SIMULATION TOOL
FOR MULTIPLE IMAGES BASED REAL-TIME SEGMENTATION

Ionut Dinulescu, Alice Predescu, Dorin Popescu

University of Craiova, Faculty of Automation, Computers & Electronics
Department of Automation and Mechatronics
107, Decebal Street, 200440, Craiova, Romania
dorinp@robotics.ucv.ro

Abstract: The aim of this work was to develop a software framework which allows building image segmentation algorithms. A graphical simulation software built on top of this framework allows fast development and debugging each phase of the segmentation process, starting with image acquisition to final feature extraction. The object oriented architecture of the framework allows later extension, without the need to modify and re-test its existing components.

Keywords: image recognition, software framework, segmentation, moving camera.

1. INTRODUCTION

The most popular approaches to describe 3D scenes are generally based on stereo vision. As an extension of the stereo vision, researchers have built vision systems formed by more than two cameras arranged in different configurations, which have the advantage to avoid occlusions. The downside of the mentioned approaches is complexity of the matching problem.

This disadvantage can be solved by using a sequence of images from a moving camera. These days there have been a lot of activity on this subject (Chun-Jen Tsai *et al.* 1999; Yosuke Ito *et al.* 2005; Fredrik Arnell *et al.* 2005; Jezouin *et al.* 1990). For instance, [*Chang Y.L et al.,* 1990] addresses the reconstruction of 3D lines from a sequences of its 2D projections. The reconstruction problem is divided in two main stages: feature representation and estimation. While the first stage decides what features to use (in this case the infinite 3D lines) and how to represent them, the second stage tries to find relationships between 3D estimations and multiple 2D observations. For the second stage a standard recursive estimator can be successfully used, such as a Kalman filter (Chehikian A. *et al*, 1989; Martinez J.M. *et al*, 1996) or recursive least-squares.

As the segmentation process is more accurate, the complexity of the used algorithms increases, making them even hard to debug, especially when dealing with real time constraints. A simulation environment speeds up the test process of such applications as it provides 100% repeatability of the algorithm input in the form of test vectors. In addition simulators can provide profiling capabilities such as completion time of critical paths and peak run durations, which may lead to the detection of execution bottle-necks. Integration of simulators with a popular development environment can take advantage of its debugging features, especially in the case of off-line simulations, when the real-time issues (e.g. strict execution deadlines) are ignored.

Existing movement-based segmentation applications have been built using different test approaches. Some of them use synthetic images as input, while others are tested against CAD models of the considered features. A drawback of the existing test frameworks is that they are used to debug only some layers of the final segmentation application.

This paper proposes a software framework and a simulation environment for development and test of

applications that do segmentation based on sequence of images.

2. APPLICATION ARHITECTURE

All of the stages of the image processing task can be simulated in their correct order, starting from acquisition, image conditioning, and end up with feature detection and extraction. The general structure is shown in Fig. 1.The simulator acquires the sequence of images from a video-camera via an IP network and uses them to extract image features.

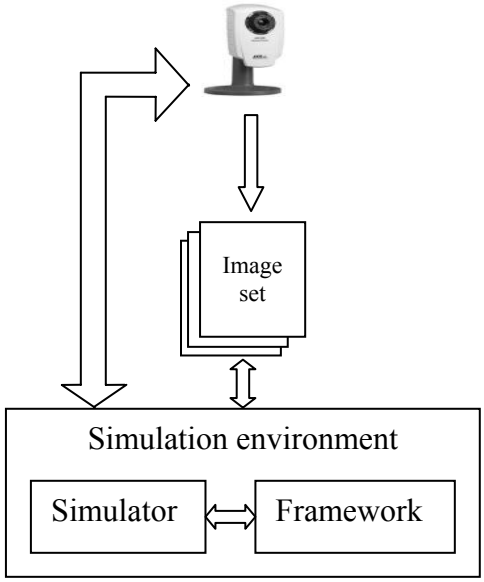


Fig. 1. Overview of the application structure.

The overall architecture of the system conforms to the Model-View-Controller (MVC) design pattern. The reasons for this approach has been used are (Dinulescu I. *et al.* 2005):

- Clarity of design: the public methods in the model stand as an API for all the commands available to manipulate its data and state. By glancing at the model's public method list, it should be easy to understand how to control the model's behaviour.
- multiple views: the application can display the state of the model in a variety of ways, and create/design them in a scalable, modular way;
- ease of growth: controllers and views can grow as the model grows; and older versions of the views and controllers can still be used as long as a common interface is maintained;

3. THE DEVELOPMENT FRAMEWORK

The framework provides all the necessary functionality, required by the segmentation process, which includes separate modules for:

- on-line image acquisition from a video camera;
- image filtering;
- feature extraction;

Additional modules provide routines that implement specific image operations, such as 2D convolution, and FFT.

A useful module is the ProfilerClass, a helper class that allows performance measurement of the employed algorithms, enabling the developer track the execution bottle-necks and find sections that need improvement.

The full object oriented architecture allows further extensions, e.g. adding new classes and/or inheritance of the existing ones, without any needed modifications in the original modules. As the UML model in Fig.2. shows, the object abstraction mechanism has been used because it allows easy addition of new classes, while preserving low-level compatibility with the existing system (Gamma E. *et al.*, 1998).

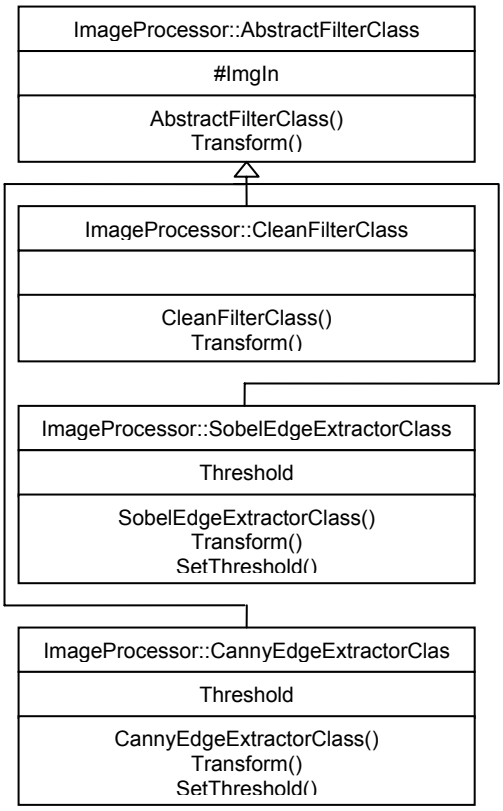


Fig. 2. Part of the UML diagram representing the filters classes. Each of the filters is a realisation of the same abstract class.

Modules that do edge detection and feature extraction have a similar class structure as in Fig.2. The framework that has been developed acts as the Controller component within the MVC design pattern (Dinulescu I. *et al.* 2005).

4. THE SIMULATION TOOL

The simulation software is an interactive application that assists the user in debugging the segmentation algorithm. It is built on top of the software framework described above and provides the following functionality via a friendly GUI interface:
- acquisition of the images sequence from the video camera;
- selection of algorithms for each phase of segmentation;
- starting the simulations;
- watch of debug messages and results.

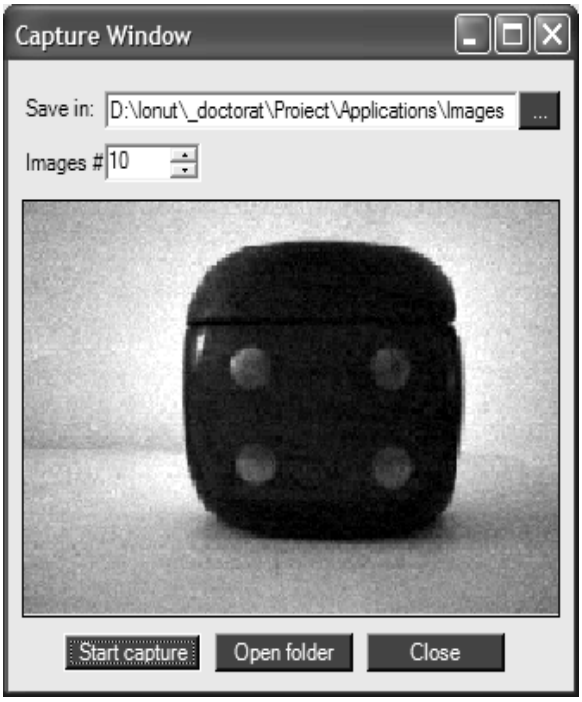


Fig. 3. The image capture window.

An additional menu allows the user test image processing algorithms individually, against a single image file. The simulator supports the most popular image files formats, such as JPG, BMP and GIF.

4.1 Capture of images sequence

The simulator allows capturing images from a video camera. The capture window is shown in Fig.3.
The specified number of images is automatically saved in the selected folder. They will be further used during the simulation.

The application provides support only for an AXIS Ethernet camera, but the software can be easily adapted to other video-camera models. The Motion JPEG (MJPEG) format that this camera supports allows downloading a stream of images with only one HTTP request. The advantage is the elimination of the overhead introduced by the HTTP request, as opposed to the simple JPEG format, which requires a separate request for each image. The downside of MJPEG, compared to JPEG is that in the first case, the frame rate is established by the camera, and it cannot be controlled that easily from the receiving application. However, for the offline simulation, the JPEG approach is still a good alternative.

Image acquisition from the AXIS camera starts with an HTTP request, which contains the IP address of the target camera. The camera response which contains the sequence of images is then parsed and each image is saved in a separate file.

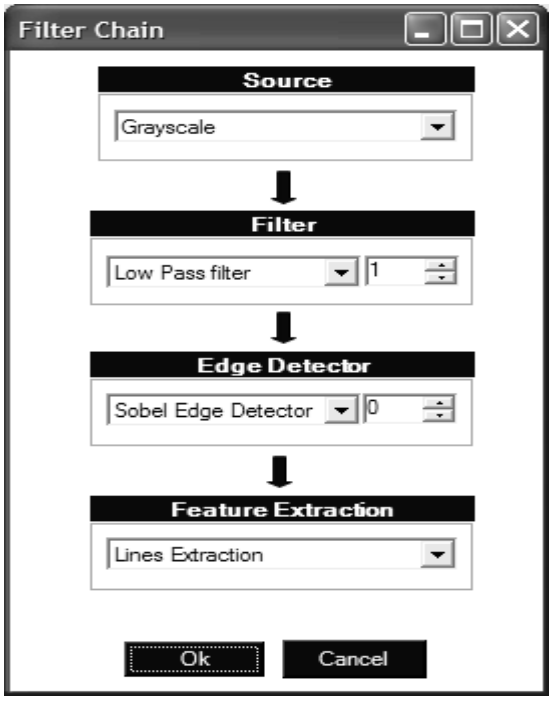


Fig. 4. The filter chain window. The user can customize the segmentation algorithm, by choosing the appropriate algorithm for each phase of the process.

4.2. The simulation

The user can customize the segmentation process, by selection of the appropriate algorithms for each phase, as shown in Fig. 4. The source image, can be whether one of the R, G, B channels, the greyscale or the binary form of the original image. The greyscale image is obtained by replacing each of the R,G,B channels of the current pixel (x,y) with the value determined by the relation (1):

$$I_{GS}(x,y) = 0.299 \cdot R(x,y) + 0.587 \cdot G(x,y) + 0.114 \cdot B(x,y) \quad (1)$$

where I_{GS}, R, G, B are the greyscale, red, green and respectively blue values of pixel (x,y).

Image conditioning
After selecting the image source, the algorithms for pre-filtering and edge detection can be selected, along with the threshold. If images in the sequence have already been processed previously, the user has the option to bypass these two phases and skip to the selection of features to be extracted.

For the pre-filtering stage, one of the following image enhancement algorithms can be selected:
- Low-pass filtering
- High-pass filtering
- Histogram equalization

The Butterworth filter has been chosen as the low-pass filter as it doesn't introduce the double edge effect. Its transfer function is shown in equation (2):

$$H(u,v) = \frac{1}{1 + (\sqrt{2} - 1)\left[\frac{D(u,v)}{D_0}\right]^{2n}} \qquad (2)$$

where D_0 is the cutting frequency, and n represents the order of the filter (Ritter G. *et al.* 1996).

For high-pass filtering, the transfer function of the Butterworth filter becomes (3):

$$H(u,v) = \frac{1}{1 + (\sqrt{2} - 1)\left[\frac{D_0}{D(u,v)}\right]^{2n}} \qquad (3)$$

Feature detection
Detection of features in the image sequence is the last stage prior to recognition. In the case of real images feature detection may be not reliable due to occlusions, so appropriate feature sets need to be chosen. Hafez W. 1999 proposes the development of a set of motion-based invariant features.

Currently, lines are the only features supported by the simulator. New feature extractors can be easily implemented by inheriting from the AbstractFeature Extractor class.

For straight line detection, the Hough transform is used (Cojocaru D. 2002). The Hough transform starts from edge detection based on local differential properties of the image (Zhou J. et al 2006). Lines in (x,y) image space are translated into points in the Hough (ρ, θ) space, according to the normal parameterization proposed by Duda *et al.* 1972:

$$\rho = x \cos\theta + y \sin\theta \qquad (4)$$

where $0 \le \theta \le \pi$ and $\rho \in [-\rho_{lim}, \rho_{lim}]$. Peaks in the parameter space correspond to collinear points in the image plane.

Hunt 1988 treats the Hough transform as a statistical signal detection problem by taking noise in consideration. In this case the Hough transform is formally described by the relation (5):

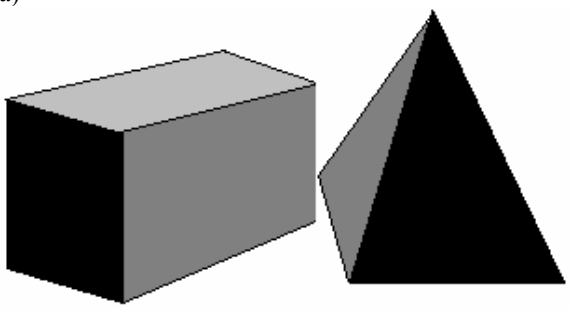
$$H : X = S(\rho, \theta) + N \qquad (5)$$

where X and N represent the edge image and noise, respectively, while $S(\rho,\theta)$ is the image that contains only the line (ρ,θ). The line is detected by using Bayes statistical approach, described by Chang L. *et al.* 1990.

For speeding up the process, the image gradient is used so the Hough transform is calculated only for points along the gradient direction (Cojocaru D. 2002; Princen 1990).

For the input image in Fig.5.a, after applying the Hough transform, the detected straight lines are shown in Fig.5.b.

a)



b)

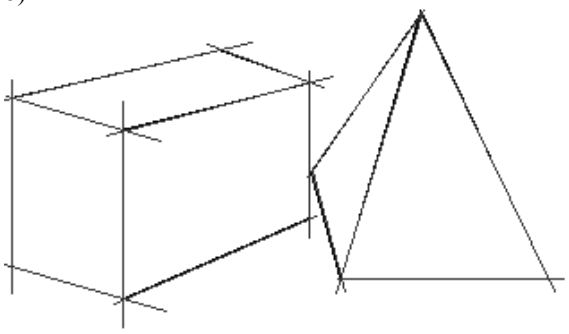


Fig.5. a) Source image; b) Straight lines detected with the Hough transform.

In addition, the simulator provides the option of tracking the evolution of points in the Hough space, as the camera moves.

5. USED TECHNOLOGIES

For defining the application architecture, Microsoft Visio UML editor has been used. Adding new classes can be done via the same tool, which is able to re-generate the source code keeping it synchronized with the existing one.

C# has been chosen as the programming language, as it is full object-oriented and has support for fast creation of graphical user interfaces. Although C# provides basic image processing routines, custom routines have been developed, for keeping the source code portable to other platforms.
Currently the Axis 205 Ethernet video camera is supported, although the software can be adapted to any other model. The main advantage of an Ethernet camera is that it can be directly accessed over on IP network, without the need of a host computer which may decrease real-time performance.

6. CONCLUSION

Using an offline simulator allows fast debugging of the image processing algorithm, by restoring the same context of the image interpretation (the role of the context in pattern recognition has been well illustrated by Suentes P. *et al.* 1992).

This article introduced a complete software framework and a simulation tool which is currently being used for development and evaluation of motion-based image segmentation algorithms. New algorithms can be easily added without internal modifications of the application's architecture.

The software framework is still subject to improvement. As further development directions, some new features are being considered, which include modules for object recognition and on-line simulation.

REFERENCES

Chang Y.L. and Aggarwal J.K. (1990). Reconstructing 3D Lines from Sequence of 3D Projections: Representation and Estimation, *3rd International Conference On Computer Vision*, pp.101-105.

Chehikian A., Stelmaszyk P., De Paoli S. (1989). Hardware Evaluation Process for Tracking Edge-Lines, *International Workshop On Industrial Applications of Machine Intelligence and Vision,* pp. 332-336.

Chun-Jen Tsai and Aggelos K.Katsagellos (1999). Sequential Construction Of 3D-Based Scene Description, *IEEE International Conference On Image Processing ICIP*, **vol.2**, pp. 512-514.

Cojocaru D. (2002). Achizitia, prelucrarea si recunoasterea imaginilor, pp.129-132, Universitaria, Craiova.

Dinulescu I., Popescu D., Terejanu G. and Marinescu A. (2005). Web Based Telematic Application Using Open-Source Technologies, *SINTES*, Craiova.

Duda R. and Hart P. 1972. Use of Hough transformation to detect lines and curves in pictures, *Communications of the ACM*, **vol. 15**, pp.11-15.

Fredrik Arnell and Lars Petersson (2005). Fast Object Segmentation from Moving Camera, Proceedings of *Intelligent Vehicles Symposium*, pp. 136-141.

Gamma E., Helm R., Johnson R. and Vlissides J. (1998). Design Patterns. Elements of Reusable Object-Oriented Software, pp.100-101, Addison-Wesley, Baarn, Holland.

Hafez W. (1999). Invariants for motion-based segmentation, *Proceedings of American Control Conference*, **Vol.4**, pp. 2925-2930.

Hunt D., Nolte L. and Ruedger W. (1998). Performance of the Hough transform and Its Relationship to Statistical Detection Theory, *Image and Vision Computing*, **vol.6**, no.2, pp.87-90.

Jezouin J. and Ayache N. (1990). 3D Structure From A Monocular Sequence of Images, *Proceedings of 3rd International Conference On Computer Vision*, pp. 441-444.

Yosuke Ito and Hideo Saito (2005). Free-Viewpoint Image Synthesis From Multiple-View Images Taken With Uncalibrated Moving Cameras, *IEEE International Conference On Image Processing ICIP 2005*, pp. 29-32.

Martinez M., Zhang Z. and Montano L. (1996). Segment Based Structure from an Imprecisely Located Moving Camera, *International Symposium On Computer Vision*, pp. 182-187.

Princen J., Illingworth J., and Kittler J. (1990), An optimizing line finding using a Hough transform algorithm. *Computer Vision, Graphics and Image Processing*, pp. 52:57-77.

Ritter G. and Wilson J. (1996). *Handbook of Vision Algorithms in Image Algebra*, pp. 130-132, CRC Press, Boca Raton, FL, USA.

Suentes P., Fua P., and Hanson A. (1992). Computational Strategies for Object Recognition, *ACM Computing Surveys*, **Vol. 24**, pp. 5-62, ACM Press, NY, USA.

Zhou J., Bischof W.F. and Sanchez Arthuro (2006). Extracting Lines in Noisy Image Using Directional Information, *The 18th International Conference on Pattern Recognition ICPR'06*., **vol. 2**, pp. 215-218