

Introducing Algorithm Design Techniques in Undergraduate Digital Image Processing Courses

Ángel Sánchez ESCET Univ. Rey Juan Carlos c/ Tulipán, s/n 28933 Móstoles (Madrid) SPAIN an.sanchez@escet.urjc.es Phone: +34-91-6647452 Fax: +34-91-6647490	José F. Vélez ESCET Univ. Rey Juan Carlos c/ Tulipán, s/n 28933 Móstoles (Madrid) SPAIN j.velez@escet.urjc.es Phone: +34-91-6647452 Fax: +34-91-3367490	Ana Belén Moreno ESCET Univ. Rey Juan Carlos c/ Tulipán, s/n 28933 Móstoles (Madrid) SPAIN a.b.moreno@escet.urjc.es Phone: +34-91-6647458 Fax: +34-91-6647490	José L. Esteban IPSA Pº. de la Castellana 165 28005 Madrid SPAIN jotaele@ipsa.es Phone: +34-91-5672900 Fax: +34-91-567 29 29
--	---	---	---

Abstract

This paper documents the development and first offering of a undergraduate course in Digital Image Processing at the Rey Juan Carlos University of Madrid (Spain). The article describes how the appropriate introduction of main Algorithm Design Techniques can successfully assist the students to achieve a comprehensive understanding of image operations and related algorithms. Image processing problems offer a natural way to present real world problems where the students can use their algorithmic knowledge. Furthermore, image processing solutions need from a methodological development and require efficient well-designed algorithms. This paper presents an effort in the integration of Algorithm Design Techniques in a Digital Image Processing course with a very practical scope.

Keywords

algorithm design techniques, digital image processing, pattern recognition, undergraduate education, assignments, course projects

1 Introduction

Digital image processing deals with the systematic manipulation of an input image to produce an output image that is better suited for viewing or subsequent analysis. Image processing is an application-oriented field and its applications relate with many other disciplines like biology, document processing, astronomy, meteorology, medicine, robotics, etc. With the recent hardware advances and reduction of computing costs, many industrial projects demand the use of real-time image processing tasks. Like in many other areas of engineering, digital image processing problems may be solved in a methodical, structured manner. The analysis and efficient design of involved algorithms in image applications is a critical component in the image processing methodology. Therefore, with the emerging of software programs that manipulate images, a computer science student needs to know about efficient image processing algorithms. We consider that this emphasis in algorithm design techniques helps with the comprehensive study of image algorithms in two ways. First, it leads to an organised method to devise (in many cases, efficient) image algorithms since there are very few algorithm design techniques. Second, the study of these techniques allows the students to categorize the explained algorithms for a better understanding of them. In this way, it is necessary to relate algorithm

design techniques with image processing in undergraduate education. A similar work for the integration of image computations and data structures has been proposed in the literature [10].

We have used our approach in an undergraduate digital image processing course taught to computer science students at the Rey Juan Carlos University of Madrid, Spain. The image processing course was elective and lasted a semester (16 weeks) with four assigned subject hours per week. Our students have previously received an obligatory one-year (32 weeks) course on data structures and algorithm design techniques.

The course as implemented was structured to have three main components which were: (1) an individual image computing assignment, (2) a group-oriented image project, and (3) a written examination. For the first two of these components the knowledge of a structured high-level programming language (like Pascal or C) was required. In some projects, the student could use some image software libraries (i.e. the MATLAB image processing toolbox [9]) for rapid prototyping and easy implementation of their programs.

The paper is organized as follows. Section 2 sketches the required algorithm design background to follow the course. Section 3 describes the theoretical aspects of the proposed digital image processing course. In Section 4, we outline several examples of proposed course assignments and image projects. These experimental components have the particular focus on the use, when it is possible, of algorithm design techniques. Section 5 shows how the course has succeeded and summarizes the conclusions.

2 Students' algorithm design knowledge

When introducing topics in a computer science course, or in any other discipline, we benefit as instructors when taking into account of students' knowledge and abilities acquired in previous courses. In our approach, the main algorithm design techniques (or algorithmic schemes) are related with the representative algorithms explained in a digital image processing course. This integration is advantageous for the learning of both courses. From the digital image processing side, the students learn to devise image algorithms in a systematic way and/or to relate these particular algorithms with the characteristics (features) of general design techniques. From the algorithms side, some considered images processing operations are good example exercises to reinforce the knowledge of algorithmic techniques explained in a previous course on data structures and algorithms.

The computer science programs for undergraduate education are in general, with some local variations in Spanish Universities, strongly inspired by the various ACM and IEEE curricula. Rey Juan Carlos University of Madrid, which was founded in 1996, actually offers two three-year degrees in Computer Science (Systems and Management, respectively). Both degrees include an obligatory one-year (32 weeks) course on data structures and algorithms. The course introduce the students the basic data structures such as lists, stacks, queues, arrays, trees,

graphs, and hash tables. These structures are explained with the approach to abstract data types, and giving special attention to good programming practices of software reusability, modularity and encapsulation, are emphasised. An important part of the course (14 weeks) is devoted to the analysis of algorithm efficiency, and to the major algorithm design techniques: divide-and-conquer, backtracking, greedy algorithms, dynamic programming and domain transformations. When teaching each particular algorithm design technique a similar presentation order is followed for each technique: a motivation, its applicability conditions, a general programming scheme for the technique, and a collection of application examples coming from different domains (i.e. arithmetic, string, searching, sorting or graphs).

Next, we recall the different considered algorithm design techniques [2]. *Divide-and-conquer* is a recursive strategy suggesting that a problem should be splitted into subproblems, then combine the resulting solutions to the subproblems into a solution of the original problem. *Greedy method* builds solutions in steps, where each step increases the size of partial solution based on a local optimization: the choice selected in each step is that which produces the largest immediate gain while maintaining feasibility. *Backtracking* is a recursive technique that deals with combinatorial problems, and involves a systematic and exhaustive search for a solution or collection of solutions. *Dynamic programming* is used when the principle of optimality holds and the solution of a problem involves a sequence of optimal decisions. In the *transformational method*, the problem is transformed from one domain into another and solved, the resulting solution is then inversely transformed to obtain the solution in the original domain. These techniques appear as the major ones for designing algorithms [2][3]. Recently, Levitin has proposed an alternative taxonomy of algorithm design techniques according to their degree of generality [8].

3 Theory elements of the image processing course

This image course, oriented to Computer Science students, takes a semester and has four weekly hours for the subject (two theory hours and two practice hours). We propose the gradual introduction of algorithm design techniques, when possible, during the course. Although our approach is mainly practice-oriented, we also consider necessary to explain the digital image processing theory in the traditional form during lectures. We try to give the students a breadth scope of fundamental image processing classes (acquisition, enhancement, analysis, compression and synthesis), and practical applications. Several texts were used during the course, although we did not find any “ideal” textbook for the complete course. Perhaps the most used book was González and Woods’ one [4]. Other complementary textbooks were Jain et al.’s [6], and Baxes’ textbooks [1].

An outline of the covered topic in theory lectures follows.

1. Introduction
(human visual system, problems and applications of digital image processing)
2. Review of algorithm design techniques
(divide-and-conquer, greedy method, backtracking, ...)
3. Image acquisition
4. (image sampling and quantization)
5. Image enhancement and restoration
(pixel and region processing, geometric transformations, frequency domain processing)
6. Image analysis
(segmentation, feature extraction, object recognition)
7. Image compression
(lossless and lossy compression methods)
8. Advanced aspects
(introduction to computer vision, dynamic vision, visualization)

When the different image processing operations are taught to the students in theory hours, these operations and involved algorithms are related to the major design techniques. At the beginning of the course, an initial lesson recalls the students the fundamental aspects of these techniques (which have been deeply explained in the previous data structures and algorithms course). Table 1 shows some examples of image processing tasks which can be successfully focused using algorithm design techniques. Following each example, in parenthesis, the corresponding image processing class is mentioned.

Algorithm Design Technique	Digital Image Processing Example
Divide-and-conquer	Quad-trees (region segmentation) Median Filtering (image enhancement) Split-and-merge (region segmentation)
Greedy method	Huffman coding (lossless compression) Minimal spanning trees (object classification)
Backtracking	Component labelling (object classification) Region growing (image segmentation)
Dynamic programming	Optimal boundary (feature extraction) Line detection (feature extraction)
Transformation of the domain	Fast Fourier transform (image enhancement) Hough transform (image segmentation)

Table 1. Some image processing examples focused on the algorithm design techniques.

Using these or other examples as laboratory assignments, students can explore the relationship between design techniques and algorithms. We consider important to stress on the efficiency aspects of image algorithms. Perhaps, an initial algorithm designed using a particular technique is not directly applicable in practice. For example, this happens when using backtracking in order to implement a Component Labelling method. In this case, a more efficient algorithm can be identified and implemented. It is useful for the students to apply this “refinement” strategy for discovering a better algorithm with respect to time and memory requirements (independently of the successful application of any algorithm design technique).

4 Assignments and projects of the image processing course

In an application-oriented discipline like digital image processing, it results very necessary to reinforce the topics discussed in the classroom. The course requires the complement of laboratory assignments and practical projects. In our University, the image processing course was first experimentally taught in 1999. Due to the reduced number of students (approximately twenty followed the course) the computer resources were very limited and consisted on four PC (Pentium III) each one with a digitizer card, two video cameras, and the digitizer software.

The practical assignments were offered during the course after the theory of a class of image algorithms was discussed. Some laboratory sessions were used at the beginning of the course in order to make familiar the students with the image capture and with the programs used for image processing. Next, we give more details about the course assignments and the projects.

4.1 Course assignments

A laboratory assignment consists of individual-oriented image operations which were previously described in theory lectures. After the description in some depth of a class of image operations and their associate algorithms, the emphasis was shifted to help students to discover how to implement these operations. At this point, students were asked to decide if a proposed algorithm has “relation” with any of studied algorithm design techniques. In some cases, a simple image task is proposed to be solved and the students must devise an algorithm for such task according to a given design techniques.

It is very important to stress on the efficiency of image algorithms [7]. Perhaps the initial algorithm designed using a particular technique (for example the use of backtracking to have a first version of a component labelling procedure for image classification) is not applicable in practice. Then, a more efficient algorithm must be identified and implemented. We consider useful that students learn “by refinement” to identify a better algorithm in relation with time and memory requirements (independently from the application of algorithm design techniques).

These proposed assignments are individual and different for each student. Each assignment states a short description of its purpose, the data to be applied in, the results to be turned in, and in some cases the algorithm technique(s) to be used for having an efficient implementation. Mainly, many of the assignments were oriented to a particular design technique and the student might also explain why the chosen algorithm scheme is applicable. If none of the known algorithm design techniques were applicable; a reasoned justification of this is required. The computational complexity of implemented algorithms were analyzed for each one of the assignments. Next, we summarise some of the proposed. In parenthesis, we indicate a suggested algorithm design technique to be applied.

- **Assignment 1. Component labelling (backtracking)**

One of the most common operations in digital image processing is finding the connected components in an image. The points in a connected component form a candidate region to represent an object. A component labelling algorithm finds all connected components in a binary image and assigns a unique label to all points in the same component. The students are asked to design and implement a recursive algorithm for the proposed problem following the backtracking scheme and given a starting pixel. As the solution is inefficient, students are then asked to devise another equivalent non recursive algorithm using an auxiliary data structure to store the neighbours of a pixel.

- **Assignment 2. Median filtering (divide-and-conquer)**

The median filter is a ranking filter, where each pixel in an image is replaced with the median of the grey values in the local neighbourhood. The resulting image is free of pixel brightness that are at the extremes of the extremes in each group of pixels. If the neighbourhood for a given pixel is a 3x3 group, this region can be converted into a vector, which is first sorted and then the middle value (median) is picked. The students are asked to design an efficient sorting algorithm, which follows the divide-and-conquer technique (these sorting algorithms have been explained in the previous course of “Data Structures and Algorithms”), to implement the median filtering.

- **Assignment 3. Optimal boundary extraction (dynamic programming)**

A boundary of an object can also be viewed as a path through a graph formed by linking the edge element together. Linkage rules give the procedure for connecting the edge elements in order to form graph with cost on arcs [5]. In this assignment, an algorithm based on heuristic graph search to follow the contour of an object from starting node is explained. The students are asked to discover if Bellman’s Principle of Optimality, to find the optimal path between two given nodes, holds. They were also asked to write the recursive equations which express the

global optimization process. Finally, an iterative algorithmic implementation for the transformation was requested. Emphasis was placed on the need to transform the recursive equations into an iterative program which requires an additional table structure for increasing time efficiency.

- **Assignment 4. Huffman coding (greedy method)**

The goal of this assignment is to construct the Huffman compressed representation of a given image. The Huffman image compression is a lossless (entropy) coding technique. Pixel brightness values are replaced with variable-length codes based on their frequencies of occurrence in the image. An initial task for the students is to compute the image histogram yielding the frequencies of occurrence for each of the brightness in the image. The students were also asked for a greedy algorithm which allows to assign shorter codes to the more frequent brightness frequencies of occurrences. Finally, an analysis of the algorithm optimality was requested.

- **Assignment 5. Fast Fourier Transform (transformation of the domain)**

In this assignment the students look into frequency transform filtering operations. They are asked to transform an image from its original spatial-domain representation to a frequency-domain representation using a Discrete Fourier Transform. The frequency-domain image is then multiplied by a given frequency mask image, using a pixel-point process. Finally, the resulting image is inverse Fourier-transformed back to the spatial domain. This exercise is proposed as a direct example of the domain transformation technique. The students were required to analyze the Discrete Fourier Transform algorithm. Due to the existence of a more efficient implementation of this transform, the Fast Fourier Transform algorithm is also formulated and students were asked for its implementation and its complexity analysis.

- **Assignment 6. Quad-trees (divide-and-conquer)**

One of the efficient region representation methods for image segmentation is that of the quaternary trees or *quad-trees*. The method for representing a given image by its corresponding quad-tree is given by the following algorithm:

- If the whole image has an only brightness value, then it is represented as a tree with only one node (terminal) containing the colour of the image.
- If the image consists of pixels with different brightness values, then it is represented as a tree which contains a non-terminal node, which is subdivided in four components of the same size at which the same process is recursively applied. The

representation of each component corresponds to a descendant node of the non-terminal node.

The students were asked to obtain a quad-tree representation of a given image using a divide-and-conquer algorithm. Example of recursive partitions of a given binary example image are shown in Figure 1 (left). Thick lines correspond to the first partition, thinner lines the second partition, and so on. The resulting quad-tree after applying the algorithm to the example is shown in Figure 1 (right).

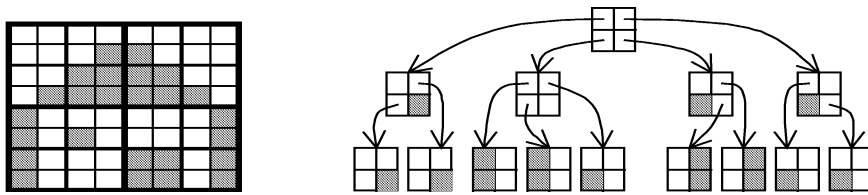


Fig. 1. Example of a binary image and its corresponding quad-tree representation.

4.2 Course projects

In the next sections we propose some simplified projects that must be solved by the students. The students are associated in groups of three, and one different project is assigned to each “work-group”. One main goal of these projects is to show the students the complexity of implementing real applications. These course projects are an excellent opportunity for the students to interact with their colleagues and to practice the acquired algorithm design and implementation knowledge and abilities. Also, as we have previously noted, when building some industrial application the computing time is a critical factor. To motivate the students with the importance of having efficient algorithm implementations, some kind of performance measurement related with their developed image processing programs is asked to them. In particular, each work-group must estimate the computational complexity of involved image algorithms in each particular project. Moreover, the performance of the implementations must be evaluated (using a timer function, such as the UNIX command `time` or the most sophisticated `gethrtime`).

- **Project 1. Mathematical morphology for image reconstruction of barcodes.**

Motivation: Reading a barcode from an image that has been previously scanned is a well-known problem in practice. Therefore, a sort of minor problems can be enumerated due to the unspecific technology used to capture the images.

Generally, scanners capture images in grey level tones, while barcode readers work over *binary* (monochrome, or black and white) images. The conversion of a grey-level image into a binary image is called *thresholding*. Advanced scanners use *regional thresholding* procedures for such operation. These methods enhance the visual quality of the images by eliminating homogeneous obscure zones, but they have pernicious effects over barcodes, precisely because barcodes are homogeneous obscure zones (as seen in Figure 2).



Fig. 2. In the left image, a degraded barcode due to the use of regional thresholding is shown. In the right image, the result of reconstruction of the same barcode using morphological dilatation is shown.

The Problem: In order to enhance the region of the image that contains the barcode, the students must propose and implement a reconstruction procedure. First, the procedure must find the region in the image that contains a barcode (i. e. using a split-and-merge algorithm). Then, a reconstruction procedure is applied (i. e. based on morphological dilation operation). The amount of computing time must be minor than one second per image in a standard PC.

- **Project 2. Automatic extraction of lighting aids in airport runways**

Motivation: A system that makes an automatic extraction of relevant objects in runway images has numerous applications. It can be useful in applications of practical importance, like the automatic maintenance of runway lights or to assist the pilot in planes not equipped with Instrumental Landing Systems (ILS). In order to simplify the problem, artificially lighted runway are the only considered images.

The Problem: The students are asked to build a system that makes an automatic extraction of lighting objects in runway images, grouping related objects in lines (as seen in Figure 3). Groups are oriented to organize the project in two main image processing stages. First, a segmentation process in which lighting aids are found in the image (p. e. using a thresholding procedure). In the second stage, these elements are joined in lines (i. e. using the Hough transform).

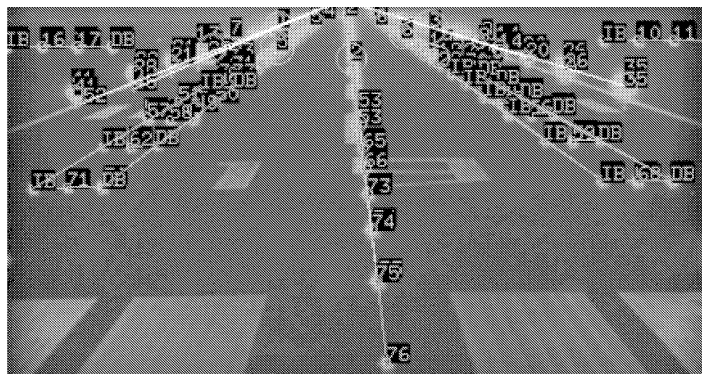


Fig. 3. Final segmented image with detected visual aids.

• **Project 3. Mathematical morphology for enhancement of typed documents.**

Motivation: Sometimes, when scanning a typed document, the thresholding procedure produces *white-noise* in the digital image, making difficult the reading process. An example is shown in Figure 4.

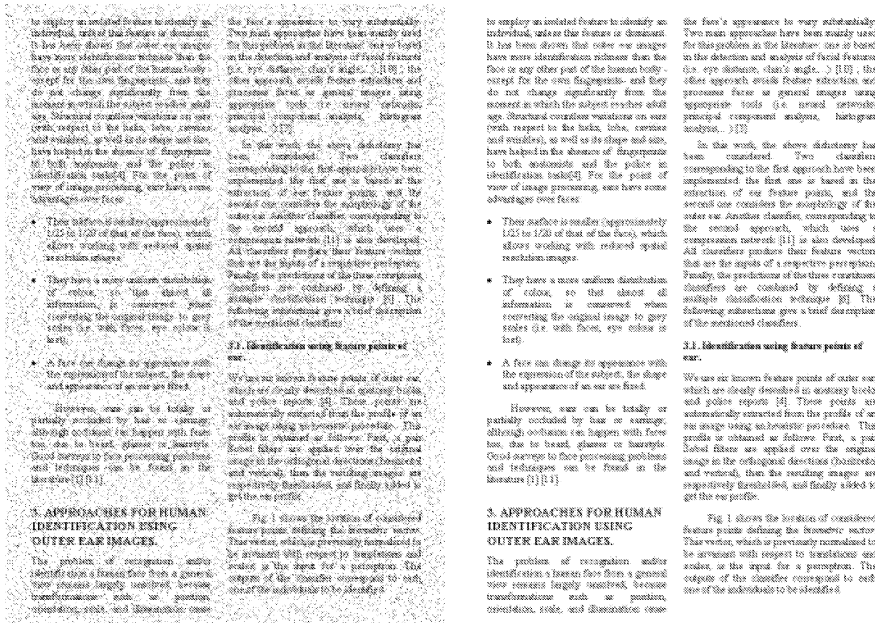


Fig. 4. The left image presents a document obtained from a scanner. The right image shows the same document after the application of a structural filter.

The Problem: In order to enhance the visual quality of typed images, the students must propose and implement a noise-reduction procedure. Two main approaches are recommended: 1) eliminate those objects with few pixels using a connected component algorithm; and 2) eliminate those objects which do not

contain some basic structured pattern (i. e. using morphological operations). The amount of computing time must be minor than one second per image in a standard PC.

- **Project 4. Automatic digit recognition.**

Motivation: The progress accomplished in the last few years in the field of pattern recognition allows the easy implementation of automatic digit recognisers. Character extraction and recognition techniques have potential applications in any domain where a large mass of document images bearing texts must be analysed (Figure 5).

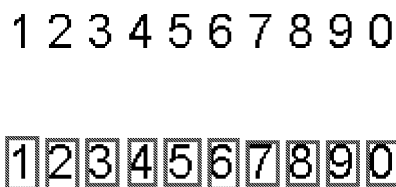


Fig.5. Example of text to recognize before and after the segmentation step.

The Problem: The automatic digit recognition problem is divided in two main image processing stages: digit segmentation and recognition. The digit segmentation process can be implemented by means of a component labelling procedure. The recognition of each kind of digit is necessary to determine the characteristics allowing to identify each digit. For such purpose, some kind of parametric classifier could be trained using a training set (which is supplied in order to build the classifier).

5 Conclusions

We presented an image processing course with a practical orientation based on the introduction of algorithm design techniques. From the image computation point of view, performance requirements for algorithms is quite demanding. Students have to be conscious about how important is the use of efficient algorithms in imaging systems. Algorithm design techniques contribute to devise algorithms and, in some cases, help us to find the optimal solutions for a computational problem. The usage of image operations which are suited to certain design techniques in assignments has resulted enriching for the students. They have reinforced their knowledge on algorithm design techniques with examples coming from the image processing

field. In addition, the availability of algorithm techniques helped the students to think about image algorithms in a way to understand them better (in the sense that all algorithms designed using the same technique share some general similar characteristics). The proposal of group-oriented image projects, which were thought as simplifications of real industrial applications, has greatly motivated the students. It has also allowed them to analyze and implement in a systematic and structured manner “practical” image solutions. The experience of some course instructors who have a previous image processing experience in industry applications has resulted very determinant in the selection and following of proposed projects. In some cases, parts of these course projects have also been designed and efficiently implemented using algorithm design techniques. Since image processing is one of the most exciting computer applications for a general audience, the relationship presented between algorithms and image processing allowed the students to be more interested in the study of algorithms and image processing than in a image processing survey course. Not only this approach has made the course to be more interesting but closer to industry applications. Therefore, the course has helped the instructors to teach the complexity of real world applications, in which computing time and system requirements are limited. The proposed approach has resulted highly rated by the students in course questionnaires. We hope to improve the course having repeated the experience in future semesters.

References

1. G.A. Baxes, *Digital Image Processing. Principles and Applications*, John Wiley & Sons, 1994.
2. G. Brassard and P. Bratley, *Algorithmics. Theory and Practice*, Prentice-Hall, 1988.
3. T. Cormen, C.E. Leiserson and R.L. Rivest, *Algorithms*, The MIT Press, 1990.
4. R.C. González and R.E. Woods, *Digital Image Processing*, Addison-Wesley, 1992.
5. A.K. Jain, *Fundamentals of Digital Image Processing*, Prentice-Hall, 1989.
6. R. Jain, R. Kasturi and B.G. Schunk, *Machine Vision*, McGraw Hill, 1995.
7. I. Kabir, *High Performance Computer Imaging*, Manning, 1997.
8. A. Levitin, “Do we teach the right algorithm design techniques?”, *ACM SIGCSE'99*, pp. 179-183, 1999.
9. The MathWorks Inc., *Matlab v. 5.1*, 1997.
10. S. Sarkar and D. Goldgof, “Integrating image computation in undergraduate level data-structure education”, *Intl. Journal on Pattern Recognition and Artificial Intelligence*, v. **12**, n. 8, pp. 1071-1080, 1998.