Learning Computer Vision through the development of a Camera-trackable Game Controller

Andrea Albarelli, Filippo Bergamasco and Andrea Torsello

Dipartimento di Scienze Ambientali, Informatica e Statistica
Università Ca' Foscari di Venezia

Abstract. The trade-off between the available classroom time and the complexity of the proposed task is central to the design of any Computer Science laboratory lecture. Special care must be taken to build up an experimental setup that allows the students to get the most significant information from the experience without getting lost in the details. This is especially true when teaching Computer Vision concepts to prospective students that own little or no previous background in programming and a strongly diversified knowledge with respect to mathematics. In this paper we describe a setup for a laboratory lecture that has been administered through several years to prospective students of the Computer Science course at the University of Venice. The goal is to teach basic concepts such as color spaces or image transforms through a rewarding task, that is the development of a vision-based game controller similar in spirit to the recent human-machine interfaces adopted by the current generation of game consoles.

Keywords: Education; Computer Vision; Game Controller;

1 Introduction

The laboratory is a tool of paramount importance in science education for several reasons. Experimentation and hands-on activities allow students to connect theoretical facts with their effects. Moreover, challenges and intellectual rewards coming from such activities help in providing motivation for further studying. However, the design of good laboratory experiences is not an easy task [1]. The quandary between the conceptual complexity of the activities to be offered and the limited scope of an experimental session is even more noticeable when dealing with advanced subjects such as Computer Vision [2] [3] or even Image Processing [4] [5]. In fact, these topics tend to be taught in graduated courses [6], still, given the increasing importance of vision algorithms in the industry, they begin to be introduced also at undergraduate level [7] [8] [9] or even during the high school [10]. In this paper we describe a laboratory setup that has been designed to be useful in teaching some basic Computer Vision concepts to high school students that are keen to join the undergraduate Computer Science course at the University of Venice. These prospective students exhibited a wide range of different knowledge backgrounds, since some of them have attended humanities studies, whilst others were more at ease with mathematics and physics.

Setecec, p. 1, 2012.

Fig. 1. Shots of the vision-based controller and of the main panel of the educational application

For this reason it has been necessary to assume no previous programming skills and a very limited set of mathematical tools. To this end, we designed a laboratory session that includes about an hour of theoretical foundations and four hours of lightweight programming activity. The theoretical session introduces two concepts that will be central in the following experience: the YUV colorspace and the Hough Transform. They are both introduced with a minimal level of technicality and using very simple math notations. The programming session is performed within a custom environment that allows to write the body of the required functions directly without the need to know details such as how the images are acquired or how the methods are called. The functions have to be written with a subset of the Java syntax, that has proved to be simple to understand and suitable for this kind of laboratory lectures [11]. Before starting the programming activity itself, each student was supplied with a one-sheet handbook of the needed Java syntax and with some general instructions about the use of the integrated editor. The overall lab experience, which will be explained in detail in the following sections, imply the use of a simple game controller cut-out (see Fig. 1) made out of a PVC sheet that must be recognized from an image with the help of the color of the band printed on it. Once the recognition happens, the orientation of the band must be determined and used to control a little game.

2 Design of the Controller and of the Recognition Pipeline

To keep the experience as simple as possible we avoided the use of recognition techniques involving the search for a specific shape. In fact, we designed a controller that contains a very specific cue: a colored magenta stripe with two small green squares at its ends. The stripe and the squares will be segmented from the rest of the image by exploiting their chroma value. The resulting masks will be further analyzed respectively to find the orientation of the controller and to determine if the buttons are pressed. The orientation will be detected through the localization of a maximum in the Hough space, while the buttons status will be assessed by checking the presence or absence of green masked pixels on the right and on the left of the stripe. These of course might not be the best practice from a Computer Vision standpoint, but they are simple enough to be implemented during a short laboratory session and they exhibit a satisfactory level of complexity with respect to the targeted students.

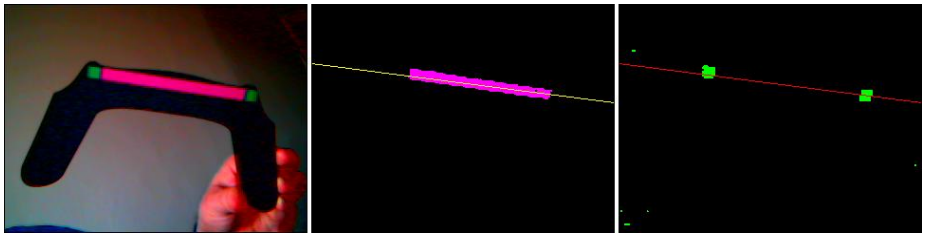

Fig. 2. Original image and magenta and green masks extracted using the YUV colorspace

2.1 Chroma-based segmentation in the YUV space

Since the detection of the controller should be invariant to changes in global illumination and gradients, the use of a colorspace that separates luminance and chroma should be preferred. We adopted the YUV color model [12], that separates the pixel information in one luminance channel (Y) and two chroma channel (U and V). The values in the chroma channels do not depend on illumination, rather, they localize the exact color shade over the U/V plane (see Fig. 3). In such plane the coordinates range from -0.5 to 0.5, the white light is locate at the origin, and at the four corners respectively the colors magenta, orange, green and blue can be found. The choice of magenta and green as distinctive colors for the controller is made specifically to allows for the maximum separability, since they are at the opposite positions in the U/V plane. To perform the segmentation of the stripe and the buttons, the students have been instructed about the principles of the YUV format and have been supplied with two matrices of floating point values (of Java type `double[][]`) named respectively `uData` and `vData`, the location on the plane of the sought color (as a pair of `double` values named respectively `u` and `v`) and a radius of tolerance (as a `double r`). Their task was to build the body of a function that fills a Boolean matrix accordingly with the fact that a given pixel was or was not within the given tolerance from the sought point on the U/V plane. As described in Sec. 3, students were not required to write the whole method, since the signature was already available and they just needed to write the `for` loop needed to implement the segmentation. In Fig. 2 we show the masking process performed on an image grabbed by a webcam for both the stripe and the buttons by setting the target values for the colors respectively at (0.5,0.5) and (-0.5,-0.5) and the radius of tolerance to 0.5.

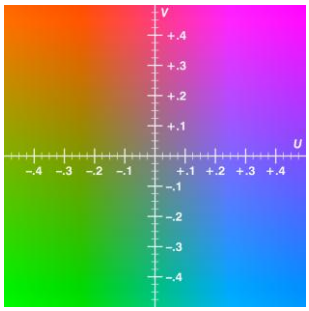

Fig. 3. The U/V plane

2.2 Controller Localization and Orientation Detection

Once the color segmentation has happened, the stripe can be localized and its orientation determined. This could be done, for instance, by applying a least square method to the points in the magenta mask. However this could lead to wrong results if a large

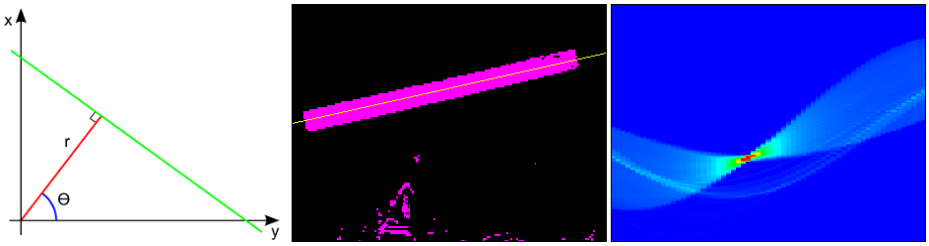

Fig. 4. Parameterization of the straight line used for the Hough Transform, mask for a stripe and respective representation in the Hough Space

number of structured outliers are present in the segmentation (which could easily happen if the student wears a purple outfit). Instead, we chose to take a different approach, which is also easier to understand with no previous knowledge other than basic Euclidean geometry of the straight line. This approach is the Hough Transform [13], which casts the problem of finding the straight line that justifies a set of observations in the Euclidean plane to the problem of finding the maximum of a voting function in the so called 'Hough Space'. The main idea of the approach is to parameterize the straight line with its distance from the origin (r) and the angle that the line going through the origin and perpendicular to it forms with the x-axis (θ) (see Fig. 4). Using this parameterization the equation of the line (for non-vertical lines) becomes:

$$y = \left(-\frac{\cos\theta}{\sin\theta} \right) x + \left(\frac{r}{\sin\theta} \right) \quad (1)$$

which, in turn, can be generalized in the equation:

$$r = x\cos\theta + y\sin\theta \quad (2)$$

that must be true for each point belonging to the line.

Any pixel that results from the segmentation of the stripe potentially belongs to the phenomenon that produced such segmentation. Since the coordinates (x_0,y_0) of the point are known, they must verify Eq. 2 for some value of r and θ. A single point is not enough to constraint two parameters, however they become related by the following equation:

$$r = x_0\cos\theta + y_0\sin\theta \quad (3)$$

Eq. 3 can be used to draw a curve in the θ/r plane, which is called the *Hough Space*. If all the pixels in the mask were aligned perfectly, all the curves in the Hough Space would pass through the same point, whose coordinates identify the correct solution for r and θ. Of course, the pixels coming from the segmentation cannot be perfectly aligned for several reasons: to begin with, the stripe is a rectangle rather than a monodimensional line, discretization errors are present and, finally, spurious pixel could be produced by background objects or any other source of noise. In practice, the

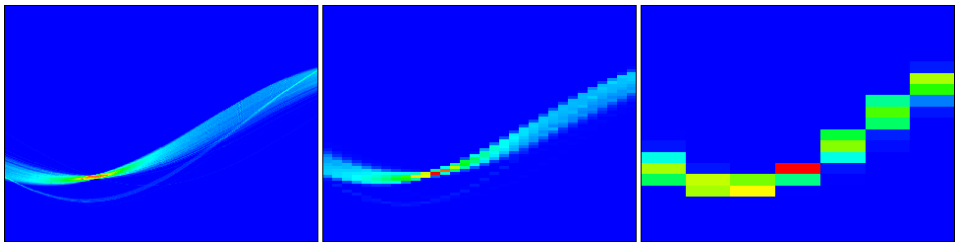

Fig. 5. Different levels of accuracy of the accumulator at various step granularity.

selection of the parameterization associated to the mask id performed by mean of a discretized set of voting bins with the following steps:

1. A discrete step is chosen for both θ and r. This step is used to build an accumulator matrix of floating point values;
2. For each one of the pixels found in the mask Eq. 3 is calculated for values of θ ranging from 0 to π with the chosen step. For each pair (θ, r) obtained a vote is inserted in the corresponding bin of the accumulator matrix;
3. After all the pixels have been processed the maximum value in the accumulator is found and the corresponding value for θ and r are retained.

In the last column of Fig. 4 the accumulator populated with the votes coming from the segmentation shown in the middle column is displayed. The values have been colored with a chromatic scale that ranges from blue (lower values) to red (higher values). It should be noted that, while the maximum is well defined, a spurious set of curves has been produced by the erroneous pixels in the bottom half of the mask. Additionally, the choice of the step values for the accumulator has effects on the accuracy in the determination of the correct parameterization. In Fig. 5 we show the difference in the accumulator for different granularities applied to the same scene. At a high accumulator resolution the maximum is well localized, whereas at a more coarse granularity the aliasing effect due to the binning hinders the result.
To perform this operation the students have been supplied with the Boolean matrix produced with the previous function, a pre-allocated accumulator matrix of floating point values named `accumulator` and the step amount for both for θ and r. Their task was to fill the accumulator with the appropriate `for` loop. The educational value of the experience lies both in understanding the process (which involves an unusual transform between spaces) and in playing with the step parameters to grasp the effect of binning on voting.

2.3 Check of the Button Status

This last step is optional and has been included in the laboratory experience in order to allow the students with higher skills or previous programming experience to perform an additional challenge (and avoid them becoming bored). The student is supplied with two Boolean arrays that contain the projection of the magenta and green segmentations along the line corresponding to the extracted orientation of the control-

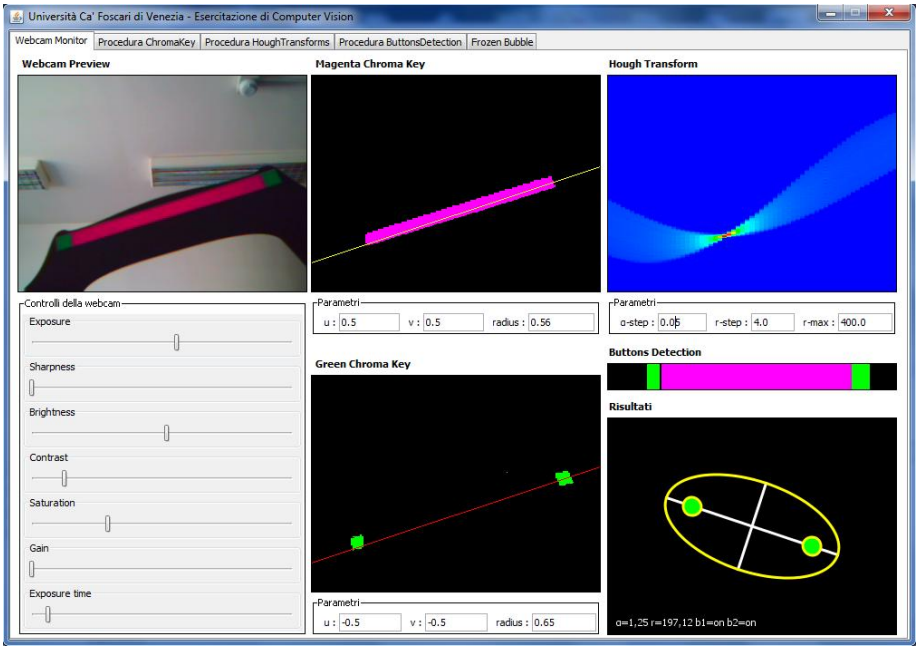

Fig. 6. The custom development environment implemented for the laboratory lecture (some text is in Italian language in order for it to be understandable by the target audience).

ler and the function written should fill a two place Boolean array with the status of the two buttons. Again, a method signature is already loaded in the system and the user only needs to fill in the body of such method, however no suggestion is given about how to proceed with the implementation. A reasonable solution is to find the barycenter of the magenta stripe and check for the presence of green pixels respectively on the left and on the right.

3 A Practical Sandbox to Simplify the Complexity

In principle, the proposed pipeline could be implemented from scratch, but this would require a lot of effort and skills, which is inappropriate considering both the target audience and the available time. Moreover, tasks such as image acquisition are notoriously cumbersome, as they require to fiddle with system specific APIs. To this end, we deployed a custom environment that takes care of most of the details and lets the students concentrate on the implementation of the algorithm and on the experimentation with the effect of the different parameters. In Fig. 6 we show the main panel of the development environment. In the first column the webcam preview is available. Here a streaming video of the frames captured is displayed together with the controls offered by the camera driver. This allows the student to modify these parameters and learn how they affect the appearance of the acquired frame and the whole pipeline. Specifically, by changing the exposure control, students can learn about the trade-off

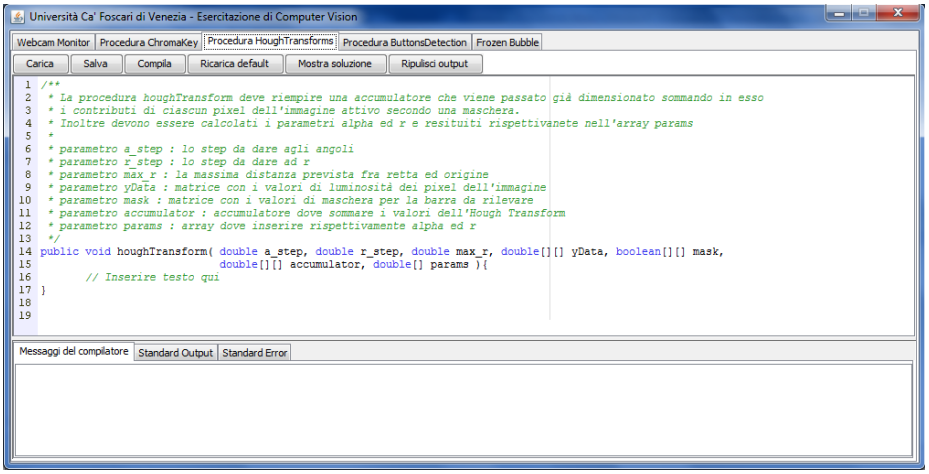

Fig. 7. The editor panel used to fill in the methods body.

between brightness and blurring for a dynamic scene, brightness and contrast controls give some insight about dynamic ranges, finally the saturation control is directly related to the intensity of the color channels and thus to the sensitivity of the detection step. In the remaining part of the panel the input parameters and the output of the different methods implemented are shown. Finally, the tool exhibits a set of three tabs that contain the editors that will be used to write the actual code and a last tab that allows to play a small game (Frozen Bubble) using the controller.

3.1 Acquiring Images from Camera and Running Custom Code on them

The acquisition of the stream coming from the webcam is done automatically by the custom environment by mean of the DSJ library. Since this library is a wrapper between Java and DirectShow, the implemented system can be used only within the Windows operating system. While this is quite a limitation, by adopting this solution all the webcams supported by Windows drivers can be used seamlessly. Each frame acquired is converted in the YUV format and supplied to the user-defined method as three matrices of floating point values. In addition to those matrices also other parameters need to be passed: those parameters are specified by the user in the text boxes shown in Fig. 6, automatically converted to floating point and then properly supplied to the custom code written by the student. The output produced by each method is also displayed on the panel in an easily readable form. In detail, the masks produced by the chroma key filter are shown as binary images in the central column, respectively for the magenta and the green segmentation. The accumulator in the Hough Space is rendered as a color coded image at the top of the last column. Here are also shown the binary masks passed to the buttons detection method and a box that reports in a schematic representation the orientation of the controller and the status of the buttons as obtained by the Hough Transform processing function.

3.2 Writing, Compiling and Loading Code

The programming activity itself happens by writing the body of three Java methods into three specialized editor panes accessible through tabs on the main window. Each editor comes with the method signature already written, as well as with some lines of comments that explain the role of each parameter (see Fig. 7). The methods to be completed are meant respectively to perform the color segmentation of the image, the estimation of the controller orientation and of the status of the buttons. Each time that the student needs to test the code written he can compile it by clicking on a button on the interface, that will trigger the compilation (using Java introspection). If the compilation fails the errors obtained are displayed to the user, otherwise the newly created method is loaded by the class loader, inserted in the running pipeline and called for each acquired frame with the parameters specified in the main panel. At this stage any runtime error is also presented to the user as well as any text that is printed on the standard output. Since the system must protect the pipeline from infinite loops that might happen in the user code, before compilation the methods body are modified at the source level and a safety watchdog variable is inserted as an additional exit condition into each `for` or `while` loop.

4 Reception by the Students

The laboratory lecture described has been given to 206 prospective students over the last academic year. The effectiveness of the lecture and its appreciation by the students have been evaluated by submitting to them a multiple choice test and a survey right after the programming session. The survey asked to express the degree of satisfaction with respect to five different aspects of the experience, each level of appreciation goes from a minimum to 1 to a maximum of 5. The results obtained were the following:

Question	Avg. score
How much were you interested in the laboratory session before participating to it ?	4.22
The lecture was clear enough and the covered topics were easy to understand ?	4.39
Are you satisfied with the organization of the laboratory and the tools supplied to perform the required tasks ?	4.33
Do you think that the experience allowed you to acquire some new skills ?	4.42
Rate the overall experience	4.65

Overall we can see that all the scores were rather good and that the outcome of the experience slightly exceeded the expectations.

The test contained 10 yes/no questions about the topics introduced in the theoretical session and further developed with the programming activity. The results of the test have been the following:

Question	% correct
Several different color spaces exist and no one perfectly fits all the application scenarios	82.5
In the YUV color space white and blue are more distinguishable colors than green and magenta	85.0
The Y channel in the YUV color space is associated to the amount of yellow in a pixel	80.0
Images acquired by a digital camera are organized in a regular rectangular grid	82.5
To find lines in an image a parameterization based on intercept and slope has been used	50.0
A point in the Euclidean space can be related to a straight line in the Hough space	45.0
The angular resolution of the accumulator has effect on the estimation of the angular position of the controller	75.0
The distance from the camera (as long as the controller is visible) has an heavy influence on the accuracy of the detection	67.5
The presence of magenta or green objects in the scene prevents the operation of the system	70.0
It is useful to accumulate in the Hough space values that are proportional to the luminance channel	72.5

While about a half of the students got wrong the more theoretical questions (which unfortunately coincides with maximum entropy), the results indicate a good understanding of most new concepts introduced during the lecture. This is also true with respect to the awareness of the influence of the parameters over the detection process.

5 Conclusions and Future Work

In this paper we presented a setup for a laboratory lecture aimed at teaching some basic Computer Vision concepts. The main goal was to mold an overall design of the experience that allows to fully enjoy the activity with no previous programming knowledge and after receiving a minimal set of theoretical insights from the lecturer. This has been done by implementing a custom development environment that takes care of acquiring images from a webcam and invoking the methods implemented by the students. The topic of choice for the lecture was the detection of a specially crafted game controller through chroma-based segmentation and Hough Transform. The

experience has been proposed to about two hundred students and has shown to be generally well received and stimulating. The results obtained by the students with a multiple choice test issued after the programming session have demonstrated a good understanding of the new concepts introduced. In the future, additional laboratory experience could be designed taking advantage of recently available 3D sensors.

References

1. Hofstein, A., Lunetta, V.: The laboratory in science education: Foundations for the twenty-first century. Science Education 88(1), 28-54 (2004)
2. Bebis, G., Egbert, D., Shah, M.: Review of computer vision education. IEEE Transactions on Education 46(1), 2-21 (2003)
3. Maxwell, B.: A survey of Computer Vision education and text resources. International Journal of Pattern Recognition and Artificial Intelligence (IJPRAI) 15(5), 757-773 (2001)
4. Roman, D., Fisher, M., Cubillo, J.: Digital image processing-an object-oriented approach. IEEE Transactions on Education 41(4), 331-333 (1998)
5. Greenberg, R.: Image Processing for Teaching: Transforming a Scientific Research Tool into an Educational Technology. Journal of Computers in Mathematics and Science Teaching 17(2), 149-160 (1998)
6. Pridmore, T. P., Hales, W. M. M.: Understanding images: an approach to the university teaching of computer vision. Engineering Science and Education Journal 4(4), 161-166 (1995)
7. Sarkar, S., Goldgof, D.: Integrating Image Computation in Undergraduate Level Data Structures Education. International Journal of Pattern Recognition and Artificial Intelligence 12, 1071-1080 (1998)
8. Egbert, D., Bebis, G., McIntosh, M., LaTouttette, N., Mitra, A.: Computer vision research as a teaching tool in CS1. In : Frontiers in Education, pp.17-22 (2003)
9. Hoover, A.: Computer vision in undergraduate education: modern embedded computing. IEEE Transactions on Education 46(2), 235-240 (2003)
10. Greenberg, R., Raphael, J., Keller, J., Tobias, S.: Teaching high school science using image processing: A case study of implementation of computer technology. Journal of Research in Science Teaching 35(3), 297-327 (1998)
11. Moscariello, S., Kasturi, R., Camps, O.: Image processing and computer vision instruction using Java. In : IEEE Workshop on Undergraduate Education and Image Computation (1997)
12. Wharton, W., Howorth, D.: Principles of Television Reception. Pitman Publishing (1971)
13. Duda, R., Hart, P.: Use of the Hough transformation to detect lines and curves in pictures. Commun. ACM 15(1), 11--15 (1972)