Transactions on Information and Communications Technologies vol 10, © 1995 WIT Press, www.witpress.com, ISSN 1743-3517

Digital image processing - an object-oriented approach

D.I. Roman,[a] M.H. Fisher,[b] J. Cubillo[b]
[a]*Communication Department, Technical University of Cluj-Napoca, 3400 Cluj-Napoca, Romania*
[b]*School of Engineering, Coventry University, Coventry, UK*

Abstract

This paper presents a class hierarchy that forms the core of a linkable library, which provides resources for essential digital image processing operations. The resulting module can be accessed from independent specialized applications, but it may also be used as the inner part of an integrated environment designed for students and researchers in the field of digital image processing. This software component was built aiming for both portability and extendibility. It contains attributes and methods that allow for a wide range of digital image processing algorithms; including point and geometric operations such as filter, edge detection, segmentation and texture operators; comprising FFT, DCT, wavelet and Gabor transformations. All these can be used for image enhancement, restoration, compression, analysis and synthesis. Besides its practical value, the class library is used to teach both object oriented software engineering methodology and digital image processing.

1 Introduction

The development of a complex software application, in a certain field, implies as an essential demand its separation into modules that interact on a client/server bases, e.g. Booch [1], Stroustrup [2], Halladay and Wiebel [3]. This article presents such a module containing essential resources for digital image processing. The module may be configured into a library which can then be linked into specialized and dedicated applications. Both static and dynamic versions of the library are available. Figure 1 shows the simplified diagram of how the library can be used.

Transactions on Information and Communications Technologies vol 10, © 1995 WIT Press, www.witpress.com, ISSN 1743-3517

The development of this library offers an excellent support for teaching object-oriented software engineering design. At the same time it is a most useful tool in teaching digital image processing.

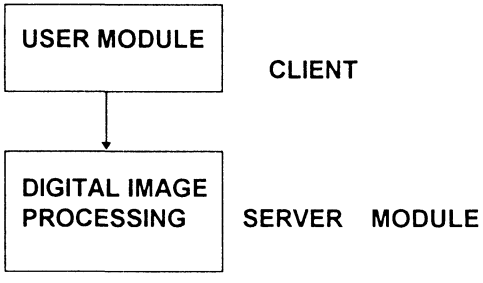


Figure 1: The digital image processing library.

The first application of the library to be considered is an integrated environment for digital image processing that has been developed for the use of students and researchers studying this subject.

The library contains resources for most of the image processing operations that are included in undergraduate and postgraduate digital image processing courses, Gonzales and Woods [4], Cubillo and Fisher [5]. These operations can then be used in specialized applications involving digital image enhancement, restoration, compression, analysis and synthesis.

The object-oriented approach has proved extremely useful in handling the complexity of the processing module. The design uses a specialized class hierarchy which in turn becomes a class library [2]. The orthogonal relationships of derivation and aggregation were taken in consideration when designing this hierarchy. All platform dependent aspects are left aside to be implemented in the user interface module and thus ensuring portability [1][3]. Also, the object-oriented implementation makes the library easily extendible, thus overcomming the problems of portability and extendibility encountered by the authors' earlier image processing package [5].

2 The *IMaGe* class

As images to be processed are available in a very large variety of file formats, e.g. Heller [6], Rimmer [7], the class corresponding to the *"image concept"* is kept abstract and provided with derived classes that specialize each constructor according to the image file format. In this way, the instantiation of an *"image object"* is unique and independent of the original file format. The *"image class"* with two descendants for two very popular image file formats is shown in Figure 2. New image file formats can easily be defined simply by declaring new descendants of IMG.

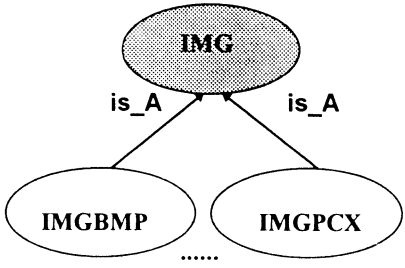


Figure 2: The IMaGe class with specialized constructors.

Instantiated objects of class IMGxxx are created dynamically in memory as required. The representation used is the same whatever initial file format is used as it is shown in Figure 3.

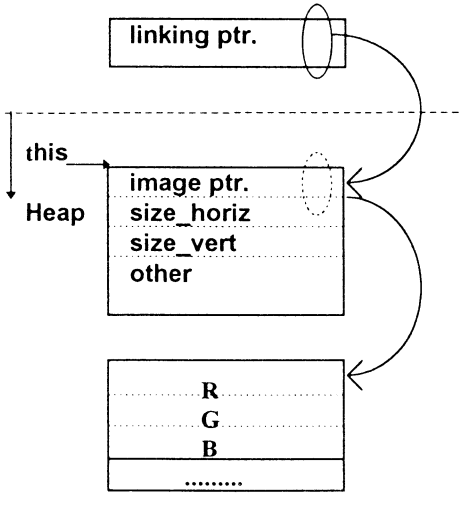


Figure 3: The memory representation of an instantiated image.

The only non-dynamic data to be defined in the client module is a linking pointer. All the aggregation attributes, pointed by *"this"*, are placed in the dynamic memory. One of these attributes is a pointer to the dynamic RGB array of pixels. In this way, images of whatever size can be easily instantiated. In summary, an image is represented internally in the 24 bits/pixel format.

3 The class hierarchy

The other main concept in the class hierarchy is that of a *"processed image"*. The corresponding class is the base class of the hierarchy. It is called ***P***rocessed ***IMaG***e (PIMG) and among other attributes it aggregates indirectly an IMG object. This object represents the target image, while a handle (index) to the source image is specified as an argument of the constructor.

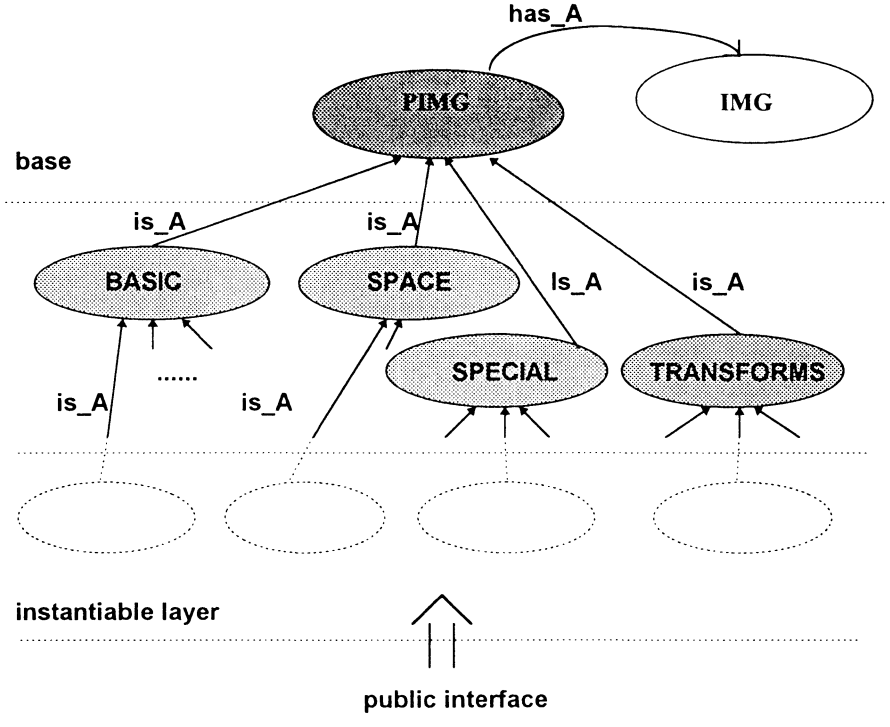


Figure 4: A simplified diagram of the proposed class hierarchy.

Obviously, this class is kept abstract and used only for derivative purposes.

Because there is a huge number of image processing operations, an intermediary layer is absolutely necessary. It gathers resources, attributes and methods, that are shared by groups of image processing operations.

The first two abstract layers of the class hierarchy can be seen in Figure 4. (The instantiable layer is too wide to be fully represented).

The classes for the intermediary level are also kept abstract because there is no need to actually instantiate them. The only other instantiable classes are the *"leaves"* of the hierarchy each of which correspond to a precise group of digital image

Transactions on Information and Communications Technologies vol 10, © 1995 WIT Press, www.witpress.com, ISSN 1743-3517

processing operations. Whenever an operation needs to be performed the right class is instantiated and the corresponding method is called. Of course, this method can itself call other methods inherited from the parent class.

All the digital image processing operations are naturally grouped in the following categories and each one has it's own corresponding abstract class:

- **BASIC** - ***point***, ***geometric*** and ***frame;***
- **SPACE** - ***space domain filters***, ***edge detectors and edge enhancement*** and ***morphological operators*** [4]***;***
- **SPECIAL** - ***segmentation*** and ***texture***;
- **TRANSFORMS**
  - ***FFT*** - Fast Fourier Transform, (e.g. Pitas, [8])
  - ***DCT*** - Discrete Cosine Transform [8]
  - ***WT*** - Wavelets Transform (e.g. Masters, [9]) and
  - ***GT*** - Gabor Transform (e.g. Vernon [10]);

Each sub-category corresponds to an instantiable class. Such a class is instantiated in order to perform a digital image processing operation. The operation itself can use all the members of the class or can use resources inherited from above.

4 The user interface

The sum of all the public interfaces of all instantiable classes forms the public interface of the whole class hierarchy. Through this interface the class library can be exploited. All other implementation details are perfectly encapsulated and hidden from the library user.

The user interface module is of course platform dependent (assuming that the machine is powerful enough). For the Windows 3.1 environment two alternatives have been implemented; one at API level and another one using the foundation of classes offered by Borland's Object Windows Library. Figure 5 illustrates a typical screen within the image processing environment built on the image processing library.



Figure 5: A typical screen within the digital image processing environment

5 Conclusions

The class hierarchy presented in this paper was built primarily to provide a large number of primitives for dedicated digital image processing applications. The software module is also used as a laboratory support for a Software Engineering course, since it contains all the necessary aspects that need to be illustrated. At the same time, the library can be used as the laboratory support for a Image Processing course.

The class hierarchy is compiled into a class library with a consistent and large public interface. The implementation details do not need to be known outside the library and so they are encapsulated. Two versions of the library are available one for static linking and the other is built as a DLL. Both were developed using Borland C++ 4.02 under Windows 3.1. All platform dependent aspects are not included in the library and so this general software resource is exportable and at the same time it is easily extendible.

Acknowledgment

This work has been carried out with the support of a TEMPUS Structured Joint European Project, in which both the University of Coventry, UK and the Technical University of Cluj-Napoca, Romania are partners.

Transactions on Information and Communications Technologies vol 10, © 1995 WIT Press, www.witpress.com, ISSN 1743-3517

References

1. G. Booch, *"Object-Oriented Analysis and Design"*, Second Edition, Benjamin/ Cummings Publishing Com., Inc., 1994, chap. 9-10.

2. B. Stroustrup, *"The C++ Programming Language"*, AT&T Laboratories, 1991, p 361-474.

3. S. Halladay and M. Wiebel, *"Object-Oriented Software Engineering"*, R&D Publications, Inc., 1993.

4. R. Gonzales, R. Woods, *"Digital Image Processing"*, Adison Wesley, Inc., 1992.

5. J.P. Cubillo and M.H. Fisher, *"Digital Image Processing and Machine Vision - an educational support package"*, IEE Colloquim, Oct. 21 1993, Savoy Place, London.

6. M. Heller, *"Advanced Windows Programming"*, John Wiley & Sons, Inc. 1992, chap. 3.

7. S. W. Rimmer, "Bit-mapped graphics", Windcrest, 1990.

8. I. Pitas, *"Digital Image Processing Algorithms"*, Prantice Hall, 1993.

9. T. Masters, *"Signal and Image Processing with Neural Networks"*, John Wiley & Sons, Inc., 1994, chap. 4 and chap. 5.

10. D. Vernon, *"Machine Vision"*, Prantice Hall, 1991, chap. 3-7.