An Object-Oriented Software Platform for Examination of Algorithms for Image Processing and Compression

Bogusław Cyganek and Jan Borgosz

University of Mining and Metallurgy
Department of Electronics
Al. Mickiewicza 30, 30-059 Kraków, Poland
{cyganek, borgosz}@uci.agh.edu.pl

Abstract. This paper presents a design and implementation of the innovative software development system for image processing and compression. This platform allows to make 3D image processing as well as fundamental operations on images. One of the most important features of this system, that we will focus on in this paper, is its ability to evaluate performance of different stereo matching methods. This is accomplished by a special software module that verifies machine computed disparities with values provided by a person. Additionally, due to OOP technology, the software constitutes an open architecture that allows for easy addition of new components for image processing and compression. The presented platform was verified experimentally and also compared with existing commercial packages.

1 Introduction

This paper reports the work on development of the software platform for the 2D and 3D image processing and compression. Part of the functionality of the presented software supports fundamental image operations, like arithmetic operators, filtration, etc., as well sophisticated stereo processing, scale space based methods and neural networks [1][2]. The described platform was designed and implemented by means of object-oriented programming techniques as well as with multiple design patterns [6]. Thanks to this feature, almost all system components can be modified and changed independently of each other.

The next very important feature of the presented system poses the module for qualitative measure of performance of a stereo image algorithm. This feature is accomplished by a verification of machine computed disparities with values provided by a person thus allowing for quality assessment of a given stereo method.

In this paper we present a description of those system modules that deal with stereo image processing and stereo quality verification, focusing on their functionality rather than programming details. We start from the description of the class hierarchy and end with the explanation of the user interface and the disparity verification module.

2 Description of the Software Platform

2.1 Representation of Images

A flexible data structure for image representation is crucial for efficient image processing. There must be a trade off between different input formats of images and their internal representation. Further, considering allowable size of images and time complexity of algorithms, it has been chosen that images will be represented internally as square matrices, programmatically denoted by the base template class `TImageFor<T>` [11], where `T` stands for a given data type chosen for representation of a single pixel. The class notation used hereafter complies with the UML notation [5]. Fig. 1 depicts the class hierarchy for the internal digital image representation.

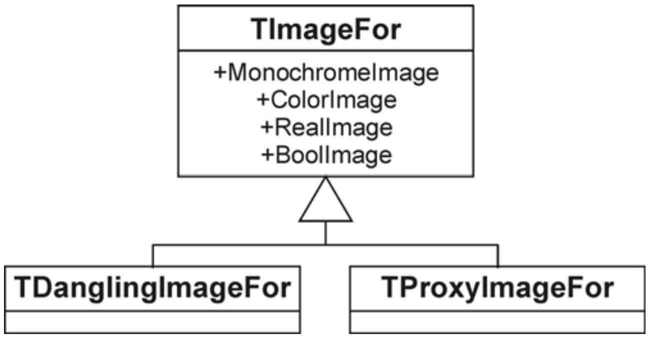


Fig. 1. The template class hierarchy for representation of digital images

The four instances of the base class in Fig. 1 are as follows: `MonochromeImage` for monochrome pixels of one-byte depth, `ColorImage` for color pixels of three-bytes depth, `RealImage` for pixels in floating point format suitable for precise computations such as convolution, and `BoolImage` for binary images.

There are two derivatives of the base class: `TDanglingImageFor` and `TProxyImageFor`. The former is used when at each pixel it is necessary to store a different length list of values rather than a single value. The latter approximates image-in-image concept that enables operations on a part of a given image as if it were a separate image by itself – however, there is only one underlying data storage. This feature creates a data abstraction that facilitates a uniform treatment of all images despite of their real representation.

2.2 Image Processing Modules

The image processing module, represented by the class hierarchy in Fig. 2, is the most important part of the evaluation platform. Due to its uniform interface, each algorithm can be used basically in the same manner, at the same time hiding all implementation details. Such an approach contributes to the platform independence and allow for easy addition of new classes implementing another image processing algorithms, such as stereo methods.

The class hierarchy interface is defined in the base class `TTwoMonochromeImageProcessor` that exposes the virtual method:

```
virtual MonochromeImage * operator() (
                 const MonochromeImage & leftImage,
                 const MonochromeImage & rightImage );
```

It is implemented in the form of the binary call operator, that takes two references to images as its input, and outputs a pointer to the outcome image [11].

The auxiliary three classes: `TAddGenerator`, `TSubtractGenerator`, and `TXORGenerator` are used simply to add, subtract or exclusively-or the two input images.

The derived class named `T_Disparity_Generator` builds a foundation for all point-oriented stereo processing algorithms [3]. All its derivatives are specializations for point-oriented stereo algorithms. These are: `T_SAD_Metric_Matching` that uses the *SAD* measure, `T_SSD_Metric_Matching` with *SSD* measure, `T_Census_Metric_Matching` that relies on *Census* measure, `T_Rank_Metric_Matching` with *Rank* measure [3]. At the same time `T_Census_Metric_Matching` is a base for all neural oriented stereo methods, such as `TNeuralCensus` that implements the Hopfield neural network and its derivatives `TCastingNetNeuralCensus` and `TCastingNetNeuralCensus` that implements a modified versions of the Hopfield neural network [4].

There are also two classes for the tensor based stereo methods [1]. The first one, named `TTensorStereo`, implements the point-oriented version of this method. The second, `Tensor_Based_Algorithm`, implements the disparity-oriented version of the tensor stereo method.

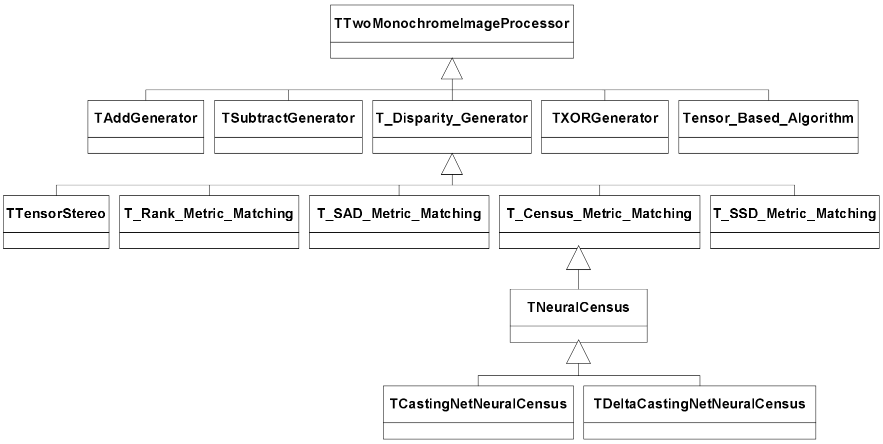


Fig. 2. The main class hierarchy for image processing modules

There are many additional classes that support the aforementioned hierarchies. All classes are well documented, so there is no trouble with the software improvements. Additionally due to the object-oriented programming style, changes in one module do not affect other software parts. This feature protects user against software bugs, and at the same time increases development speed of new components.

2.3 User Interface and Its Functionality

Fig. 3 depicts user interface of the presented system. Visible are different kinds of windows, such as: input stereo pair, disparity map, binary representation of an image, etc. User interface was implemented with the Microsoft's Visual C++ 6.0 and the Microsoft Foundation Classes. Exemplary output data from the disparity verification module are presented in Fig. 4.

The following list presents functional specification of the stereo processing related part of the software:

- loading single and stereo images in many recognizable formats,
- preprocessing on the loaded images: filtration with median, gauss, binominal and user mask, extraction of edges
- histogram generation,
- level adjusting based on histogram, brightness adjustment,
- operations like add, subtract and XOR on the loaded images,
- image value thresholding,
- correction picture geometry,
- test image generation for the stereo processing algorithms
- standard disparity calculation algorithms (*SAD*, *SSD*, *SCP*, *SSD-N*, *SCP-N*, *ZSSD-N*, *CoVar*, *Census*, *Rank, Tensor, etc.*),
- neural network based disparity calculator (Hopfield, Hamming),
- advanced tensor disparity finder,
- stereo map cross-checking module
- verification module,
- injection of different types of noise.

3 Module for Verification of Disparity Values

In practice, quality estimation of a computed disparity map is a very difficult task, because there is no objective measure and all known metrics require a true disparity map which is usually known only for synthetic stereo images. Many researchers tried to solve this problem (e.g. Ishikawa and Geiger [8], Leclerc, Luong and Fua [9]), but it still seems to be open.

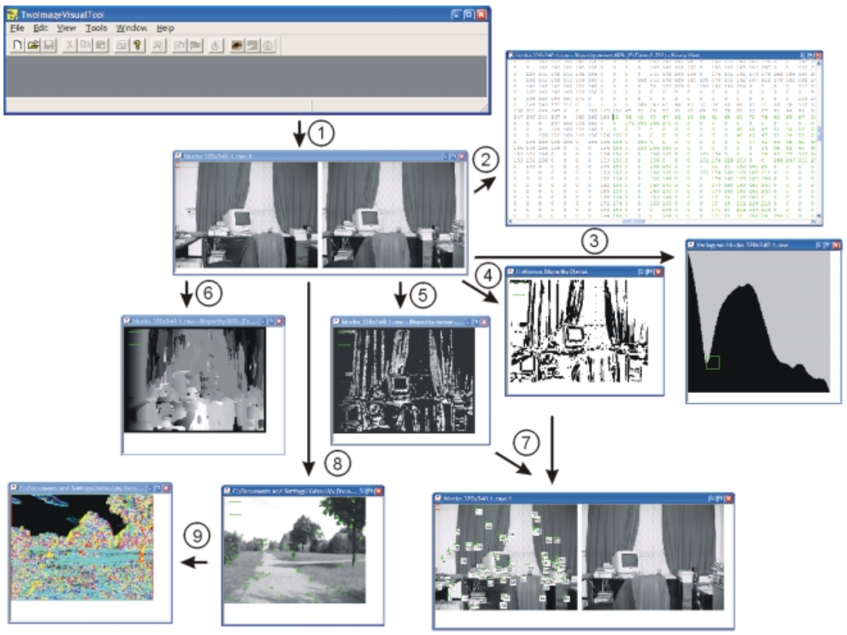


Fig. 3. Example of the view windows of the user interface: 1) loaded stereo-pair window, 2) binary view of the selected part of the image, 3) generation of the stereo–variogram, 4) tensor processing (local structures), 5) stereo tensor processing (disparity for the places with structure), 6) stereo – processing with *SAD*, 7) disparity quality verification, 8) a bitmap loaded independently and filtered with 3x3 median filter, 9) tensor angular component for the loaded bitmap

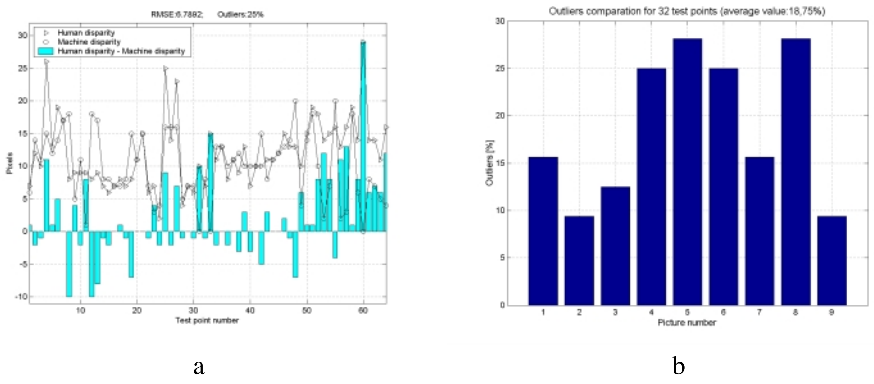
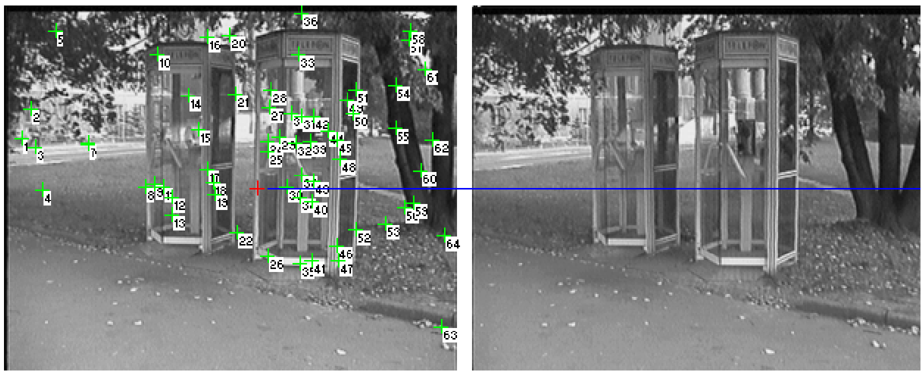
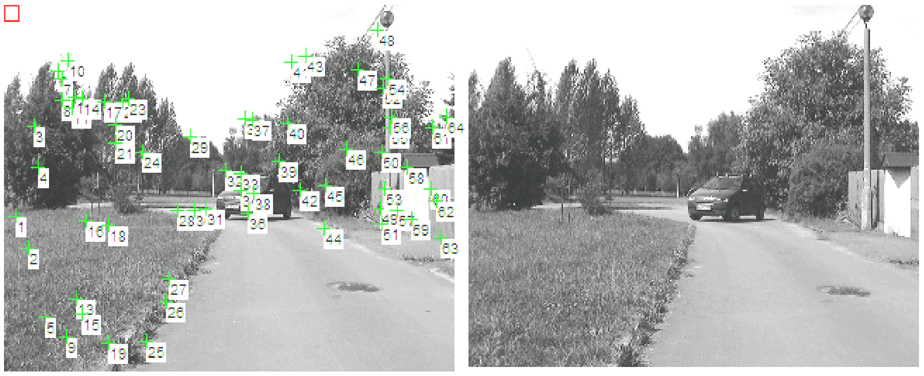


Fig. 4. Results of the disparity quality verification module: a – a bar chart with marked differences between machine and human generated disparities for test points, b – a bar chart with stereo matching outliers ratio for tested stereo-pairs



a



b

Fig. 5. Exemplary stereo images with marked test points and epipolar line (green color) for a chosen pixel: a – „*Phone*”, b – „*Punto*”

In this paper the new method of the verification and quality assessment for stereo algorithms is proposed, based on the comparison of matching results produced by a stereo algorithm and simultaneously by a testing person – similar method was proposed by Wei, Brauer and Hirzinger [12].

The disparity verification method operates as follows: test points are generated in a Monte Carlo fashion but only at image regions that are highly textured. Those areas are computed by means of the structural tensor operator [1], which is quite different and much faster than the method proposed by Wei, Brauer and Hirzinger [12]. Then disparity values are found at the generated test points. This is done independently by a testes stereo algorithm and a testing person. Both disparity sets are then compared and the final quality measure of the tested algorithm is computed. The method operates under tacit assumption that human generated disparity values outperform in quality those generated by a machine. This is true at least in an aspect of evident false matches, frequently encountered in machine stereo processing.

The presented software exposes the following interface for disparity verification:

- generation of chosen number of test points,
- epipolar line drawing for each test point,
- guided selection of a point lying on an epipolar line,
- calculation of error after testing
- export of results in text and graphical formats.

Exemplary windows involved in a disparity verification process can be seen in Fig. 5.

4 Comparison with Other Image Processing Platforms

The short comparison of the presented and other software platforms for the image processing was summarized in Table 1. The two flexible and user modifiable platforms were considered: Intel® Software Library [7] and Matlab® Image Processing Toolbox [10].

Table 1. Comparison of Image Processing Platforms

Feature	Intel® Library	Matlab®	This Platform
OS platform	Windows	Windows, Unix	Windows
User interface	Left for user development	Left for user development	Prepared, easily modifiable
Programming language	C++	Matlab script	C++
Programming style	Object	Classic/Object	Object
Library contents	From basic to advanced algorithms	Basic operations	From basic to advanced algorithms, focused on the stereo images processing
Speed	High	Low	High
Test application development cost	Medium	Low	Medium
Advanced application development cost	Medium	High	Low
Platform documentation	Rich	Rich	Rich

5 Conclusions

This paper presents structure and experimental results of the software development platform and its stereo processing capabilities. The described solution of the evaluation platform proves to be very useful and easily extensible. At the other side, execution times have been verified to be shorter than times of corresponding algorithms which run with commercially available scientific packages.

Simultaneously, a researcher has a freedom of easy addition of almost any new ideas, such as for example new neural network configurations.
The It is strongly recommended to well design all modules prior to the programming. Especially important is a choice of the interface methods chosen for each class. Also, a usage of the design patterns can greatly help and clarify the design stage.
The presented software framework was extended recently to the other image processing functions, such as image compression. In this new form it is used also by students of our faculty for implementation and testing of their image processing ideas.

Acknowledgments. This paper and its publication were sponsored by the Polish State Committee for Scientific Research (KBN) grant number 4 T11D 005 23.

References

1. Cyganek, B.: Stereo Matching with Tensor Representation of Local Neighborhoods. Ph.D. thesis (in Polish). University of Mining and Metallurgy, Kraków, Poland (2001)
2. Cyganek, B.: Neural Networks Application to the Correlation-Based Stereo-Images Matching, Proceedings of the EANN ’99, Warsaw, Poland (1999) 17–22
3. Cyganek, B., Borgosz, J.: A Comparative Study of Performance and Implementation of Some Area-Based Stereo Algorithms, CAIP'2001, Warsaw, Poland (2001) 709–716
4. Cyganek, B., Korohoda, P.: Improved Neural Network for Fast Extraction of Information from Stero-Images, Fourth Conference Neural Networks, Zakopane, Poland (1999)
5. Douglass, B.P.: Doing Hard Time. Developing Real-Time Systems with UML, Objects, Frameworks, and Patterns, Addison-Wesley (1999)
6. Gamma, E., Helm, R., Johnson, R., Vlissides, J.: Design Patterns. Elements of Reusable Object-Oriented Software, Addison-Wesley (1995)
7. Intel: Documentation for the Intel Image Processing Library, Intel (2000)
8. Ishikawa, H., Geiger, D.: Occlusions, Discontinuities, and Epipolar Lines in Stereo. ECCV’ 98, 5th European Conference on Computer Vision, Vol. 1, June (1998)
9. Leclerc, Y.G., Luong, Q.-T., Fua, P.: Self-Consistency: A Novel Approach to Characterizing the Accuracy and Reliability of Point Correspondence Algorithms. Artificial Intelligence Center, SRI International (1999)
10. Matlab: Image Processing ToolBox Documnetation, MathWorks (2002)
11. Stroustrup, B.: The C++ Programming Language. 3rd edition, Addison-Wesley (1998)
12. Wei, G-Q., Brauer, W., Hirzinger, G.: Intensity- and Gradient-Based Stereo Matching Using Hierarchical Gaussian Basis Functions. IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 20, No. 11, November (1998)