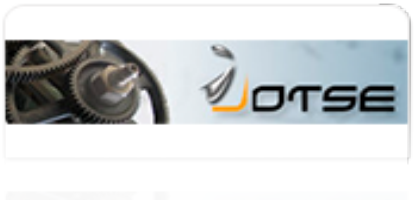


Journal of Technology and Science Education
ISSN: 2014-5349
info@jotse.org
OmniaScience
España

Cazorla, Miguel; Viejo, Diego
EXPERIENCES USING AN OPEN SOURCE SOFTWARE LIBRARY TO TEACH COMPUTER VISION SUBJECTS
Journal of Technology and Science Education, vol. 5, núm. 3, 2015, pp. 214-227
OmniaScience
Barcelona, España

Journal of Technology and Science Education

EXPERIENCES USING AN OPEN SOURCE SOFTWARE LIBRARY TO TEACH COMPUTER VISION SUBJECTS

Miguel Cazorla, Diego Viejo

University Institute for Computing Research, University of Alicante

Spain

miguel.cazorla@ua.es, dviejo@dccia.ua.es

Received December 2014

Accepted April 2015

Abstract

Machine vision is an important subject in computer science and engineering degrees. For laboratory experimentation, it is desirable to have a complete and easy-to-use tool. In this work we present a Java library, oriented to teaching computer vision. We have designed and built the library from the scratch with enfasis on readability and understanding rather than on efficiency. However, the library can also be used for research purposes.

JavaVis is an open source Java library, oriented to the teaching of Computer Vision. It consists of a framework with several features that meet its demands. It has been designed to be easy to use: the user does not have to deal with internal structures or graphical interface, and should the student need to add a new algorithm it can be done simply enough.

Once we sketch the library, we focus on the experience the student gets using this library in several computer vision courses. Our main goal is to find out whether the students understand what they are doing, that is, find out how much the library helps the student in grasping the basic concepts of computer vision. In the last four years we have conducted surveys to assess how much the students have improved their skills by using this library.

Keywords – Computer vision teaching, Open source, Engineering.

---

1 INTRODUCTION

Computer vision is an important subject in computer science degrees. For laboratory experimentation (practical teaching), we need a complete and easy-to-use tool. In this work we present a Java library, which is oriented to teaching (JavaVis, 2015). For this purpose, we have designed and built the library bent on readability and understanding rather than efficiency.

We have developed three different modules to meet the three different needs we have spotted as fundamental in our subjects related to machine vision. The first need is to develop a basic library to process images. The library allows easy sequence processing by incorporating geometrical data (edges, segments, points, etc.) and implementing several well-known computer vision algorithms. Furthermore, the design allows new methods and algorithms to be added to the library in a simple way, inviting the students to develop their own algorithms almost effortless. The second need is to apply the same working procedures as previously, but contemplating 3D data. The third need is to actually see the results. For this purpose, we have developed a visual desktop to visualize how the different algorithms work and what the results of combining them are.

Several works have explored the use of computational tools for teaching computer vision. A good review can be found in the work of Bebis, Egbert and Shah (2003). According to them, the use of a visual tool is essential when dealing with images. In this line, we have incorporated the visual desktop module to our library. Yapp and See (2008) explore the use of an interactive program for teaching image processing. However, the number of implemented computer vision algorithms is small. Martin and Chiang (2002) constitute another example of a limited system. So far, the number of implemented methods is limited; the system not only becomes closed but also focuses only on robotics. Chiu and Lee (2009) also explore the use of a visual application for teaching computer vision. Apart from these works, there are a lot of different computer vision libraries (Zuke & Umbaugh, 1997), (Khoral, 2015), (Vigra, 2015), (Jiggler, 2015), (Jai, 2015), (Javavision, 2015), (ImageJ, 2015). However, none of them present one of the main features of JavaVis which is its integration of 2D and 3D processing in a common framework.

Open Computer Vision Library (OpenCV, 2015) can be said to be the most popular computer vision library at the moment. It is available for Windows, Linux and Mac. It is distributed under Intel's license for both commercial and non-commercial (research and teaching) purposes. The library includes over 300 image analysis and processing methods from morphology, geometry, image treatment … to the recently added methods for computing stereoscopic correspondence, face recognition and 3D tracking. OpenCV is one of the most complete and efficient computer vision libraries, yet it does not incorporate a Graphic User Interface (GUI) to visualize and evaluate results. Furthermore, it is not teaching oriented and so its learning curve (how long it takes the student to be able to work comfortably --understand the library-- with the library) is high. OpenCV may integrate another library: Point Cloud library (PCL, 2015). This library is implemented in C++, but it can be used with other programming languages. It also implements state-of-the-art 3D methods, but for teaching purposes its GUI is not good enough. As is the case with OpenCV, its learning curve is too high.

Java, on the other hand, has been used profusely, proving its validity and multi-platform support of the language. Javenga (Baloukas, 2012), which is a visualization environment based on Java proves useful in teaching concepts related to network and graph algorithms: a Java platform has recently been developed by Jara, Candelas, Pomares and Torres (2011) for teaching robotics.

The rest of the paper is organized as follows: Section 2 enumerates the subjects where JavaVis are currently being used. Then, Section 3 describes the pedagogical issues to be addressed with this library. In Section 4, we describe the main features of JavaVis. Inside this section, we detail the three different parts of JavaVis: JavaVis2D for processing 2D images, JavaVis3D for 3D data, and JavaVisDesktop a visual desktop tool used to learn and build machine vision algorithms. Section 5 shows first the teacher perspective and then, relates the students' experience using JavaVis together with some analytic data from questionnaires. Finally, we draw our conclusions and propose future work lines.

2 COURSES DESCRIPTION

JavaVis is currently being used in the teaching of several subjects in Computer Science degree at Alicante University in Spain. It was previously used in the delivery of some subjects in master and predoctoral degrees. Since 2001 we have been using JavaVis as a framework to teach Computer Vision in related courses. We shortly describe all of them:

- Artificial Intelligent Techniques: This subject consists of one hour and half lecture and one hour and half lab session per week for fifteen weeks. Eight weeks of the course are dedicated to computer vision and the remaining to machine learning. In the part dedicated to Computer Vision, we use the tool presented in this paper as the basis for carrying out the lab assignments. Sometimes we are able to use JavaVis in the whole course.
- Computer Vision: This subject consists of two-hour lectures and two-hour lab sessions per week for fifteen weeks. It is dedicated to 3D vision, object recognition, image segmentation, feature extraction, etc. JavaVis is used for laboratory experimentation throughout the course.
- BS Degree course: Students tend to favour to complete projects related to computer vision; some of them are implemented with this tool. In fact, the library began its development as a BS Degree project. Throughout the years JavaVis has been used in more than 20 projects, which in turn have contributed with code and new ideas to the library.
- Postgraduate courses: JavaVis is used as a supporting tool in several Master subjects on Computer Vision and Image Processing. It has been also used in several PhD theses.

When students complete any of these courses, they inevitably acquire sound programming skills. However, the concepts related with Computer Vision can sometimes be so complex that it is necessary to have a tool easy to learn and use.

3 PEDAGOGICAL ISSUES

Javavis library is a teaching tool that we use as teachers to implement our methodological proposals. There are many different teaching tools available, each one oriented to a particular learning activity. In our case, our main pedagogical goal is to serve as a supporting tool for the laboratory experimentation. As a teaching tool, the library must satisfy several fundamental requirements:

- It must facilitate the learning/teaching process.
- It must contribute to an enjoyable but rigorous learning.
- It must help students to learn (and successfully pass) the course.
- It must motivate students to do research on its applications.
- It must enable the interaction between the teacher and the students.

In our opinion, JavaVis satisfies all of these requirements, becoming a powerful tool to support the learning and teaching process.

- The aim of the learning/teaching process is the student training. So, there are two agents involved in this process (the teacher and the student), dealing with some contents that must be taught (learned) and some resources that must be taught (to learn them). JavaVis eases this process to a great extent. On the one hand it helps the teachers in their pedagogical task because it is an easy to use tool and it provides a friendly graphical environment. On the other hand, it improves the learning task: the student can clearly understand the algorithms thanks to the code inspection. For the learning process to be constructive, the student must analyse and construct it. As a consequence it will bring a real assimilation of the knowledge in the long term instead of a mere weak and short-term learning.
- A lot of computer vision algorithms involve complex mathematical concepts. But our students are not studying Mathematics as a degree. So we must try to combine a rigorous learning process with an easy and helping way. The library promotes this goal because the students can clearly see the implemented algorithms, which are explained in the theoretical session. This way they can better understand concepts and see how the algorithms work.
- JavaVis includes several practical applications, most of them being the result of past courses projects. This fact motivates the student, because the students realize that other students in the past have developed parts of the library. In fact, several students (one or two per year) have been involved in developing some important parts of JavaVis.

In short, the use of this tool not only serves to strengthen the practical skills of the subject but also substantially strengthens the theoretical understanding and, thus, the overall understanding of the subject.

4 USING JAVAVIS FOR TEACHING COMPUTER VISION

A subject on computer vision usually includes the study of basic concepts together with algorithms to solve related tasks. A brief description of this kind of subject could be:

- Image formation. Cameras (calibration, geometry), image representation, colour and light.
- Image Enhancement. Histograms, smoothing and noise reduction, gradients.
- Feature extraction. Edges, corners, segmentation, high level description.
- 3D images. Stereo, low cost sensors, homography.
- Object recognition. Mapping, bag of features, face detection

JavaVis has been primarily developed to assist in the teaching of computer vision. This kind of concepts are, usually, hard to understand for the students. For example, gradients, which are defined as a derivative of the image function, are rendered by just a pixel difference. The student understands the concept better when he/she sees the implementation of the theoretical concepts.

In JavaVis, we have developed three different modules, based on three different needs we have discovered in our surveys across the subjects related. The first one is the need of a basic library to process images. Next, a special image format to enable easy sequence processing. For this purpose, geometrical data (edges, segments, points, etc.) are incorporated and several well-known computer vision algorithms are implemented. The students can, therefore, develop their own algorithms with ease --the second module consistently applies the same working schema of the first one, but 3D data is applied. The final need is that of providing a visual desktop to visualize how the different algorithms work and the results obtained by combining them. The use of Java is justified as our students use different operating systems. Java is multiplatform and allows us to give a program, which can run in all of different operating systems.

In the next subsections, we briefly describe the three modules. Figure 1 shows the GUI for the three modules.

4.1 JavaVis2D

This part refers to traditional computer vision and, in fact, it was our first step. We could import different image formats even though traditional computer vision has its own. In JavaVis2D each image is composed of one or several frames. A frame represents an image that can be either bitmap or geometric. A bitmap frame is an image represented as a matrix in which each element is a pixel (picture cell). JavaVis features five types of bitmap frames: Bit (0,1), Byte (0..255), Short (0..65525), Float (float range of Java) and Colour (three bands of Byte type each). In addition, each bitmap frame can be formed by one or several bands. Note that each band must have the same dimensions. Internally, each type stores a one-dimensional array of data to be more efficient. For example, the Bit type is stored using a Boolean Java type, the Byte type uses the Java type byte. When accessing a pixel, the data is converted from its original value: for example, the byte type in Java is signed but the Byte type of JavaVis is unsigned. Conversion is done internally. A sequence can be organized in two ways: several frames in a sequence, or several bands per frame, which can also form a part of a sequence. However, to be able to insert several bands in the same frame, every band must have the same size and be of the same type whereas different frames can have different size, type and number of bands. A geometric type frame, on the other hand, manages information differently. For example, the Segment frame type just stores the coordinates of the initial and end points of a segment. A Segment frame may have several segment objects stored not into a pixel matrix but into a segment object list. So the representation of geometric frames is not only more compact but its computations are faster. This kind of frames is useful in several computer vision algorithms.

An important point in JavaVis2D is the organization of its functions (implementation of algorithms). This organization becomes crucial for other users to be able to easily implement their algorithms in a standardized way. In order to implement an algorithm, the function code needs to be developed and its input and output parameters specified. This is precisely one of the main goals of the library: implement once, use anywhere. When an algorithm is implemented, then the library is in charge of input and output parameters checking, showing images through the GUI, and so on.

Currently, there are more than 60 state-of-the-art computer vision algorithms already implemented. For example, AddNoise, Rotate, Mirror, ColorToGray, RGBToColor, etc. provide us with functions for manipulation and colour transforming. ConvolveImage, Gabor for convolution, Equalize, Smooth, Brightness for image adjustment, Binarize, Kmeans, MeanShift for segmentation, InterSegment, RandomPoint for geometry manipulation, Canny (Canny, 1986), Susan (Smith, 1997), Grad) for edge extraction, Junctions (Cazorla, 2000), HoughLine (Ballard, 1981), MSER (Matas, Chum, Urban & Pajdla, 2002) for feature extraction, Erode, Clousure for mathematical morphology and others (Skeleton, Pca, Manhattan).

4.2 JavaVisDesktop

Our goal in this project is to build a utility which serves to better understand the partial results when processing an image. We can use different algorithms to get a complex one: it is by means of the Desktop utility that we are allowed to check partial results, by showing the images obtained after applying an algorithm.

This utility allows the building a sequence of functions as a finite-state machine. Each state in the sequence is a function (algorithm) from JavaVis. A state shows the result (an image) of applying such algorithm. Thus, the student can easily adjust different parameters of an algorithm, observing what the consequence of modifying that parameter in the complete sequence involves. Once the parameters are adjusted, there is an option to

generate a new function in JavaVis2D directly. This new function will contain the sequence of functions included in the desktop reflecting the selected parameters.

4.3 Java3D

Since the Kinect device was introduced, the use of 3D data and its algorithms have been more popular. JavaVis3D module allows the representation and manipulation of 3D data sets from any 3D sensor, such as stereo cameras, RGB-D sensors, 3D laser, etc. Most of the basic 3D geometric entities have been implemented in the core of JavaVis3D. Points, vectors, normals, segments and planes classes are included in a geometric package that allows JavaVis3D to operate with them, applying transformations, performing input/output operations, etc. We can load a 3D point set captured from a scene using a stereo camera or Kinect device, compute these points to obtain the normal vector for the scene surfaces at every point and finally store this set of normal vectors in a file to be used in the future.

JavaVis3D includes a GUI for showing the 3D data. This GUI is built through the 3D Java API Java3D. Java3D can be found for almost every operating system. 3D Data sets can be loaded/unloaded into the GUI. The user can freely transform the observer's point of view, the virtual camera so that he/she can have an adequate view of the scene. JavaVis3D GUI also provides an easy-to-use method for launching different 3D functions over the 3D data. We have also implemented some of the state-of-the-art 3D algorithms in JavaVis3D such as Iterative Closest Point (ICP) (Pomerleau, Colas, Siegwart & Magnenat, 2013) to compute the transformation that best aligns any two sets of 3D points.

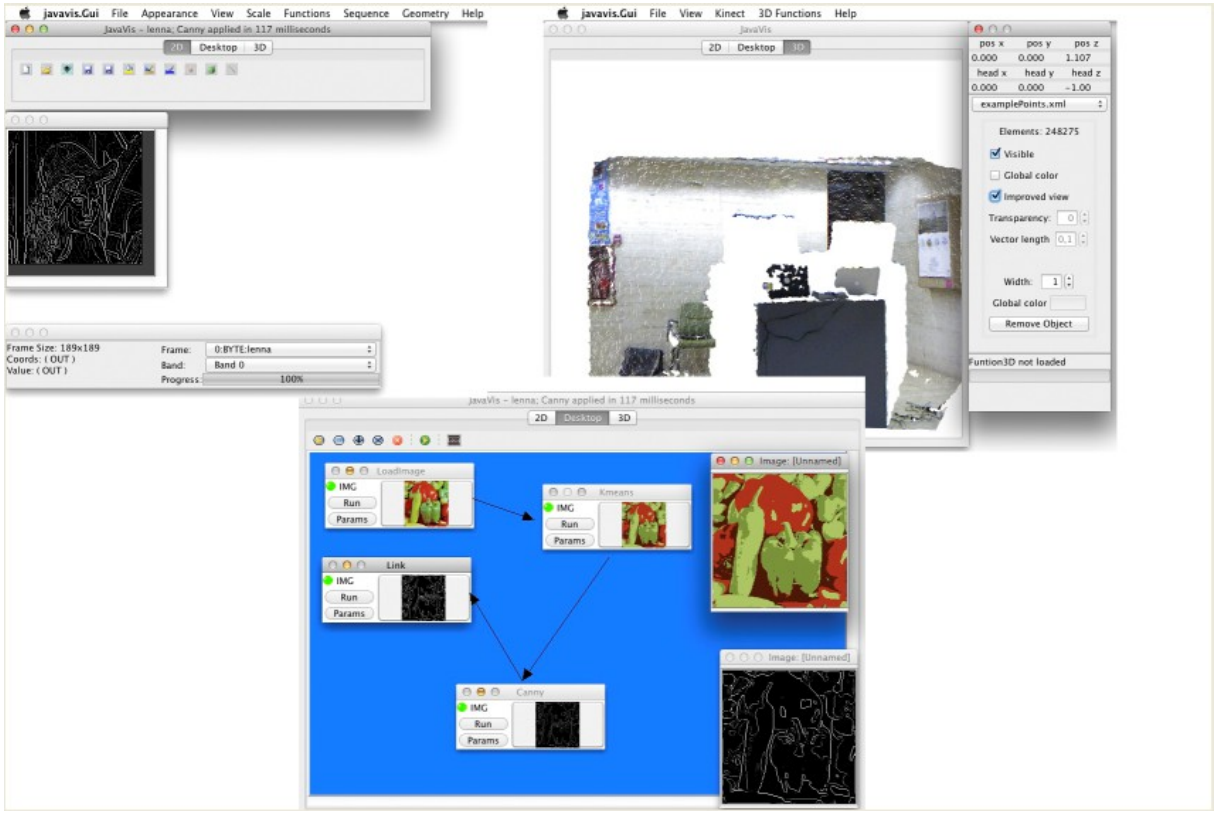


Figure 1. GUI for JavaVis. Left, JavaVis2D; bottom, JavaVisDesktop; right, JavaVis3D

5 TEACHING EXPERIENCE

JavaVis library has been used for 10 years as a tool to teach computer vision in the subjects detailed in Section 2 at the University of Alicante (Spain). JavaVis has let us work at two different levels. First of all, it has allowed us to show some of the most useful and popular computer vision algorithms, as many of them are already implemented in JavaVis. At this level, students observe what happens when the algorithm already explained in the lecture is applied. Furthermore, they can take a look at the implementation of the algorithms, also explained in the lectures, thus obtaining a deeper understanding. However, just by using algorithms, they are not learning the complexity of the implementation, i.e. how much time it takes to implement. So, the second level follows. Students can try to develop their own algorithm. At this level we aim at simplicity. Students do not have to know much about the library. What they must know the classes and methods for manipulating an image, but nothing about graphical classes or file structure.

During these four years, we have proposed several practical assignments. We have followed a kind of project-based learning strategy, but only in the practical part of the subject. Students are grouped in groups of 3-4 people. A problem to be solved is presented to the groups. For example: we need a system that must be able to count the number of coins in a conveyor belt. This belt has a camera that provides images of the belt. The expected output is the number of coins at a given moment on the belt. Summarizing, each group needs to define the problem and propose a viable solution, taking into account all the possible problems (changing illumination, presence of others circular objects, occlusions, etc.). Some of the groups propose a similar solution, based on the use of methods already implemented in JavaVis and some additional code. First, students have to develop a non-implemented algorithm: Hough transform applied to circumferences. This function is added to the library. Then, a new function is created. This new function calls for several functions: Canny (to obtain contours), Binarize (to binarize the Canny output), HoughCirc (the last function developed, applied to Binarize output) and, finally, we have to count the number of pixels inside the circumferences detected. This last step let us eliminate false coins --those which do not have enough pixels—in the domain defined by the coin and, having the same colour as the coin. Figure 2 shows an example of the execution of this assignment.

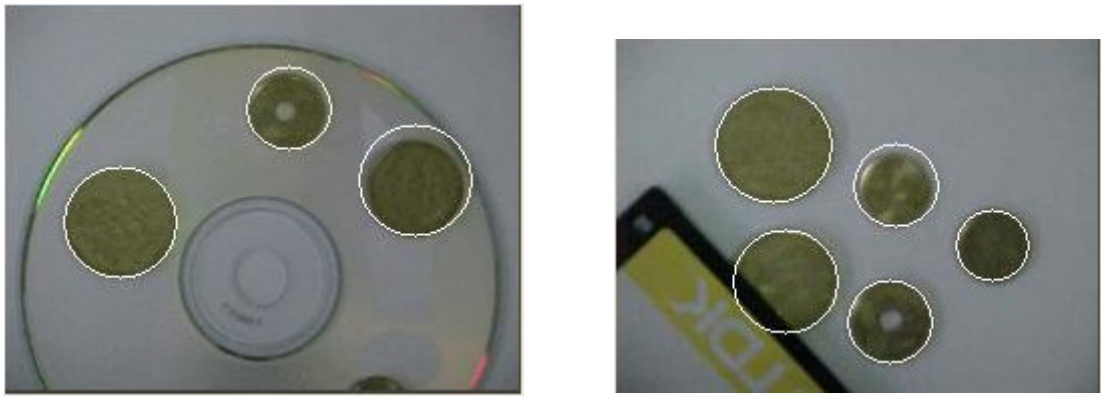


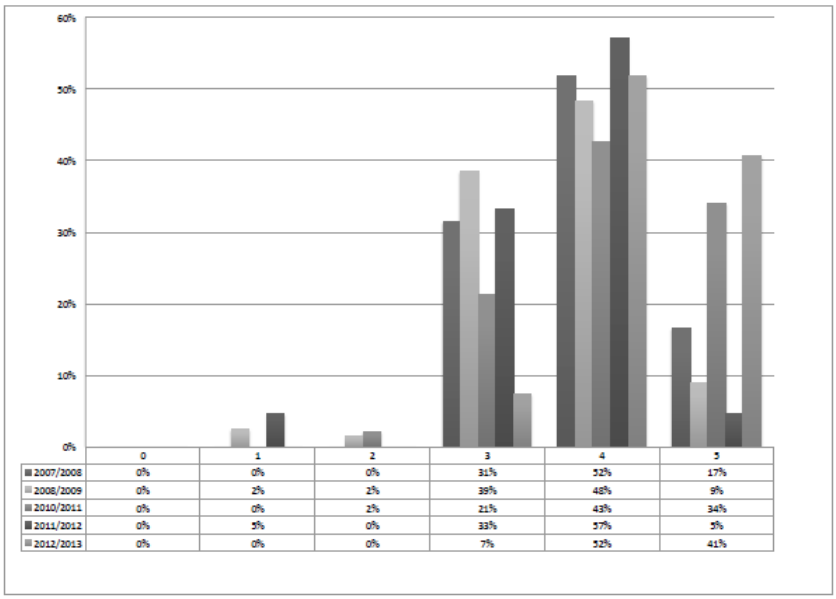
Figure 2. Two examples of the results of the lab assignment: counting coins. The white circumferences are the coins detected. Note that the CD is not detected as it has a different colour, and the inner circle of several coins is not detected either

5.1 Student opinion

During the terms from 2007/2008 all through 2012/2013, we conducted a survey from where we gathered information from the students who used JavaVis for their laboratory assignments. In 2007/08, 54 students answered the questionnaire, in 2008/09, 122, in 2010/2011, 47, in 2011/2012, 30 and finally in 2012/13, 27. We wanted to know how JavaVis helped them in their overall lab assignment performance and in grasping theoretical concepts. We also wanted to know if JavaVis was the tool we purported to be easy to understand and use.

We now show questions asked to the students and detail the results. The results from the student answers are shown in the referenced figures. For the first two questions, the students were required to answer them by selecting a value from 0 (beginner) to 5 (expert). For the rest of questions, the answers ranged from 0 (strongly disagree) to 5 (strongly agree).

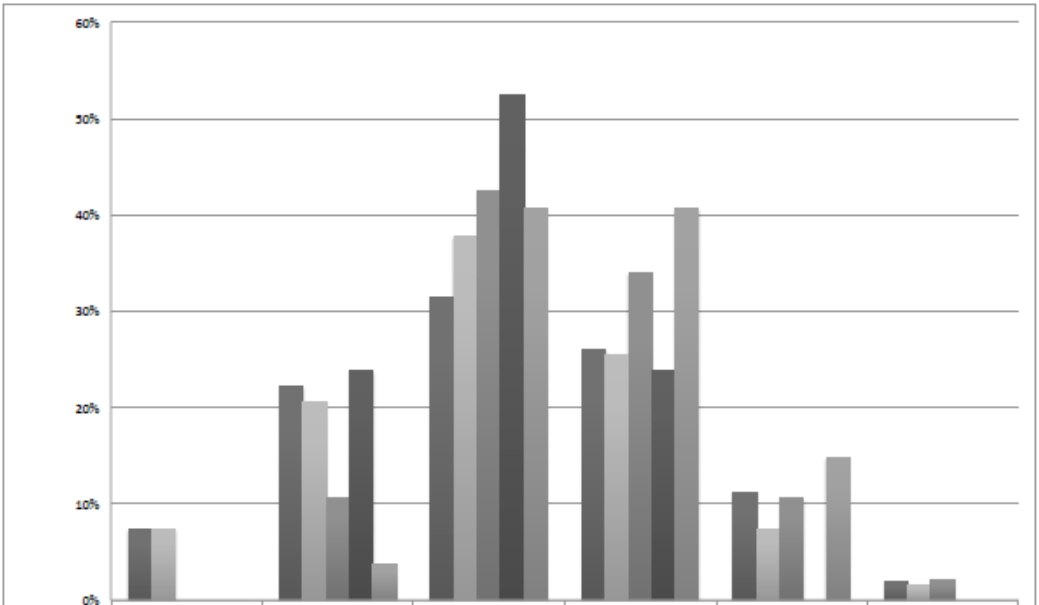
- *Your knowledge of Java programming language (syntax, arrays, lists, etc.)*. The results showed that almost all the students had an elementary knowledge of the Java programming language with a mean value around value 4. This allowed us to use Java as a programming language. Far from being a trivial question, it proved not only that the use of Java was not widely spread but also that an introductory session was called for.



	0	1	2	3	4	5
2007/2008	0%	0%	0%	31%	52%	17%
2008/2009	0%	2%	2%	39%	48%	9%
2010/2011	0%	0%	2%	21%	43%	34%
2011/2012	0%	3%	0%	33%	57%	3%
2012/2013	0%	0%	0%	7%	52%	41%

Figure 3. Your knowledge of Java programming language (syntax, arrays, lists, etc.)

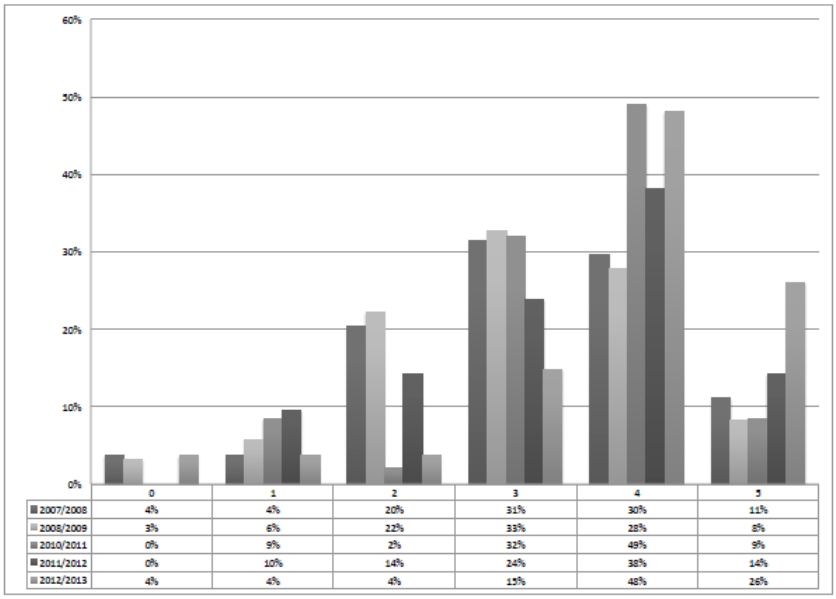
- *Your knowledge of advanced Java programming language (threads, graphic user interface, etc.)*. With a mean in 2.1, the first two terms and the fourth one and above 2.5 in one intermediate term together with the last one, we learnt that not all the students knew much of GUI API, threads and so on. Then, we desestimated introducing lab assignments in which students had to work with this programming language because it required an additional effort on their part. This situation created the need to build JavaVis as a tool to bypass advanced programming. The students, then, could entirely focus on implementing the algorithm.



	0	1	2	3	4	5
2007/2008	7%	22%	31%	26%	11%	2%
2008/2009	7%	20%	38%	25%	7%	2%
2010/2011	0%	11%	43%	34%	11%	2%
2011/2012	0%	24%	52%	24%	0%	0%
2012/2013	0%	4%	41%	41%	15%	0%

Figure 4. Your knowledge of advanced Java programming language (threads, graphic user interface, etc.)

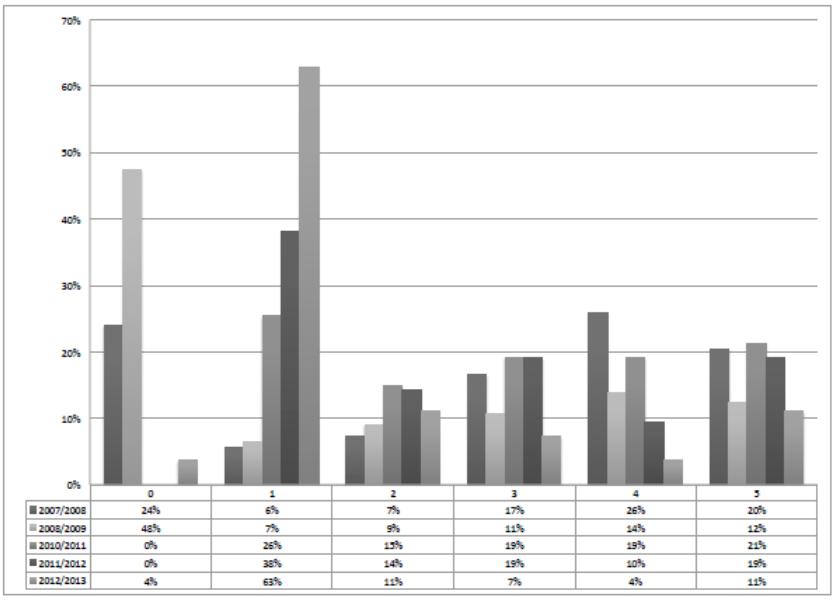
- *Laboratory assignments require a solid understanding of the Java programming language*. In this question, we wanted to perceive the students' general feeling from towards the language. The mean value was around 2.5, indicating that the students did not need any advanced knowledge of Java language, which has been our goal throughout.



	0	1	2	3	4	5
2007/2008	4%	4%	20%	31%	30%	11%
2008/2009	3%	6%	22%	33%	28%	8%
2010/2011	0%	9%	2%	32%	49%	9%
2011/2012	0%	10%	14%	24%	38%	14%
2012/2013	4%	4%	4%	15%	48%	26%

Figure 5. Laboratory assignments require a solid understanding of the Java programming language

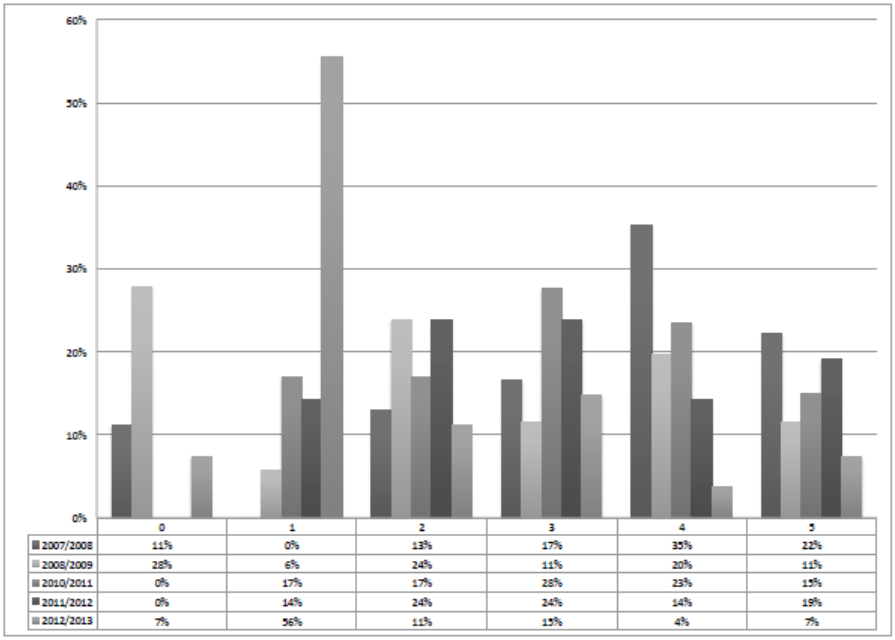
- *You have used the JavaVis Desktop framework to carry out laboratory assignments*. Here the answers from one of the terms differ from the others. In 2007/2008, 2010/2011 and 2011/2012 the lab assignments were more difficult than the ones assigned in 2008/2009 and 2012/2013. JavaVis Desktop proved to be more appropriate in 2007/2008 as the answers show. Even though the application of JavaVis Desktop was not a requisite for the lab assignments, the students made use of it. It should be noticed that the use of Desktop was not mandatory.



	0	1	2	3	4	5
2007/2008	24%	6%	7%	17%	26%	20%
2008/2009	48%	7%	9%	11%	14%	12%
2010/2011	0%	26%	15%	19%	19%	21%
2011/2012	0%	38%	14%	19%	10%	19%
2012/2013	4%	63%	11%	7%	4%	11%

Figure 6. You have used the JavaVis Desktop framework to do lab assignments

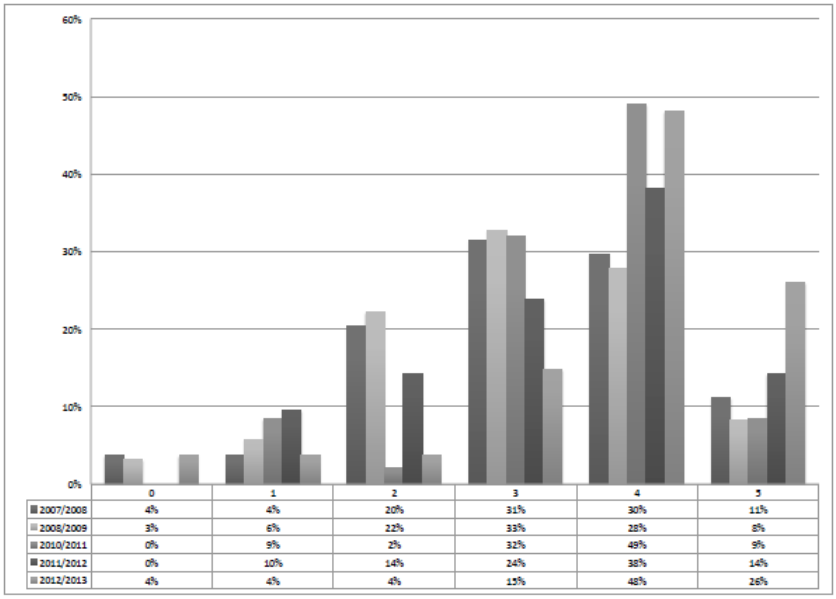
- *JavaVis Desktop framework is useful for laboratory assignments*. As in the previous question, there are differences in the answers from term to term. In 2008/2009 and 2012/2013 many students did not apply the Desktop, so their answers reflected 0 for almost half of them. Nevertheless, the students generally conceded that the framework was definitely useful.



	0	1	2	3	4	5
2007/2008	11%	0%	13%	17%	35%	22%
2008/2009	28%	6%	24%	11%	20%	11%
2010/2011	0%	17%	17%	28%	23%	15%
2011/2012	0%	14%	24%	24%	14%	19%
2012/2013	7%	56%	11%	15%	4%	7%

Figure 7. JavaVis Desktop framework is useful for lab assignments

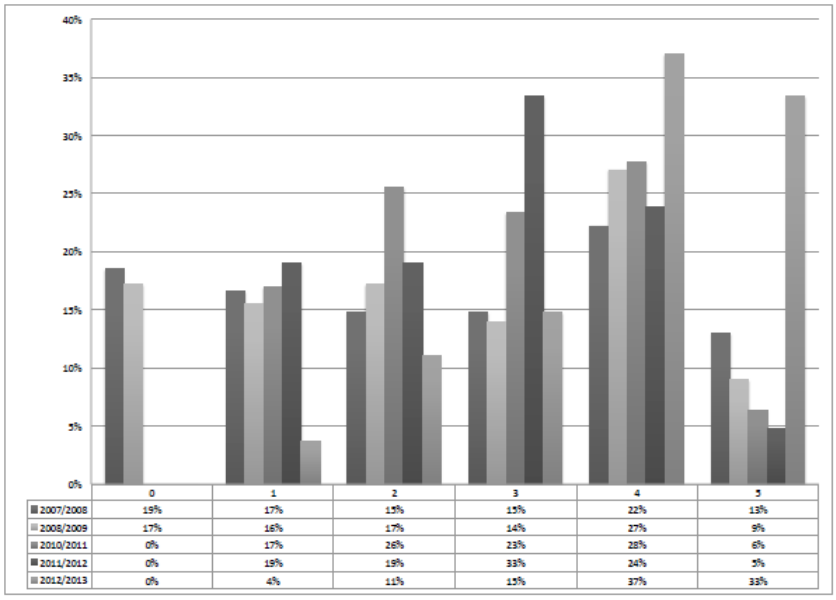
- *JavaVis did help you to get a deeper understanding of the concepts*. In this case, all the courses agreed: JavaVis helped them get a deep insight of the course contents. The mean value was 3.34, surprisingly higher than that obtained for 2012-2013.



	0	1	2	3	4	5
2007/2008	4%	4%	20%	31%	30%	11%
2008/2009	3%	6%	22%	33%	28%	8%
2010/2011	0%	9%	2%	32%	49%	9%
2011/2012	0%	10%	14%	24%	38%	14%
2012/2013	4%	4%	4%	15%	48%	26%

Figure 8. JavaVis did help you to get a better grasp of the concepts

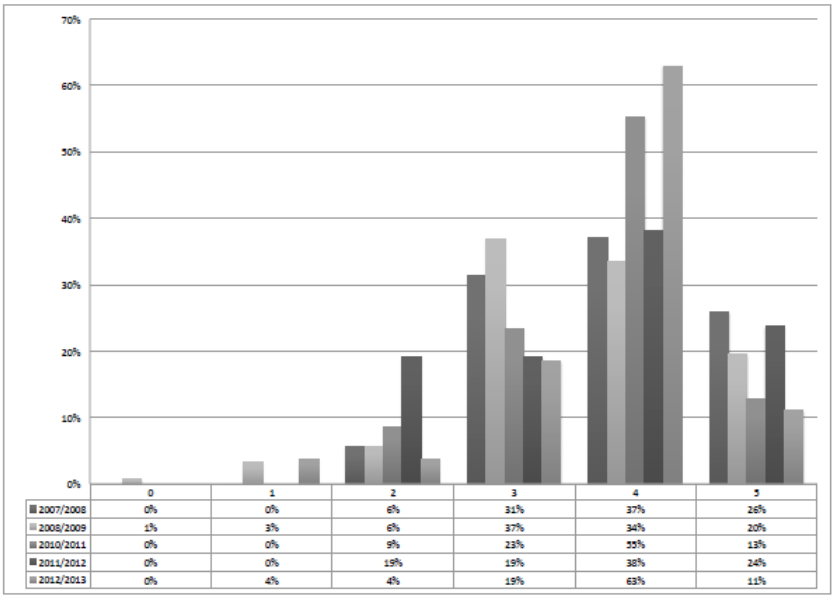
- *You have seen how some JavaVis algorithms are implemented in the various methods explained in the lectures*. We did not contemplate (and consequently, did not ask) the students to look inside the implementations of the algorithms, but more than half of them did. The tendency appears to be that more and more students are tempted to scrutinize the algorithms to understand theoretical concepts.



	0	1	2	3	4	5
2007/2008	19%	17%	15%	15%	22%	13%
2008/2009	17%	16%	17%	14%	27%	9%
2010/2011	0%	17%	26%	23%	28%	6%
2011/2012	0%	19%	19%	33%	24%	5%
2012/2013	0%	4%	11%	15%	37%	33%

Figure 9. You have seen how JavaVis algorithms are implemented in the various methods explained in the lectures

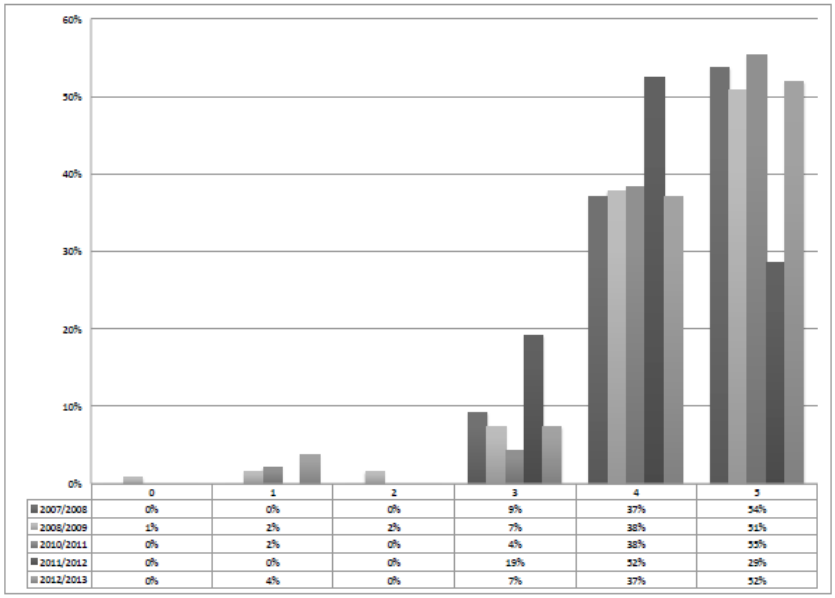
- *JavaVis is a tool easy to learn.* In this case, the answer favoured JavaVis. The students coincided in that JavaVis was easy to learn. They could develop their lab assignments with ease. Furthermore, the adjustments on JavaVis on a yearly basis reveal the students' acceptance.



	0	1	2	3	4	5
2007/2008	0%	0%	6%	31%	37%	26%
2008/2009	1%	3%	6%	37%	34%	20%
2010/2011	0%	0%	9%	23%	35%	13%
2011/2012	0%	0%	19%	19%	38%	24%
2012/2013	0%	4%	4%	19%	63%	11%

Figure 10. JavaVis is a tool easy to learn

- *JavaVis is useful for laboratory assignment implementation*. For this question we obtained the greatest support. Students agreed that this tool was definitely useful. According to them, computer vision had been simplified by JavaVis, with a mean value close to 4.5 at each year.



	0	1	2	3	4	5
2007/2008	0%	0%	0%	9%	37%	54%
2008/2009	1%	2%	2%	7%	38%	51%
2010/2011	0%	2%	0%	4%	38%	55%
2011/2012	0%	0%	0%	19%	52%	29%
2012/2013	0%	4%	0%	7%	37%	52%

Figure 11. JavaVis is useful for laboratory assignment implementation

The conclusion we draw is precisely what we expected: there is enough knowledge of basic Java programming, but a low level of advanced concepts. As the advanced skills (graphical interface, threads, etc.) are necessary for some computer vision tasks, it is a good idea to provide a framework to provide abstractions of those skills. Students used JavaVis as a tool not only to carry out the lab assignments but also to learn theoretical concepts.

6 DISCUSSION

Teaching experience is covered from two points of view. On the one hand, we take in account our teachers' experience in computer vision subjects. It allows us to stablish the basis of the proposal computer vision library. In this way, teaching experience tell us that it is worthy to make JavaVis an easy to use library for the development of new computer vision applications. The main efforts are then focused on offer an abstraction layer that frees the students from tasks such as image management, application layouts and events, or multi-thread programming. Thus, the teachers involved in computer vision subjects in which JavaVis is being used have reported that the use of the library makes easier the teaching of the subject as well as its assimilation by students.

On the other hand, it is also important to know the opinion of the students who have developed their practical assignments using JavaVis. Thus we made a questionnaire over five years to learn how students perceived the provided tool. The questionnaire was focused on understanding different issues related to level of the students before and after taking the course both in relation to their programming skills and their knowledge of machine vision. In this way we could confirm that the programming skills of most students before taking the course is enough to develop a simple to medium application. However, they often lack the advanced skills that would enable them to cope with programming multi-threaded user interface like the one they are provided with. As for the advantages of using the library, students positively valued the tool as a whole. In addition, most students consider that the use of the library has been useful for assimilating the concepts studied in the course. They themselves realize the advantage to have available an open source implementation of the algorithms discussed in the lectures. Finally, for most students the use of JavaVis has made easier the resolution of the problems they faced.

7 CONCLUSIONS

The contents of the subjects in computer vision could be tough for computer science students. In fact, the subjects dealing with computer vision are usually placed in the last years of an engineering degree. As the students' surveys indicate, the teaching/learning process of computer vision could be assisted using open source libraries or tool kits. This applies to not only the practical issues but also the theoretical ones the subject involves. The students are able to break the code open to better understand how an algorithm works. Furthermore, the JavaVis Desktop is another useful tool for understanding the workflow of a complex algorithm.

For this reason, we have developed a new framework, JavaVis, which is teaching oriented. It is written in Java and is Open Source. JavaVis is divided into three parts. First, JavaVis2D, which enables 2D-image processing. It features image format and geometrical data. Then, JavaVis3D, which supports 3D data, —3D points, planes, trajectories, etc. Finally, JavaVis Desktop. It enables the students to understand changes in parameters and develop new algorithms with certain ease.

JavaVis has been used for several years in several computer vision-related subjects at University of Alicante (Spain). We are presenting a practical example of JavaVis in this paper. We also discuss the results from the different surveys conducted for a number of terms and we have reached the conclusion that JavaVis has, in fact, proved useful for the students and helped them gain insight of theoretical concepts.

We can conclude by saying that an Open Source library serves the students as a launchpad to keep on learning by themselves.

As future work we are planning to continue incorporating new computer vision algorithms and the features we identify in our teaching. We want also to use JavaVis in conjunction with OpenCV and PCL.

REFERENCES

Ballard, D.H. (1981). Generalizing the Hough Transform to Detect Arbitrary Shapes. *Pattern Recognition*, 13(2), 111-122. http://dx.doi.org/10.1016/0031-3203(81)90009-1

Baloukas, T. (2012). Javenga: Java-based visualization environment for network and graph algorithms. *Computer Applications in Engineering Education,* 20, 255-268. http://dx.doi.org/10.1002/cae.20392

Bebis, G., Egbert, D., & Shah, M. (2003). Review of computer vision education. *IEEE Transactions on Education*, 46, 2-21. http://dx.doi.org/10.1109/TE.2002.808280

Canny, J. (1986). A Computational Approach To Edge Detection. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 8(6), 679-698. http://dx.doi.org/10.1109/TPAMI.1986.4767851

Cazorla, M. (2000). *Un enfoque bayesiano para la extracción de características y agrupamiento en visión artificial*. Tesis doctoral. Available online in: http://www.cervantesvirtual.com/obra/un-enfoque-bayesiano-para-la-extraccion-de-caracteristicas-y-agrupamiento-en-vision-artificial--0/

Chiu, C.F., & Lee, G.C. (2009). A video lecture and lab-based approach for learning of image processing concepts. *Computers & Education,* 52, 313-323. http://dx.doi.org/10.1016/j.compedu.2008.09.003

ImageJ (2015). *The imagej website*. Available online in: http://rsbweb.nih.gov/ij/

Jai (2015). *The Java advanded imaging website*. Available online in: https://java.net/projects/jai

Jara, C.A., Candelas, F.A., Pomares, J., & Torres, F. (2011). Java software platform for the development of advanced robotic virtual laboratories. *Computer Applications in Engineering Education,* 21(S1), E14-E30. http://dx.doi.org/10.1002/cae.20542

JavaVis (2015). *JavaVis web site*. Available online in: http://javavis.sourceforge.net.

Javavision (2015). *The Java vision toolkit website.* Available online in: http://javavision.sourceforge.net/

Jiggler (2015). *Java imaging and graphics library*. Available online in: http://jiggler.sourceforge.net/

Khoral (2015). Available online in: http://www.khoral.com.

Martin, J.F., & Chiang, L. (2002). Low cost vision system for an educational platform in artificial intelligence and robotics. *Computer Applications in Engineering Education,* 10, 238-248. http://dx.doi.org/10.1002/cae.10026

Matas, J., Chum, O., Urban, M. & Pajdla, T. (2002). Robust wide baseline stereo from maximally stable extremal regions. *Proc. of British Machine Vision Conference*, 384-396. http://dx.doi.org/10.5244/c.16.36

OpenCV (2015). *Open source computer vision library*. Available online in: http://opencv.willowgarage.com/wiki/

PCL (2015). *Point cloud library*. Available online in: http://pointclouds.org/.

Pomerleau, F., Colas, F., Siegwart, R., & Magnenat, S. (2013). Comparing ICP Variants on Real-World Data Sets. *Autonomous Robots*, 34(3), 133-148. http://dx.doi.org/10.1007/s10514-013-9327-2

Smith, S.M. (1997). *Method for digitally processing images to determine the position of edges and/or corners therein for guidance of unmanned vehicle*. UK Patent 2272285. Proprietor: Secretary of State for Defence, UK. 15 January 1997.

Vigra (2015). *The vigra library website.* Available online in: https://github.com/ukoethe/vigra

Yapp, C.H.W., & See, A.K.B. (2008). Teaching image processing: A two-step process. *Computer Applications in Engineering Education,* 16, 211-222. http://dx.doi.org/10.1002/cae.20149

Zuke, M., & Umbaugh, S.E. (1997). Cviptools: A software package for computer imaging education. *Computer Applications in Engineering Education,* 5, 213-220. http://dx.doi.org/10.1002/(SICI)1099-0542(1997)5:3<213::AID-CAE8>3.0.CO;2-G

Citation: Cazorla, M., & Viejo, D. (2015). Experiences Using an Open Source Software Library to Teach Computer Vision Subjects. *Journal of Technology and Science Education (JOTSE), 5(3), 214-227.* http://dx.doi.org/10.3926/jotse.143

On-line ISSN: 2013-6374 – Print ISSN: 2014-5349 – DL: B-2000-2012

AUTHOR BIOGRAPHY

Miguel Cazorla

Dr. Miguel Cazorla received a BS degree in Computer Science from the University of Alicante (Spain) in 1995 and a PhD in Computer Science from the same University in 2000. He is currently Associate Professor in the Dept Computer Science and Artificial Intelligence at the University of Alicante. His research interests are focused on computer vision and mobile robotics (mainly using vision to implement robotics tasks).He has published more than 100 papers in JCR journals and international conferences.

Diego Viejo

Dr. Diego Viejo obtained his BSc and MSc in Computer Science in 2002 and his PhD in 2008 both from the University of Alicante. Since 2004, he is a lecturer and a researcher in the Department of Computer Science and Artificial Intelligence (DCCIA) at the University of Alicante. His research interests are focused on 3D vision applied to mobile robotics.

Published by OmniaScience (www.omniascience.com)