavioral information.

II. MoBeTrack System Architecture

This system makes use of a centralized architecture that, as shown in Figure 1, has two main actors: the iOS SDK on the client side, and the Deep Learning (DL) platform on the server side. The mobile SDK is an iOS framework that exposes some APIs to monitor all the user interactions during the mobile app usage. Among these features, there is the possibility to activate the camera that takes different photos with a certain frequency.

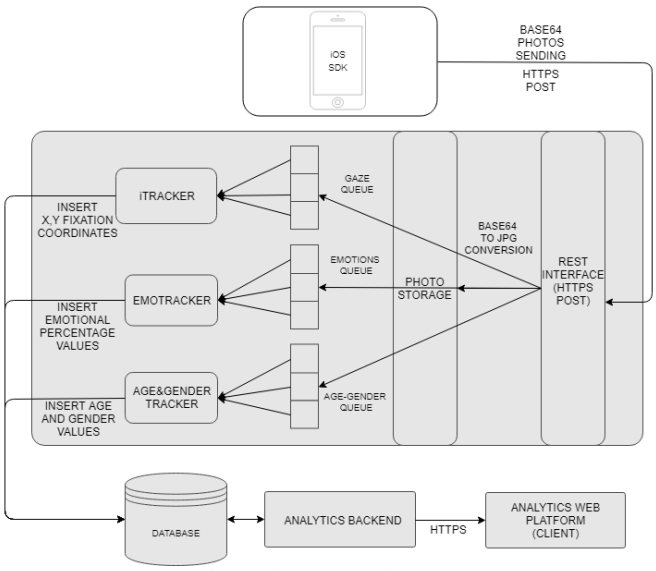

Fig. 1. The system architecture

These photos are Base64-encoded and sent to a server by HTTPS protocol. The central server that supports all the platform architecture, handles incoming calls from the iOS framework through a REST interface developed in Python, that waits for POST HTTPS calls addressed to the exposed endpoint. Once the call is received, the content is parsed and decoded to get all the data, included the original JPEG subsequently stored in the physical memory. After that, the JPEG file name, that uniquely identifies the photo, is stored in three different Redis queues so that, through the path of the directory where the files are physically located, it is possible for every DL Tracker module to obtain the position of the photos every time it arrives. These queues are so used by the three Tracker modules to respectively obtain the estimation of the user's gaze x-y coordinates, his emotional state, and the gender and age information. All of DL Tracker modules are based on Convolutional Neural Networks (CNN) implemented in Python. Whenever the processing of a photo is terminated for all CNNs, the resulting data will be stored in a database and the photo itself will be permanently deleted from the server. All the stored data will be available through an Analytics Web Platform.

III. System Features

A. iTracker

MoBeTrack implements a CNN for eye tracking described in [5], called iTracker, which has been trained with a large-

scale dataset for eye tracking (i.e. GazeCapture) that contains data from 1474 subjects (~2.5M frames). Such dataset was collected through crowdsourcing, so that its large data variability allows to improve model robustness. The CNN takes in input the images of the left eye, the right eye and the face detected and cropped from the original image, and a binary mask (face grid) used to indicate location and size of the face within the frame, and provides in output the x-y coordinates from the camera (in centimeters).

B. EmoTracker

A CNN was defined, based on a revised version of the VGG13 [7]. It consists of 10 convolution layers spaced with max pooling and dropout layers. It was trained from scratch on the FER+ dataset [6] able to recognize 8 types of emotion: neutral, happiness, surprise, sadness, anger, disgust, fear, and contempt. The CNN takes in input an aligned grayscale face image at 64x64 and gives in output a probability for each emotion.

C. Age and Gender Tracker

Estimation of age and gender is accomplished by another CNN trained from scratch on the IMDB-WIKI dataset [8]. The Wide Residual Network (WideResNet) [9] architecture was adopted and two classification layers were added: one with 101 units for age estimation and another with 2 units for gender classification. The deepening and widening factors of the network are set to 16 and 8 respectively. The model is fed with 256x256 aligned face image and it outputs the age and gender probabilities.

D. iOS SDK

The client-side of MoBeTrack is a framework designed for Apple smartphones. This framework, once embedded in iOS applications, allows to send to the server all the activities performed by the user with the app. Firstly, by using the smartphone front camera, it takes photos silently with a frequency of 0.5 Hz and analyzed them at the server-side. Secondly, each time a user taps the screen, it stores the x-y coordinates of the tapped point (in pixels). Also information about scroll activities is taken into account. Each time the user scrolls the current view, the y-offset from the left-top corner of the screen is updated. All these values are sent to the server whenever one of the aforementioned actions is performed.

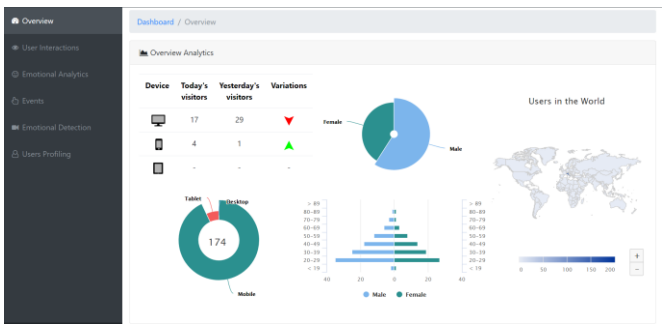

Fig. 2. The Overview section of the Analytics platform.

E. Analytics Web Platform

A web platform (Figure 2) has been developed to visualize the analytic data about the app usage. The tap and gaze information is used to generate heatmaps.

IV. Results and Conclusion

This work proposed a system able to collect, analyze, and visualize data about user interactions/behaviour with mobile applications supporting UX designers.

Preliminary tests have been carried out on iPhone 4, iPhone 6s and iPhone 7 Plus to determine CPU and battery consumption and required RAM allocation, by varying the screen brightness and the frame shots frequency. Results are reported in the following table.

Table 1. Test Results

Shots frequency	Screen brightness	iPhone 4 CPU (%)	iPhone 4 RAM (MB)	iPhone 4 Battery (%)	iPhone 6s CPU (%)	iPhone 6s RAM (MB)	iPhone 6s Battery (%)	iPhone 7 Plus CPU (%)	iPhone 7 Plus RAM (MB)	iPhone 7 Plus Battery (%)
0.5 Hz	50%	25%	10	11%	7%	21	7%	14%	30	4%
	100%			16%			15%			13%
1 Hz	50%	28%	6	12%	9%	15	9%	19%	38	5%
	100%			17%			15%			14%
3 Hz	50%	56%	9	12%	15%	21	10%	24%	90	7%
	100%			19%			18%			15%

Future system improvements mainly concern SDK versions for Java/Android and cross-platform frameworks, and lighter CNN models to be embedded directly inside the SDK, reducing network traffic.

References

[1] Ravindranath, L., Padhye, J., Agarwal, S., Mahajan, R., Obermiller, I., & Shayandeh, S. AppInsight: Mobile App Performance Monitoring in the Wild. In *OSDI*, Vol. 12, pp. 107-120.

[2] Balagtas-Fernandez, F., & Hussmann, H. A methodology and framework to simplify usability analysis of mobile applications. In *Proc. of the 2009 IEEE/ACM International Conference on Automated Software Engineering*, pp. 520-524.

[3] Zduniak, M. (2007). Automated gui testing of mobile java applications. *Master's thesis, Poznan University of Technology Faculty of Computer Science and Management*.

[4] Lettner, F., & Holzmann, C. Automated and unsupervised user interaction logging as basis for usability evaluation of mobile applications. In *Proc. of the 10th ACM International Conference on Advances in Mobile Computing & Multimedia*, pp. 118-127.

[5] K. Krafka, A. Khosla, P. Kellnhofer, H. Kannan, S. Bhandarkar, W. Matusik, and A. Torralba, "Eye tracking for everyone", In *Proc. of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 2176-2184.

[6] E. Barsoum, C. Zhang, C. C. Ferrer, and Z. Zhang, "Training deep networks for facial expression recognition with crowd-sourced label distribution," In *Proc. of the 18th ACM International Conference on Multimodal Interaction*, 2016, pp. 279-283.

[7] K. Simonyan, and A. Zisserman, "Very deep convolutional networks for large-scale image recognition." In *arXiv preprint arXiv:1409.1556, 2014*.

[8] R. Rothe, R. Timofte, and L. Van Gool, "Dex: Deep expectation of apparent age from a single image," In *Proc. of the IEEE International Conference on Computer Vision Workshops*, 2015, pp. 10-15.

[9] S. Zagoruyko, and N. Komodakis, "Wide residual networks," In *arXiv preprint arXiv:1605.07146*, 2016.