MVISION, A TOOLBOX FOR COMPUTER VISION COURSES

J.R. Ruiz-Sarmiento, F.A. Moreno, J. Monroy, J. Gonzalez-Jimenez

Machine Perception and Intelligent Robotics Group, System Engineering and Auto. Dept., Instituto de Investigación Biomédica de Málaga (IBIMA), University of Málaga (SPAIN)
{jotaraul, famoreno, jgmonroy, javiergonzalez}@uma.es

Abstract

This paper presents the mVision toolbox to the lecturers' community and computer vision practitioners, which is publicly available at https://github.com/jotaraul/mVision. mVision provides a number of intuitive Graphical User Interfaces (GUIs) to apply different computer vision techniques like: image smoothing or enhancement, edge detection, segmentation into regions, etc. The interesting point of these GUIs is that they offer handy ways to select and configure different popular algorithms and, in addition to the results of their execution (*e.g.* detected edges or regions), they also provide intermediate information that is useful for understanding what it is underneath. With these features, the toolbox attempts to fill the gap between low level implementations of computer vision techniques, which are efficient resources but hard to leverage for teaching purposes, and high level interfaces that bring to you just the output of these techniques, neglecting interesting intermediate results. In this way, mVision aims to become an engaging tool for both lecturers, pursuing a better and quicker understanding of the basis of these techniques, and computer vision students and practitioners seeking to enhance that basis. The toolbox is written for MATLAB, a popular software application in academia. This work also yields directions about its utilization for educational purposes, and describes our experience with it in undergraduate and graduate courses since 2009.

Keywords: mVision, educational software, MATLAB, toolbox, computer vision courses

1 INTRODUCTION

Computer Vision is a trending field nowadays, as evidenced by its application in cutting-edge technological sectors like the autonomous navigation of vehicles or robots [1][2], the modelling and/or identification of objects and environments [3], or the registration of images [4], to name a few. In fact, computer vision techniques are present in our daily life each time that we take a panoramic picture, or we apply funny effects to our face through a video application running in our smartphones. Moreover, our faces are also the main characters when these techniques seek for suspicious behaviors or people in crowded places like shopping centers or airports [5]. Computer vision is also present when we see our favorite sports at TV in different ways, like with the Hawk-eye in tennis or cricket [6], or in combination with augmented reality to draw guide lines to check if a football player was offside. In medicine, there are numerous diagnosis tools that exploit similar techniques. The interest on this field can be also noticed in the number of courses addressing it in online platforms like Coursera [7], or in Bachelors and Masters studies.

A software tool for visually illustrating the principles behind typical computer vision techniques would be a resource of significant interest for lecturers teaching those courses. Usually, computer vision algorithms consist of a pipeline of techniques applied to an image in order to retrieve the desired information. Although there exist many tools implementing these techniques, most of them focus on providing a computational efficient solution, like OpenCV [8] or the Core Computer Vision library [9], hence being necessary a certain level of understanding of low-level details and therefore not optimal for teaching purposes. On the other hand, there are also tools focused on illustrating the results of those techniques, for example showing the edges detected in an image (e.g. [10]), but they lack in providing intermediate information that illustrates what it is underneath. The ideal tool would be a trade-off between these two common options.

In this paper we present the mVision toolbox, a set of Graphical User Interfaces (GUIs) implemented in MATLAB [11] to show the performance and relevant details behind most common computer vision techniques, including segmentation, edge detection, or object recognition, among others. In this way, each GUI offers the possibility of executing and configuring a number of algorithms pursuing the same goal, e.g. segmenting the regions in an image, also showing illustrative intermediate information that assists the lecturers in their explanations and helps the students to understand the details at a glance.

The development of the toolbox started in 2009, and it has been constantly updated according to the experience acquired in our offered courses. Recently we have made mVision public under the GNUv3 license for the lecturers' community at https://github.com/jotaraul/mVision, and we also welcome any contribution to it.

The second section of this paper presents the components of mVision by describing each of its GUIs. The third section gives indications about how to use it in education, and also reports some of our experience in that matter. Finally, Section 4 discusses the conclusions and possible future work.

2 THE MVISION TOOLBOX

The mVision toolbox consists of a number of Graphical User Interfaces (GUIs) implemented through the GUI Development Environment (GUIDE) tool of MATLAB. In its beginnings the toolbox was made available at SourceForge, where some previous versions are still available, but its release as a mature software (version 1.2) has been recently done in the trendy GitHub platform. Concretely, it is composed of 6 GUIs grouping the available techniques by the task that they accomplish, that is: *Smoothing*, *Enhancement*, *Edges detection*, *Segmentation*, *Description*, and *Recognition*, plus a general GUI that facilitates their launching (see *Figure 1*-left). An interesting feature of mVision is that the resulting image of any GUI can be copied to the clipboard, so it can be used by any other GUI.

Figure 1-right shows the toolbox structure in GitHub, including a directory for each computer vision task containing the developed techniques. A directory called GUIs contains the implementation of the GUIs visual aspect, while an auxiliary directory provides useful functionalities for different components. Finally, it is also included a directory with images to test the algorithms. The next sections describe these GUIs in more detail. The interested reader can check general computer vision books for more information about the mentioned techniques, like for example [12] or [13].

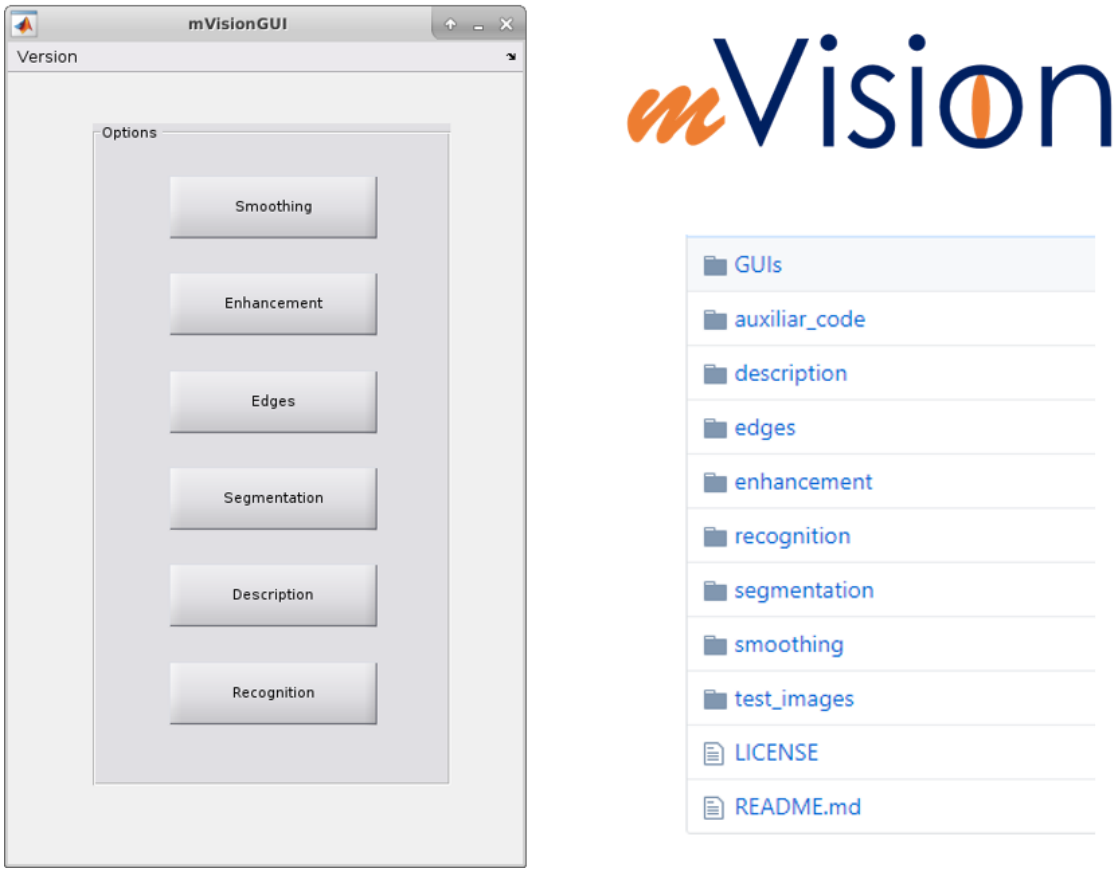


Figure 1. Left, mVision main GUI. It provides an easy way to access the different implemented techniques grouped by the task that they address. Right-top, mVision logo. Right-bottom, structure of the toolbox code available at GitHub.

2.1 IMAGE SMOOTHING

Smoothing is a typical Image Processing technique that pursues the removal of noise in images while keeping their important features. This is a typical pre-processing step that is applied previously to other computer vision techniques in order to improve the quality of their results. The noise appearing in an image can provide from different sources, like the camera sensor response, the digitalization of the image, or its transmission. Noise can be also of different nature:

- *Gaussian noise*. This noise is produced during the image acquisition due to poor illumination conditions, inadequate sensor temperature, or electronic noise, and manifests as an additive zero-mean Gaussian error affecting the pixel intensities.
- *Salt and pepper noise*. An image affected by this noise shows black pixels in bright regions and white pixels in dark regions, and it is mainly due to digitalization and/or transmission errors.
- *Impulse noise*. Similar to the previous one, but in this case only black or white pixels appear at random locations in the image.

The toolbox provides three smoothing techniques for removing this noise, namely:

- *Averaged environment*. This is a simple smoothing approach where the value of a pixel is replaced by the mean value of the pixels surrounding it. This operation is efficiently completed by a linear convolution operation with a kernel [12], where the kernel size sets the number of pixels considered for averaging. Large kernel sizes can produce blurred images.
- *Gaussian filter*. This filter has the same basis as the previous one but, in this case, the neighbor pixels contribute to the averaging according to their distance to the pixel being

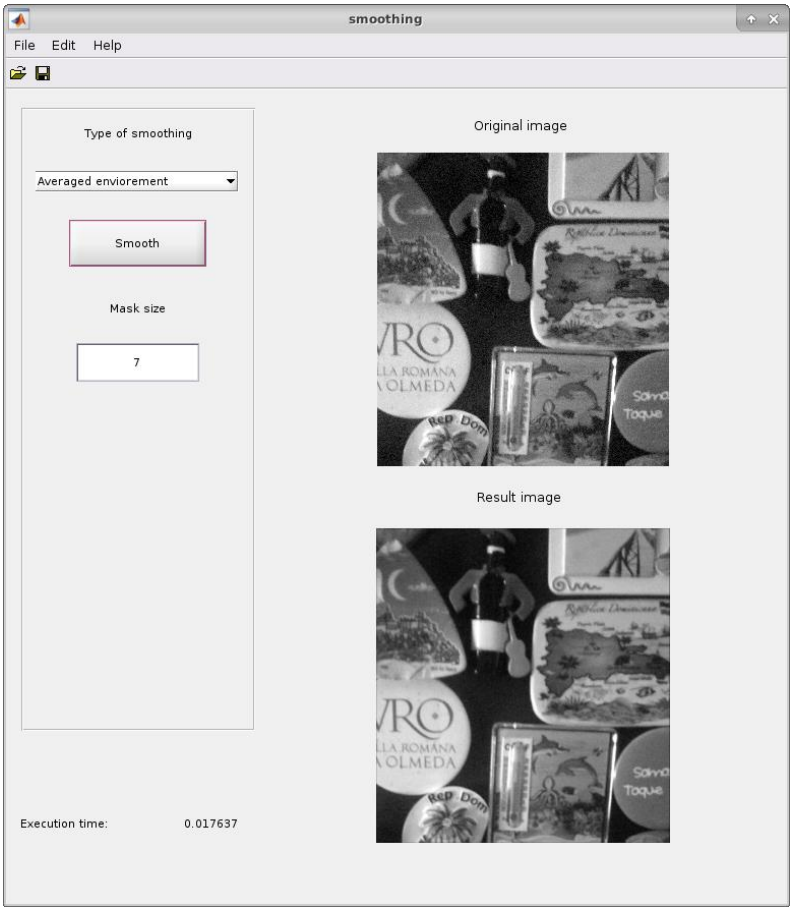


Figure 2. Screenshot of the Smoothing GUI with the execution of a Gaussian Filter. The mask (kernel) size and sigma (standard deviation) employed by the filter can be set by the user. The GUI provides the resulting image, the filter execution time, and the kernel used.

processed (where the center of the kernel is placed). This smoothing technique is especially efficient against Gaussian noise.

- *Median filter*. The last filter replaces the value of each pixel by the median of its neighborhood. This is not a linear operation, so it is computationally expensive, but it is effective removing salt and pepper and impulse noise, and does not blur the edges/contours in the image.

To configure the execution of these techniques, the GUI yields some options to the user, like the size of the neighborhood (kernel, see Figure 2), or the standard deviation (weighting) of the Gaussian filter.

2.2 ENHANCEMENT

The goal of the enhancement techniques is to increase the image quality by improving its brightness and contrast. Like smoothing, the enhancement of an image is usually a preliminary step in other computer vision tasks. At this point, we focus on the image histogram, a useful image representation that encodes the frequency of the pixel intensities, hence yielding valuable statistical information about the distribution of such intensities. This enables us to check the suitability of both the image brightness and contrast.

For example, in the left side of Figure 3 it is shown an image of the Pentagon and its histogram, where it can be seen how the image does not use the whole intensity range (from 0 to 255), thus indicating that its contrast can be improved. On the other hand, its mean brightness is almost in the center of the histogram, which indicates that it is suitable. mVision provides two ways to enhance an image:

- *Palette transformation*. The idea behind this technique is to stretch and shift the histogram to gain better leverage of the intensity range. For that, a new *palette* is built where the intensity values in the initial image are assigned to different values according to a look-up table, which is pre-built in order to be computationally efficient. Figure 3 illustrates how this operates, with the initial image on the left side, the look-up table in the middle, and the resulting image on the right side, along with their corresponding histograms. This information is shown to help to understand how this technique works.
- *Histogram equalization*. This algorithm modifies the image histogram so that it has a uniform distribution. It is especially useful in images where both the foreground and background are bright or dark, which correspond to over or under-exposed images.

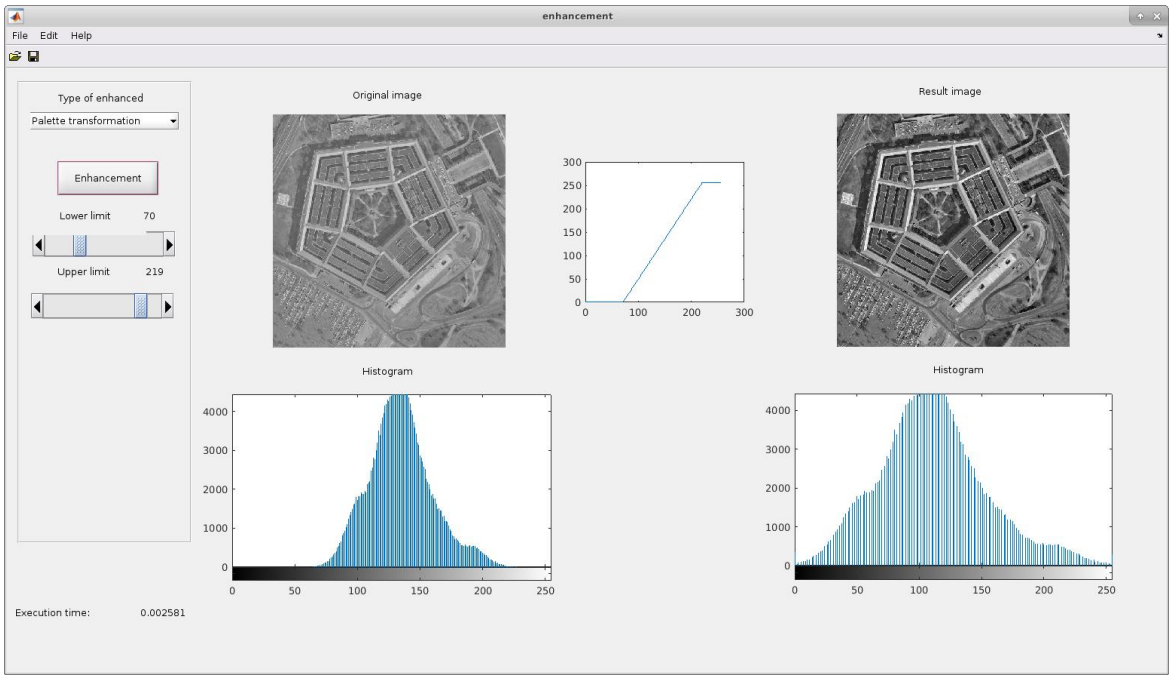


Figure 3. Enhancement GUI showing the results of the Palette transformation technique on the Pentagon image.

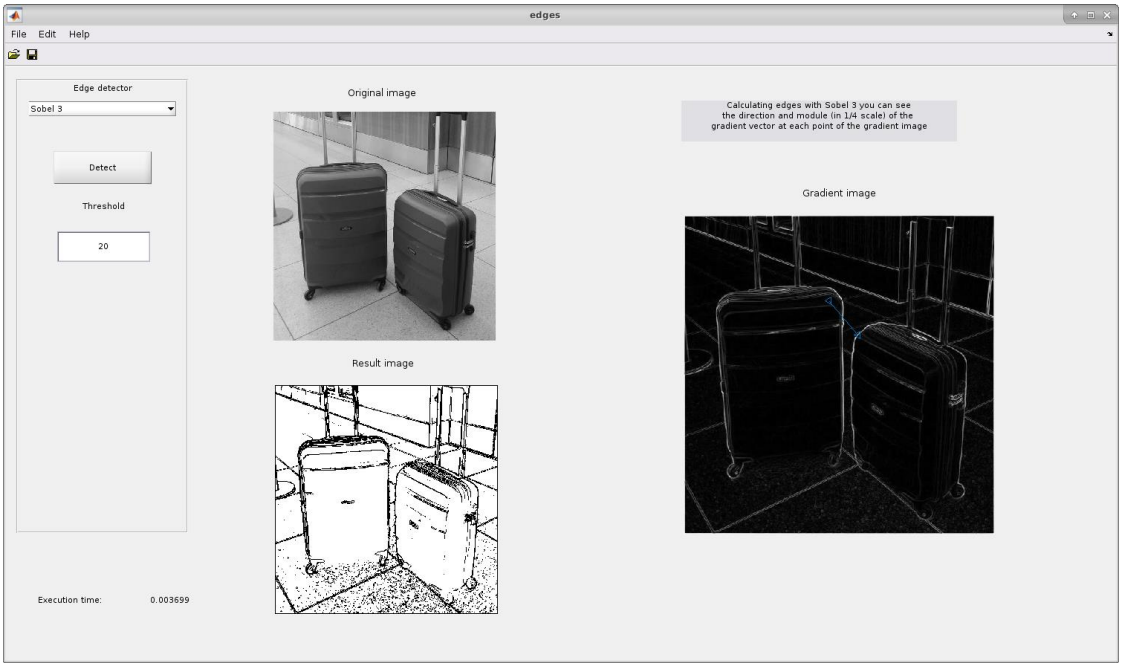


Figure 4. Screenshot of the Edge detection GUI with an execution of the Sobel operator.

2.3 EDGES DETECTION

The detection of edges in an image gives information about the contours of the elements within it, which are useful for tasks like segmentation, object detection, stereo matching, etc. An edge can be defined as an abrupt transition between image regions that have different intensities (*e.g.* a transition from white to black, or vice versa). For example, Figure 4 depicts on its left-bottom part the detected edges in the image just above it. Among the bunch of existing different edge detection techniques, our mVision toolbox provides implementations for the following three:

- *Sobel.* It is based on the idea of computing the first derivative of the image employing the Sobel operator, *i.e.* two 3x3 or 5x5 kernels that compute the vertical and horizontal derivatives of an image through a correlation operation. The value of the horizontal and vertical derivatives for each pixel are sum up and, if they are over a certain threshold, the pixel is considered part of an edge. This technique is highly sensible to noise, so it is necessary to smooth the image prior its use.
- *Derivative of Gaussian.* The Derivative of Gaussian (DroG) technique blends smoothing and derivatives to avoid such sensibility to noise. In this way only one operation (convolution) is necessary to robustly detect edges, hence being computationally more efficient.
- *Canny.* The well-known Canny edge detector was developed by John F. Canny in the 80's while he was designing a detector with low error rate, good localization and single response. It is based on the DroG operator, a non-maxima suppression step, and the application of a threshold with hysteresis. Although it is more complex than the previous two, nowadays it can be executed in real time and it is widely employed.

Apart from the results, the toolbox reports interesting information after the execution of these techniques. For example, the utilization of the Sobel operator produces a gradient image whose intensity values are the sum of the horizontal and vertical derivatives (in absolute value), and clicking over it at a certain pixel you can check the direction of maximum variation of these derivatives (blue line in the right image of Figure 4).

2.4 SEGMENTATION

The segmentation of an image groups its pixels into regions that share some type of property (intensity, color, texture, location in the image, etc.). Thereby, adjacent regions are significantly different according to that property. By doing so, the image can be represented in a simpler or more

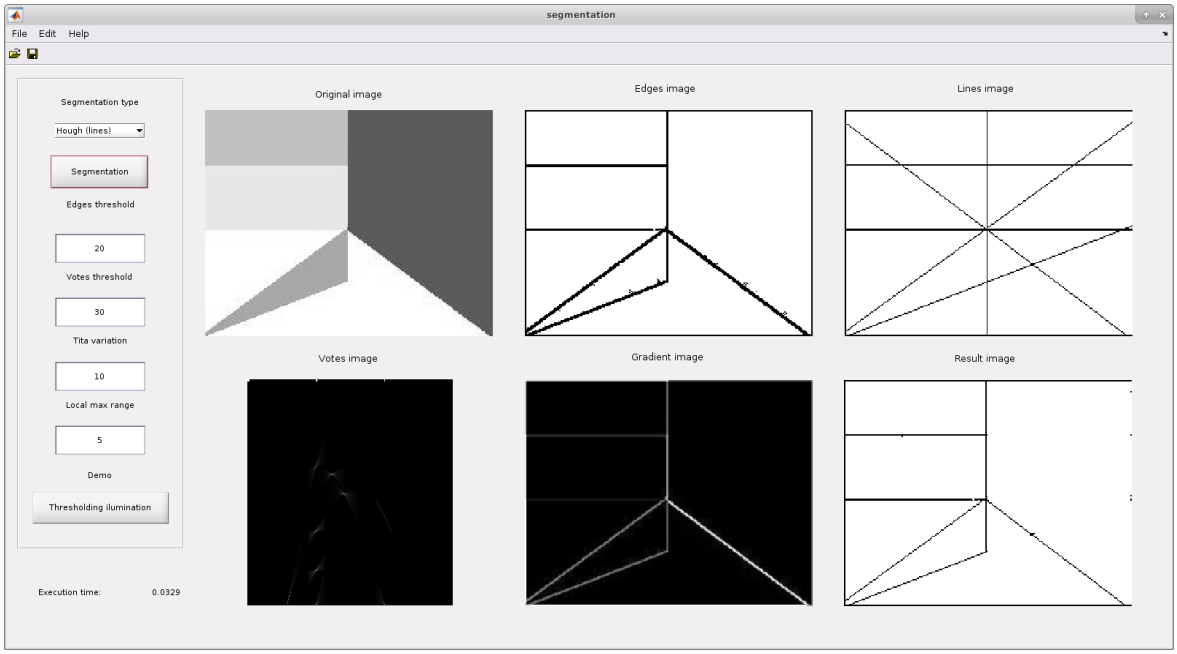


Figure 5. Lines detected by the Hough transform within the Segmentation GUI.

meaningful way that makes easier a further analysis, *e.g.* for detecting or recognizing objects. This segmentation can be accomplished by defining regions that cover the entire image, or by detecting the contours or edges of those regions (see previous section). mVision implements a method from each approach:

- *Hough transform*. This technique aims to find instances of objects with a certain shape following a voting scheme. This voting is carried out in a parameter space, which defines the shape we are looking for. Although it was originally devised for detecting lines (see Figure 5), it has been extended and it is able to detect shapes with other analytic forms, like circles or ellipses, as well as numerically described forms. The output of this algorithm are the detected shapes.
- *Regions growing*. This method initializes a seed (or a set of them) at a given location in the image, and expands it while there are neighbor pixels that share some certain criterion. Therefore, there will be, at the most, as many regions in the image as defined seeds. In general, there are some interesting design decisions that influence the performance of the method: the number of seeds to use, their location in the image, and the chosen similarity criterion. These criteria can be based on the textures in the image, the pixels color/intensity, or the expected shape of the regions, to name a few.

Figure 5 shows an example of the lines detected by the Hough transform in the image on the right-up part, as well as a bunch of intermediate results: the gradient image, the edges and lines detected in the image, the votes in the parameter space, and the resulting image, formed by the intersection between the gradient and lines images.

2.5 DESCRIPTION OF REGIONS

The segmented regions within an image can be described through a number of features, typically forming a vector of them that gives concise information about their shape, color, texture, position, etc. Once the regions have been characterized, object recognition techniques can take advantage of these descriptions to classify them into categories, which depend on the application. For example, in a computer vision software that analyzes a football match, they could be the football players, the ball, the referee, the field lines, the goals, etc. Our toolbox includes two methods for describing regions:

- *Thinning*. This technique is a morphological operator, working typically on binary images, that retrieves the *skeleton* of a region. *Thinning* is also useful to further process the output of edge detectors that produce contours with multiple responses (*i.e.* a width larger than one pixel) as

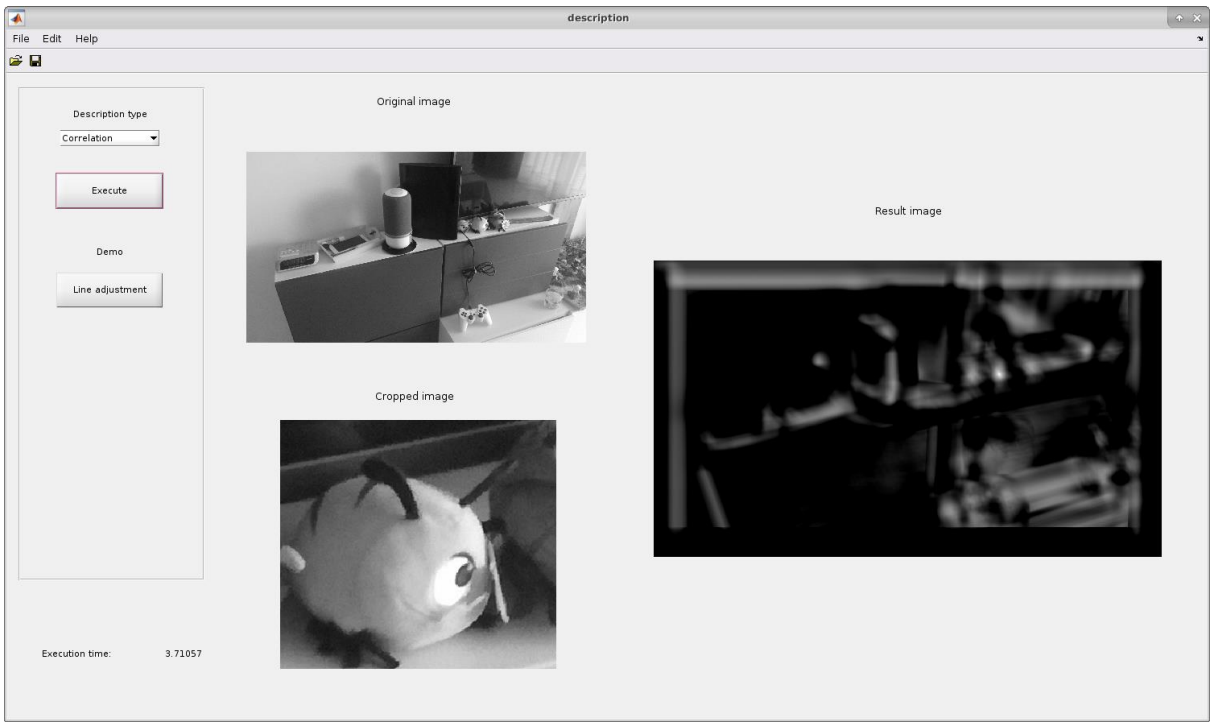


Figure 6. Example of execution of the Cross Correlation technique, where the stuffed bug on the left-bottom part of the image is detected in the image on the left-top part. The result of the algorithm is the key detected at the brightest point in the image on the right.

happens with the aforementioned Sobel or DroG techniques. Applying the *Thinning* technique, the edges can be reduced to lines with single pixel thickness. The provided implementation is inspired in the well-known algorithm by Zhang and Suen [14].

- *Cross correlation*. The aim of this algorithm is to employ the pixel intensities of a region as features of an object, using them to find the location of such object in another image (see Figure 6). Although it looks more like an object detection, it is included here as it is also used to describe and match image *keypoints* (*e.g.* extracted with the Harris keypoint detection).

This GUI also includes a demo showing the fit of a line to a set of points, a way to describe them using a mathematical, parametric expression. These points could correspond to selected pixels from an image, for example.

2.6 OBJECT RECOGNITION

One of the most popular computer vision tasks is to recognize the objects appearing in an image, *i.e.* to assign a label or category to the characterized segmented regions. There is a wide range of recognition techniques, including those based on partitions of the feature space[1], assigning a label to each partition. To classify an object, these methods determine the partition which the object belongs to according to its particular features. This approach is commonly referred to as *geometric*, and the algorithms implementing it can be roughly divided into parametric and non-parametric methods [12].

Parametric methods are based on the Bayesian decision theory, and try to optimize the obtained probabilities when assigning an object to a certain category (partition). Unlike these, non-parametric methods do not make any assumption about how the features are distributed. Examples of methods from the first group are those based on Normal or Binomial distributions, while from the second one are K-means, Support Vector Machines, Neuronal Networks, etc.

The recognition GUI includes an implementation of a Bayesian classifier, allowing the user to introduce instances of objects belonging to two different classes characterized by two features. Figure 7 shows an example of this, with two categories of objects (red and blue), and 6 instances of objects

[1] Space defined by the features used to describe the regions, where each dimension corresponds to a certain feature.

Figure 7. Recognition GUI illustrating the recognition of an object instance (magenta point) as an object of the red class.

belonging to each one. The user can then introduce a new object instance from an unknown category, and the GUI will categorize it as belonging to one of these groups.

For doing so, a Bayesian decision function is used, whose result, along with other intermediate information, is shown on the right part of the interface. In this example, the *Mahalanobis distance* is employed within the decision function, and the object is classified as belonging to the red category. If the *Euclidean distance* was used, its output would be the blue category.

3 MVISION IN EDUCATION

The toolbox is being used since 2009 in the *Computer Vision* course of the *Computer Science* degree, and in the *Perception Systems* course of the master in *Mechatronics Engineering*, both hosted by the University of Málaga. These courses are lectured by staff from the *Machine Perception and Intelligent Robotics group* (MAPIR) [15], who has an extensive experience in Computer Vision, specially applied to Robotics [16][17][18].

The GUIs and algorithms included in the toolbox are the result of analyzing both i) at what point of the lectures and for what techniques a visual explanation of the methods being studied was needed and ii) for which exercises the students could benefit from the GUIs demonstrations to better understand the basis of those methods. Just to illustrate the opinion of the students about the tool, during the last edition of the *Computer Vision* course (2017/2018), they were asked to fill a questionnaire that included the question: *Did mVision helped you to better understand the principles behind the used techniques?*, being the possible answer a number from 1 (absolutely not) to 5 (absolutely yes). The average score from 17 students was 4.7.

We have made available on the group webpage[2] further information about this work, as well as some course materials with examples of how do we use *mVision* in the practical exercises. As an example, Figure 8 shows one of these exercises, specifically the one dealing with object recognition by means of a Bayesian classifier.

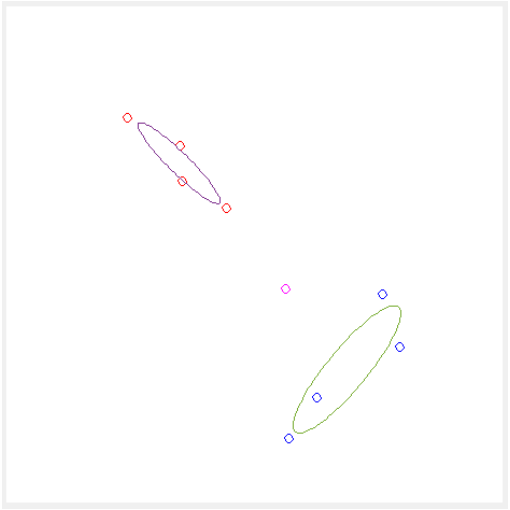
Briefly, the students first have to launch the *mVision* main GUI through the command `mVisionGUI`. Then, by clicking on the button that launches the object recognition GUI, and completing the points described in the exercise text, they can understand in depth the concepts behind that technique. This allows them to more confidently face the following parts of the exercise, which require them to implement further functionalities.

[2] http://mapir.isa.uma.es/mapirwebsite/index.php/papers/276

EXERCISE 6a: Object recognition with mVision

Concepts: Bayesian Classifier, Decision function, Mahalanobis distance, Euclidean distance

1. **Running the GUI:** Launch the **recognition** GUI of mVision, and follow the instructions on the bottom part of it to insert objects (points) in a similar way to how they are shown in the following figure (try to insert them in similar positions). The red points represent two features of objects of one class, the red ones features for other class, and the magenta point is the object to classify:



2. **Recognition:** Push the **Recognize** button and interpret the results obtained on the right part of the GUI (decision functions, covariance matrices, etc.).

3. **Same probability and covariance:** Now force the classes to have the same probability. With that, the GUI also forces the classes to have the same covariance matrices. This emulates situations where both classes have the same dispersion. Discuss the new results.

4. **Isotropic covariance:** Force them to have isotropic covariance. Discuss the obtained results.

5. **Distances:** Finally, comment which distances have been used in each section (2 to 4) to compute the decision functions: Mahalanobis or Euclidean?

4 CONCLUSIONS

In this paper we have introduced the *mVision* toolbox, a set of Graphical User Interfaces (GUIs) that have been designed to provide both the results and intermediate information of core computer vision techniques. *mVision* is implemented within the MATLAB ecosystem, which is widely used in academia, and it is publicly available at https://github.com/jotaraul/mVision. Its purpose is to be a valuable tool for supporting the lecturer during its explanations of the theory behind these techniques, as well as for helping the students to address the practical exercises. The six GUIs offered by the toolbox have been described, covering the following computer vision topics: image smoothing, enhancement, edges detection, segmentation, description of regions, and object recognition. Finally, our experience with the toolbox in bachelor and master courses has been commented, adding some examples that describe how to use *mVision* to address practical exercises.

In the future, we plan to keep adding new techniques to the presented GUIs, as well as to develop new ones, accordingly to the courses' needs. For example, it could be interesting to design a GUI to show the performance of different keypoint or feature descriptors and/or detectors, like Harris, SIFT, SURF, etc.

REFERENCES

[1] J. Ernesto Solanes, L. Armesto, J. Tornero and V. Girbés, "Improving image-based visual servoing with reference features filtering," *2013 IEEE International Conference on Robotics and Automation*, Karlsruhe, 2013, pp. 3083-3088.

[2] J. González-Jiménez, C. Galindo and J. R. Ruiz-Sarmiento, "Technical improvements of the Giraff telepresence robot based on users' evaluation," *2012 IEEE RO-MAN: The 21st IEEE International Symposium on Robot and Human Interactive Communication*, Paris, 2012, pp. 827-832.

[3] J.R. Ruiz-Sarmiento, "Probabilistic techniques in semantic mapping for mobile robotics," *Ph.D. thesis*, Dept. of Systems Engineering and Automation, University of Malaga, Malaga, 2016.

[4] R. Dasari and C. W. Chen. "A joint visual-inertial image registration for mobile HDR imaging." *Visual Communications and Image Processing (VCIP)*, 2016. IEEE, 2016.

[5] R. Arroyo, J. J. Yebes, L. M. Bergasa, I. G. Daza, and J. Almazán, "Expert video-surveillance system for real-time detection of suspicious behaviors in shopping malls," *Expert Systems with Applications*, 42(21), 7991-8005, 2015.

[6] N. Owens, C. Harris and C. Stennett, "Hawk-eye tennis system," *2003 International Conference on Visual Information Engineering VIE 2003*, 2003, pp. 182-185.

[7] Cousera webpage, https://www.coursera.org/ [Accessed online Nov'17]

[8] Itseez. Open Source Computer Vision Library, At https://github.com/itseez/opencv [Accessed online Nov'17].

[9] Core Computer Vision Library webpage, http://libccv.org/ [Accessed online Jan'18]

[10] Pinetools edges detection webpage, http://pinetools.com/image-edge-detection [Accessed online Nov'17].

[11] MATLAB webpage, https://www.mathworks.com/ [Accessed online Jan'18].

[12] D. Forsyth and J. Ponce, "Computer vision: a modern approach," *Upper Saddle River, NJ; London: Prentice Hall*, 2011.

[13] J. Gonzalez-Jimenez, "Visión por computador," *Thomson Paraninfo,* 1999.

[14] T. Y. Zhang and Y. S. Ching, "A fast parallel algorithm for thinning digital patterns," *Communications of the ACM* 27.3 (1984): 236-239.

[15] Machine Perception and Intelligent Robotics group (MAPIR) webpage, http://mapir.isa.uma.es [Accessed online Jan'18].

[16] F.A. Moreno, J.L. Blanco, J. Gonzalez-Jimenez, "Stereo vision specific models for particle filter-based SLAM", *Robotics and Autonomous Robots*, 57(9), 955-970, 2009.

[17] F.A. Moreno, J.L. Blanco, J. Gonzalez-Jimenez, "ERODE: An efficient and robust outlier detector and its application to stereovisual odometry", *2013 IEEE International Conference on Robotics and Automation (ICRA)*, Karlsruhe, 2013, pp 4691-4697.

[18] J.R. Ruiz-Sarmiento, C. Galindo, J. Gonzalez-Jimenez, "Building Multiversal Semantic Maps for Mobile Robot Operation," *Knowledge-Based Systems*, Volume 119, 2017, Pages 257-272.