Setting the iOS mobile platform for visual servoing applications

Marco Perez-Cisneros[1], Sabrina Vega-Maldonado[2], Erik Cuevas[1], Daniel Zaldivar[1], Patricia Sanchez-Rosario[1]

[1] División de Electrónica y Computación, CUCEI, Universidad de Guadalajara,
{marco.perez, erik.cuevas, daniel.zaldivar, patricia.sanchez}@cucei.udg.mx
[2] Maestría en Sistemas de Información, CUCEA, Universidad de Guadalajara,
vms842@alumnos.cucea.udg.mx

Abstract. This work aims to contribute towards developing a novel platform for deploying Visual Servoing (VS) applications over mobile processing units. As several operating systems and programming environments are available within a quite volatile market, one platform must be selected in order to explore its actual capacities for deploying a visually guided robotic application. This paper presents the on-going evolution of one project targeting the development of a real-time VS control toolkit under the iOS platform and its related tools. The discussion includes first results along with some conclusions and future work.

Keywords: iOS programming, visual servoing, objective-C language.

1 Introduction

Recent technology developments have enriched the consumer market by providing powerful processing platforms ranging from mobile telephone to gaming consoles. In particular, the technology around the last generation of mobile phones, largely known as smartphones, have ubiquitously integrated several devices which encompass novel user interfaces, powerful processors and a handy sensor set with relevant telecommunications ports under a unified architecture.

A comprehensive consumer market analysis from 2006 [1], estimated that there were more than 2.5 billion mobile phones worldwide at the time, with numbers at 2009 reaching 175 million of smartphones alone [2]. In the last four years, the development has been accelerated as many companies are investing largely on developing new hardware platforms and software applications. In 2010, the mobile market widened with experienced companies re-launching their efforts either to relocate their technology surpassing competitors in the segment or to introduce more powerful processing devices.

Advances in Computer Science and Electronic Systems
Research in Computing Science 52, 2011, pp. 77-86

Received 23/01/11
Accepted 20/02/11
Final version 07/03/11

Under such context, mobile processing in general is likely to drive computing subjects for the foreseeable future as new applications and services are becoming available worldwide [3]. Subjects such as distributed computing and pervasive mobile processing are rapidly evolving and becoming attractive to several subjects on science and engineering.
On the other hand, Visual Servoing (VS) is a research area regarding the commanding of robotic devices from visual information. It naturally shares common issues with robotic control, real-time systems and in particular with computer vision as several visual algorithms are demanded by VS, including active vision, visual pose estimation and dynamic vision.

This work aims to contribute towards developing a novel platform for deploying VS application over mobile processing units. As several operating systems and programming environments are available within a quite volatile market, one platform must be selected in order to explore its actual capacities for deploying a visually guided robotic application. This paper presents the on-going evolution of one project targeting the development of a real-time VS control toolkit under the iOS platform and its related tools.

This paper organized as follows: Section 2 provides information about PDA processing platforms while Section 3 depicts a particular discussion on the iOS platform including its application development environment. Section 4 presents a quick overview of VS while Section 5 presents the target VS application. Section 6 offers details about some simulation results and Section 7 discusses on some conclusions and the work ahead.

2 Mobile processing platforms

Following a wide heterogeneity on the mobile processing market, developing applications for a given selected platform still possess a clear challenge for any programmer. Cross-platform programming is still at a very early development stage and knowing more than two mobile programming languages up to an expert level, still demands a breath-taking exercise. However, under such circumstances, exploring the use of an integrated processing unit has become extremely attractive to robotics and its control requirements.

A quick overview on the available hardware platforms and its correspondent software libraries is presented by Table 1 as an overview of the information recently presented in [2]. Such table includes a reduced notation as follows: IDE = Integrated development environment, SDK =software development kit, ADT = Android development tools, JDE = Java development environment, PDK = Palm development kit and NDK = native-code development kit.

Table 1. Available programming platforms to date and their correspondent compatible hardware

OS	Runs on	Related to	Development languages	IDEs, Libraries, frameworks
Android	Open Handset Alliance	Linux	Java	Android SDK and NDK; ADT plug-in for Eclipse
BlackBerry OS	BlackBerry	Unix (BSD &NeXTstep)	Java	BlackBerry JDE
iPhone OS	iPhone, iPad, iPod Touch	Mac OS X	Objective-C / C++	iPhone SDK
PALM WEB OS	Palm Pre, Pixi	Linux	HTML, CSS, Java Script; C / C++	PDK; WebOS plug-in DK; Project Ares (Web based)
Symbian OS	ARM processors	Psion EPOC	C++, Java, other	
Windows Mobile	Windows Mobile smartphones	Windows CE	Visual C++	Windows Phone SDK (works with Visual Studio)

Table 2 shows a handy comparison between commercially available Smartphone platforms. It is easy to browse through their capabilities and services [4].

Table 2. Available smartphone platforms: one hardware comparison array

Platform	iOS 4	Android 2.1 with Sense	Symbian^3	WebOS	Windows Mobile 7
Processor	Apple A4	1GHz Qualcomm Snapdragon	680MHz ARM_{11}-based	600MHz TI $OMAP_{3430}$	1GHz Qualcomm Snapdragon
Storage	16GB / 32GB internal	440MB internal, microSDHC expansion	16GB internal, microSDHC expansion	16GB	Approx. 200MB internal,
Cellular	Quadband GSM, pentaband HSPA	CDMA, EV-DO Rev. A, WiMAX	Quadband GSM, pentaband HSPA	CDMA / EV-DO Rev. A	Quadband GSM, dualband HSPA
WiFi	802.11 b/g/n	802.11 b/g	802.11 b/g/n	802.11b/g	802.11 b/g[1]
Display size	3.5 inches	4.3 inches	3.5 inches	3.1 inches	4.3 inches
Display resol	960 x 649	800 x 480	640 x 360	480 x 320	800 x 480
Display tech.	IPS LCD	LCD	AMOLED	LCD	LCD
Primary camera	5 megapixel AF, LED flash	8 megapixel AF, LED flash	12 megapixel AF, xenon flash	3 megapixel, LED flash	5 megapixel AF, LED flash
Video record	720p at 30fps	720p at 24fps	720p at 25fps	VGA 30fps	VGA at 30fps
Location / orientation sensors	AGPS, compass, accelerometer, gyroscope	AGPS, compass, accelerometer	AGPS, compass, accelerometer	AGPS, accelero-meter	AGPS, compass, accelero-meter

The iPhone platform has been selected as primary hardware set thanks to its mature pervasive computing model which includes well consolidated hardware sensors, communication ports and processing units under one integrated operating system. Remarkably for robotics, most devices include image acquisition and video processing units despite a non-trivial programming language is required. The energy management hardware has been impressively redesigned with sufficiently long

service availability which reinforces its usage for robotic applications. Poor cross-platform code transportation still represent its main drawback as the iOS platform has developed its own programming language and developer kit, making practically impossible to transport code function from other programming platforms.

3 Apple's iOS platform

Apple mobile operating system, also known as iOs, has evolved to become the third most popular system accessing the Internet worldwide. It is also the third regarding any existing operating systems, recently surpassing Linux and lying just behind the popular MS Windows for desktop computers and its own relative, the desktop version of Apple's Mac OS [5].

The iOS platform works over an Apple-Samsung hardware set which includes a GPU, built-in RAM memory and a central ARM processor running over several velocities, depending on the hardware version. For instance, the iPhone© 3G employs an ARM processor at 412Mhz, while its predecessor 3GS includes an ARM Cortex A8 at 600Mhz. Apple more advanced iPhone©, version 4, includes an ARM processor running at 1GHz holding a double core architecture which is considered for future device versions over the Cortex A15 processor. A quick overview of A15 architecture is sketched below at Fig. 1, following the official manufacturer's data sheet [5].

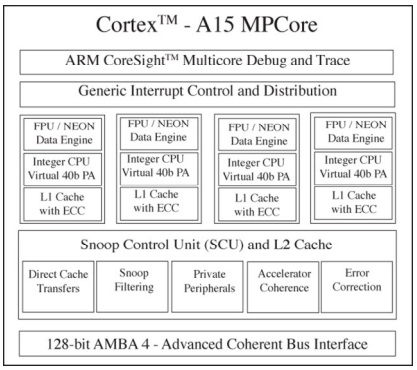

Fig. 1: Cortex A15 ARM microprocessor architecture [5].

The screen size is another important feature to be considered in vision-based robotics applications. The iPhone© platform includes a 480 x 320 pixel screen which has been regarded, until date, as the best voted graphic user interface, including file management and data base access from the same environment.

4 iOS Application Development

The development under the iOS standard follows the popular convention of Model-View-Controller which organizes the overall code by separating data regarding the user interface definition from the information controlling the application. The first component, known as the Model, validates access to all required data and their functionality. A second component, known as the View is basically concerned to define the user interface while the third component, known as the controller, replies to common user events that may eventually modify the goal-achievement management of the program.

The integrated development environment which natively serves the iOS programming is the Xcode©. The environment is a friendly-user interface that makes intensive use of graphic tools and manages different platforms under the same environment. Depending on the final target for the application, Xcode provides Navigation-based applications, OpenGL libraries, utility applications and view-based applications, among others. Fig. 2 shows a classical view of the Xcode integrated development environment containing one the seminal applications for this paper: gray-scale image conversion.

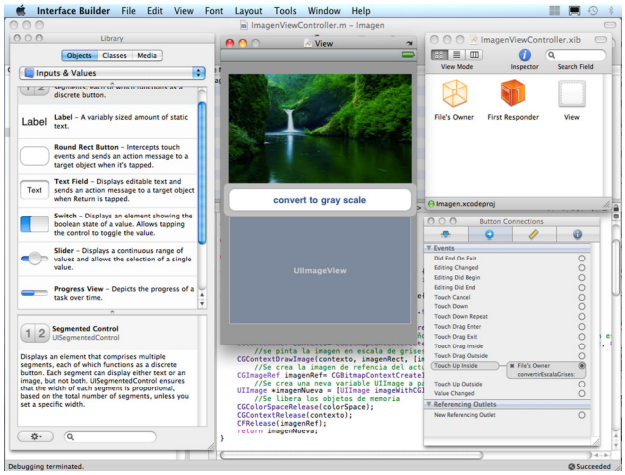

Fig. 2: Interface builder and related windows.

The Interface Builder is the code development hosting the designing of the user's visual interface and its final user interface. The builder includes a link to the code behind each control in the window.

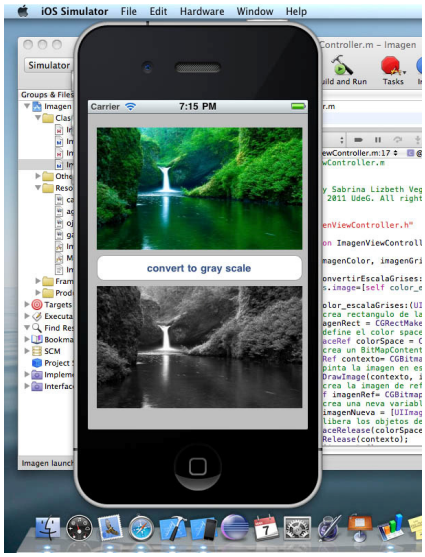

Fig. 3: Gray-level image converted working on the iOS device simulator provided by Xcode

Another important component on the iOS environment is the Device Simulator. Xcode includes a full device view for quick interaction to operating controls and output fields in the software. Fig. 3 shows the view for the gray-level program previously shown in Fig. 2.

4.1 Objective-C programming Program Code

The overall iOS programming environment requires its own programming language known as Objective-C which is a C-language-based meta-set. The Application Program Interface (API) includes high level functions and waste intelligent recollection features. The overall control structure is similar to C language, except by the fact that all objects are created and maintained over a dynamically appointed memory. Despite Objective-C is the main language for the platform, it may welcome other languages such as C#. One of the most important steps towards using the iPhone device for visual servoing applications is its ability to process several computer vision algorithms. A first important step is to generate the gray-level image which can be subsequently used by other processing algorithms [10]. An example showing the programming code for gray-level conversion is presented below.

```
//Gray-scale conversión.
-(UIImage *)convertGrayScale: (UIImage *) image {
   //One rectangle from the image is taken.
   CGRect imgRect = CGRectMake(0,0,image.size.width,image.size.height)
   //the color to be converted into gray-scales is chosen.
   CGColorSpaceRef gray = CGColorSpaceCreateDeviceGray();
```

```
    //A bitmap is created following
    CGContextRef context= CGBitmapContextCreate(nil,image.size.width,
image.size.height,8,0,gray,kCGImageAlphaNone);
    //image is converted into a gray-scale
    CGContextDrawImage(context, imgRect, [image CGImage]);
    //One reference image is created
    CGImageRef imgRef= CGBitmapContextCreateImage(context);
    //A new image is created from the new reference image.
    UIImage *imgNew = [UIImage imageWithCGImage:imgRef];
    //memory is freed
    CGColorSpaceRelease(gray);
    CGContextRelease(context);
    CFRelease(imgRef);
    return imgNew;
}
```

5 Target VS application

Visual Servoing (VS) is a mature research area which shares common issues with robotic control, real-time systems and in special with computer vision, involving subjects often employed in VS schemes such as active vision, visual pose estimation and dynamic vision [6]. Different implementations of VS have been traditionally identified within two main classifications: position-based and image-based visual servoing. An introductory overview of both classes is presented in the now classic VS tutorial in [7]. Basically, in the image-based visual servoing (IBVS) the error signal is computed in the image plane and the regulation commands are generated with respect to such error by means of a visual Jacobian. On the other hand, in the position-based schemes, the image features are used to estimate an object-workspace characterization in such a way that the error can be computed in the Cartesian space and used in the control loop.

Traditionally image-based systems have been regarded to possess a good robustness to calibration errors [7], even in the absence of the object and workspace model. However image-based VS schemes also exhibit some weaknesses such as singularities in the visual Jacobian which may lead to conflicts in the control loop. Other major drawback resides in the fact that an image-based system does not control the robot's end-effector in the Cartesian space, sometimes resulting in complicated or even unrealistic joint configurations being demanded to the robot. New IBVS schemes have been proposed to avoid these problems. Some of the new schemes combine 2-D and 3-D information and pose estimation to create a more complete visual servoing algorithm, for instance the 2-1/2 Visual servoing [7]. However these servoing schemes often require the description of object or its visual features.

In this paper, a low-profile anthropomorphic planar manipulator is equipped with a commercial CCD camera attached to its end-effector. An IBVS algorithm is used to achieve low-speed tracking of a moving object, which is model-free in the sense that no object model is provided [8]. Like other tracking problems, the aim is to keep the camera in a plane parallel to the tracked object. The objective is to investigate the use of mobile processing units to implement computational vision task in a first stage and the overall servoing control scheme in a second stage.

5.1 Experimental Set

The VS platform in this paper, requires a special but easy to conform hardware setup. The mobile telephone unit and one computer are required to implement the visual servoing system. Basically the smartphone performs all the artificial vision processing and computer controls the robot. The first one is known as the Eye Processor (EP) whereas the second one as the Visual Servoing Controller (VSC). Their tasks can be simple enumerated as follows: the EP provides a high level interface with camera drivers and other user interfaces to two visual tracking algorithms, the color-based and the optic-flow-based trackers. Also it provides the access point to the network communication subprocess which is in charge of establishing the high-speed link to the VSC computer through a reliable filtered link which includes a set of digital filters to overcome noise in the image and in the network link. On the other hand, the VSC computer provides the server for the network link, with an optional input filter to cancel out noise in the network hardware. Most VS schemes do not assume significant mechanical problems, low latency and high resolution on the driven robot. This is not always the case as this paper employs one low-cost TQ MA2000 6-DOF manipulator, designed primarily for teaching purposes. It has limited repeatability, low accuracy and sometimes high mechanical backlash. The planar manipulator is equipped with the iPhone 3GS attached to its end-effector, allowing the phone's camera to see a wide perspective of the robot's workspace. The iPhone's camera can provide high-quality (720p) and low-noise images but at the low rate of 30fps.

5.2 Camera Calibration

The intrinsic and extrinsic camera parameters should be experimentally calculated by using the camera model and calibration method [9]. The iPhone camera model is thus compacted into the classical C matrix:

$$C = \begin{bmatrix} f_u & \alpha_c f_u & c_u \\ 0 & f_v & c_v \\ 0 & 0 & 0 \end{bmatrix} = \begin{bmatrix} 1724 & 0 & 696 \\ 0 & 1720 & 722 \\ 0 & 0 & 0 \end{bmatrix} \tag{1.1}$$

With with $f_u$and $f_v$being the focal distance in pixel units whereas α_c=0, f_c, u_0 and v_0 represent the camera intrinsic parameters.

5.3 Experiment simulation

The control task can be described as follows: the TQ MA2000 Robot is resting over a flat surface. A circular miniature railway is placed within the robot's workspace, 0.10 meters away from its base. The camera registers four features of one replica train running over the track in order to perform the tracking. The train motion is simulated as a simple circular motion with a constant angular velocity as follows:

$$\begin{bmatrix} x_{train} \\ y_{train} \\ z_{train} \end{bmatrix} = \begin{bmatrix} 0.44\cos(-0.698t - 3/4\,\pi) + 0.54 \\ 0.44\sin(-0.698t - 3/4\,\pi) \\ 0.04 \end{bmatrix} \tag{1.2}$$

with r = 0.44 meters and the railway circle centered at (0.54, 0, 0). The phase value of $3/4\pi$ locates the initial position of the train in the right corner of the robot's workspace. The negative sign in the expression produces a clock-wise rotation. Notice that the train speed is chosen to be ω=0.112 m/sec. Using this value the train is able to ride through the arc of an angle of 45 degrees in 13.99 seconds.

6 Simulation results

The system is simulated through 13.99 seconds, which is the time required for the train to ride all the way in front of the robot. Fig. 4 shows the trajectory of the features in the image plane using a PI regulator with K_p = -2.0 and K_i = -0.1. Fig. 4 shows the difference between the target and current features in the image.

7 Conclusions and future work

This paper has presented the on-going research on developing a full set of VS libraries for mobile processing units such as smartphones or similar devices. To date, the work has focused on exploring few available platforms in the market, in particular the conveniences of the iPhone mobile set. However, much remains for future implementation such as the real-time operation and the cross-platform implementation of final VS libraries for mobile processing units.

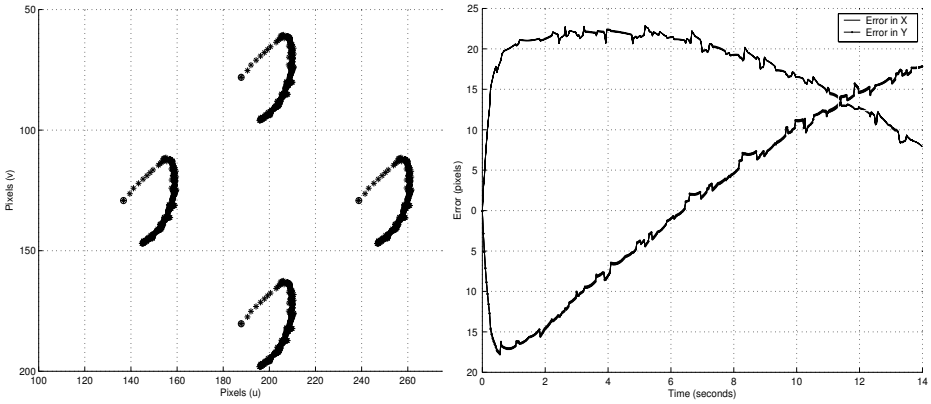

Fig. 4: Feature trajectory on the image plane (left) and the difference between the target and the current feature set in the image plane (right).

References

1. Lars Kulik, "Mobile Computing Systems Programming: A Graduate Distributed Computing Course", IEEE Distributed Systems Online, vol. 8, no. 5, 2007.
2. Daniel Derns.: Tools & Toys, Writing Small. In: IEEE Spectrum. June 2010
3. Gregory D. Abowd, Georgia Teach, Liviu Iftode, Helena Mitchell: The Smart Phone: A first Platform for Pervasive Computing. In: IEEE CS and IEEE ComSoc April-June 2005.
4. "Smartphone comparative table" http://www.iphonefanatic.net/apple/esto-si-es-una-tabla-comparativa-entre-smartphones", 2010
5. Cortex-A15 Processor, http://www.arm.com/products/processors/cortex-a/cortex-a15.php.
6. M A Perez-Cisneros & P A Cook, 2004, Open Platform for Real-Time Robotic Visual Servoing**,** Procs of 10th IASTED Int Conf on Robotics & Applications Hawaii, August, pp.142-147
7. Francois Chaumette, Seth Hutchinson, "Visual Servo Control Part I: Basic Approaches". IEEE Robotics and Automation Magazine, USA 13:4 (2006), 82-90.
8. C. Collewet and , F. Chaumette. "Positioning a camera with respect to planar objects of unknown shape by coupling 2-D visual servoing and 3-D estimations". IEEE Transactions on Robotics and Automation, 18(3):322-333, 2002.
9. Heikkilä, Janne and Silven, Olli. "A Four-Step Camera Calibration Procedure with Implicit Image Correction", IEEE Computer Society Conference on Computer Vision and Pattern Recognition, CVPR 1997, San Juan, Puerto Rico, pp. 1106-1112
10. Cuevas, E., Zaldivar, D. and Perez-Cisneros, M., "Procesamiento Digital de Imágenes con Matlab y Simulink", Book, written on Spanish, ISBN 9785478979738, RA-MA, SPAIN