Lavatube – A Software Framework for Computer Vision Research and Development

Kenji Iwata, Yutaka Satoh and Katsuhiko Sakaue
National Institute of Advanced Industrial Science and Technology (AIST)
AIST Tsukuba Central 2, 1-1-1 Umezono, Tsukuba, Ibaraki 305-8568 Japan
kenji.iwata@aist.go.jp

Abstract

We develop Lavatube as a software framework for efficient research, development, and education. Lavatube is an object-oriented framework optimized for constructing a computer vision system, particularly a video and image processing system. Lavatube enables a description of a processing extension by combining various functional components. Since the data flow is easy to describe by graph-connecting icons on a GUI, a system can be created intuitively. The standard support of input from a USB camera and various video files makes complicated work or knowledge unnecessary. For efficient and clear-cut system construction, Lavatube also provides functions for the dynamic generation of parameter setting dialog boxes and for perpetuation by XML. Some actual video processing cases as examples are also introduced.

1. Introduction

A computer vision system captures information from the external world as image data by using a camera and other devices, and analyzes the images on a computer for studying or measuring [1]. This kind of system is used as a major means of inspection, especially for semiconductors and electronic boards, because objects can be measured without contact. The range of applications, such as security, robot vision, medicine, welfare, and sports, has been growing even more in recent years.

A computer vision system that is expected to be applied so widely requires high-level knowledge and programming techniques for its design and adjustment. So, at a company or a university, what kind of abilities should be acquired to learn this system from the beginning? There is a lot of substantially important knowledge, such as statistics and geometry. In reality, however, he or she faces such problems as difficult programming for image acquisition from a camera or time-consuming analysis of a predecessor's program that requires great overhead.

Even an experienced person tends not to be committed to programming because it requires a lot of time to construct a framework where arbitrary processing can be visualized and various parameters can be adjusted in real time. This consequently makes it necessary, yet sometimes impossible, to evaluate installed algorithms and set parameters satisfactorily.

To solve these problems, we developed Lavatube as a visual programming framework. Lavatube has the following advantages:

(1) A system can be constructed easily by graph-connecting icons that express various functions on a graphical user interface (GUI), such as the one shown in figure 1.
(2) By arranging icons with the capturing function, a USB camera and various video files (AVI, MPEG, etc.) can be used as input. This realizes online and offline experiments using previously recorded videos on a single platform.
(3) Parameter setting dialog boxes can be generated dynamically and adjusted by viewing of the processing results online.
(4) Persistence by XML enables storage and reproduction of working environment.
(5) The framework operates at high speed with extremely small overhead. Since parallel flows are automatically threaded for parallel execution, this programming makes processing even faster than ordinary programming.

Previously, the image processing environments, such as Khoros [2] and XITE [3] have been developed. More recently however, many visual programming environments for processing flows visually through this type of GUI have been available commercially. For example, MAX/MSP [4], which is mainly used by artists, is a visual programming environment extended from voice processing to video processing. Other marketed products include MATLAB/Simulink [5] for simulation and AVS/Express [6] for visualization. These are basically visual programming environments in which existing modules are combined to create a processing flow. If there are not enough modules, extension modules are created by using C or other languages. Since existing modules are not adequate for creating new algorithms, programmers need to create extension modules.

Lavatube was developed not as a visual programming environment via a GUI but rather as an object-oriented framework optimized for video and image processing. Lavatube is designed so that users can develop original function extensions easily by combining various functional components. These patches are compatible with

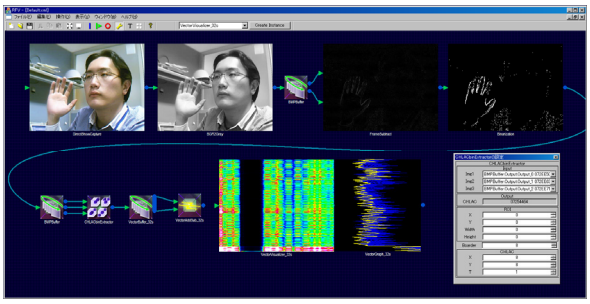


Figure 1. A GUI of Lavatube

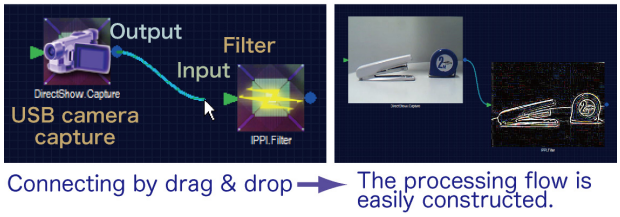


Figure 2. Description of data flow

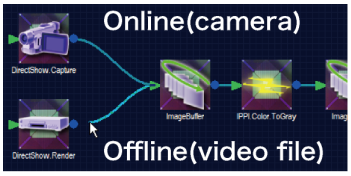


Figure 3. Switching online to offline environment.

GUIs, XML, and other support functions, and realize an efficient research and development environment, in which such functions are immediately available as a user interface, visualization, and perpetuation.

This paper outlines the Lavatube functions and introduces actual cases.

2. Outline of Lavatube

Lavatube is an object-oriented framework that supports research on video and image processing, computer vision, and also trial application programming and tuning. The characteristic functions of Lavatube are described below.

2.1 Description of Data Flow by the GUI

In the GUI of Lavatube, each process (called a "work patch") is expressed as an icon. By connecting the input and output of each icon using a mouse, a data flow can be described very easily. Figure 2 shows an example of a data flow description. In this example, the image output from the USB camera is connected to the input of the contour detection filter.

This type of GUI makes it easy to partially modify and add processes. For example, online experiments and offline experiments can be easily realized in the same environment by switching the source of image input immediately to a camera or a video file, as shown in figure 3.

2.2 Extensibility

A program is described in small units called patches, and Lavatube operates by interpreting these patches automatically. As figure 4 shows, work patches are created by combining patches of functions such as data I/O, parameters, and by describing a processing procedure on the object interface. Any user experienced in C++ programming can easily create work patches of arbitrary functions.

At present, Lavatube has functions for image capturing from a USB camera, basic filtering, and arithmetic operations. The high extensibility allows for easy addition of functions to use other types of cameras.

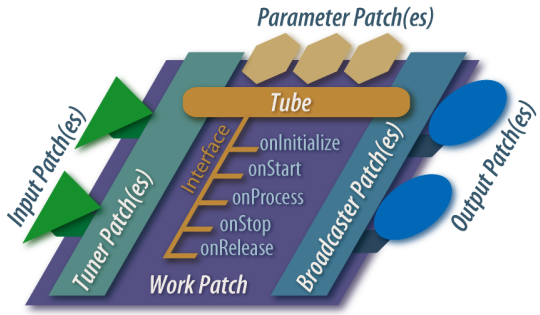
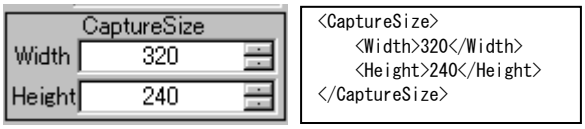


Figure 4. Structure of “work patch”



(a) Dialog (b) XML output

Figure 5. Parameter setup

2.3 Storage and Reproduction of Working Environment by XML

Lavatube can output a working environment including a data flow and parameters to an XML file for storage. The working environment can be completely reproduced by loading the XML files. As the system becomes complicated in ordinary programming, this processing becomes time-consuming and often causes a programming error. In Lavatube, each patch has a function that can be realized easily by separating the description. This function enables verification by reproducing the environment, as well as later additions or modifications of functions.

2.4 Parameter Setup via the GUI

Lavatube provides a GUI through which parameters can be adjusted easily. Since dialog boxes, such as figure 5(a), are created dynamically, programmers are free to use GUIs for additions or modifications without using a GUI builder or other software. The dialog boxes are automatically generated by describing any parameter in parameter function patches. Since parameter sets determined via the GUI are stored in an XML file, such as figure 5(b), a description of constant parameters can be separated from the source code to improve program maintainability.

2.5 Visualization of Operation Status

The real-time display of image data on a data flow allows the user to visually check the operation status sequentially for efficient parameter tuning and other tasks.

For real-time demonstration, Lavatube is designed to

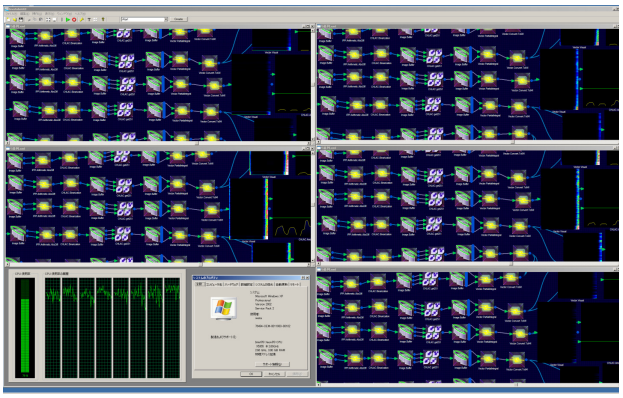


Figure 6. A parallel operation example

optimize the processing overhead and to operate the constructed system extremely quickly. The automatic parallel processing of tasks in each work patch is optimum for multi-core processors, which are becoming more widespread. Figure 6 shows a parallel operation example. In this figure, 25 tasks are operate on 8 core PC.

3. Actual Cases

3.1 Wrapping external libraries

By wrapping OpenCV and other external libraries as work patches, Lavatube can handle them easily. Figure 7 shows an example of detecting a human face by using the Viola-Jones face detector [7]. The left-side image is a work patch captured from a camera. According to the flow from there to the lower stage, color-to-gray image conversion and facial detection patches are connected. Since the facial detection patch outputs the position coordinates and dimensions of a face, it is connected to a work patch that draws a circle at the face position. This work patch is created to receive image and coordinate inputs.

Figure 8 shows an example of optical flow estimation by using the Lucas-Kanade method. Because optical flow estimation requires current image and previous image, a work-patch that buffers previous images is connected to an optical flow patch.

3.2 Background / Foreground Segmentation

The background/foreground segmentation is a fundamental and important problem for vision systems, such as video surveillance systems. The simplest method is subtracting the current image from the background image. However the simple method is susceptible to illumination change such as shadows. Radial Reach Correlation (RRC) [8] evaluates local textures and realizes robust background/foreground segmentation. Other approach uses the pixel intensity distribution, such as Gaussian mixture model, estimated from the past images. Tanaka et al. proposed a fast estimation of the pixel intensity distribution using Parzen density estimation [9].

Implementation of background/foreground segmentation using simple method, the RRC and the Parzen

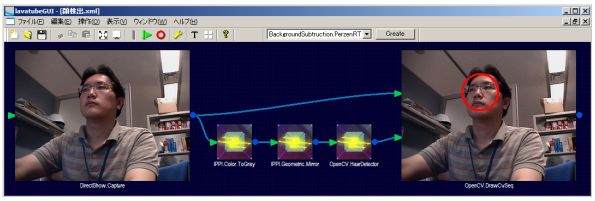


Figure 7. Face detection

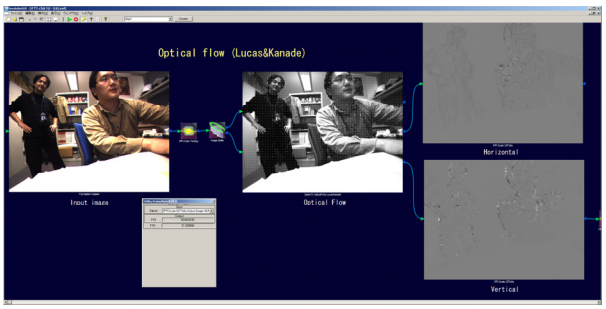


Figure 8. Optical flow estimation

estimation on Lavatube is shown in figure 9. The simple method procedure is created by connecting work patches for gray image generation from color image, calculation of absolute difference, and binarization. The RRC and the Parzen estimation method is implemented respectively as a work patch. RRC work patch inputs a gray-scale current image and a gray-scale background image. Parzen estimation work patch inputs color images.

Each method has some parameters that require adjusting, such as threshold of binarization. Lavatube generate dialog boxes automatically to adjust these parameters. Multiple data flow with different parameter values can be processed in parallel. In figure 7, each method operates by two kinds of the parameter values. Each parameter values can be adjusted in real-time by using the dialog boxes. Therefore, optimum values can be verified instantaneously for various parameters that differ depending on the environment.

PETS2001[1] is used in figure 7. The data is outdoor video sequence including moving people under significant lighting variation. An outdoor vision application needs to be robust against weather and sunlight changes. The simple method cannot detect people because of the illumination change. Because RRC uses not the pixel intensity but the texture, the moving objects, such as people and clouds, are detected. Because Parzen estimation uses past image sequence to estimate the background model, only moving people are detected.

In this way, users can compare two or more methods, and confirm the parameter values and the processing results at the same time to efficiently construct the vision system.

4. Conclusion

This paper outlines Lavatube as a visual framework to support trial programming and research for computer vision or video or image processing by a GUI and XML, and introduced some cases as examples. In the future, we

[1] Available at ftp://pets.rdg.ac.uk/.

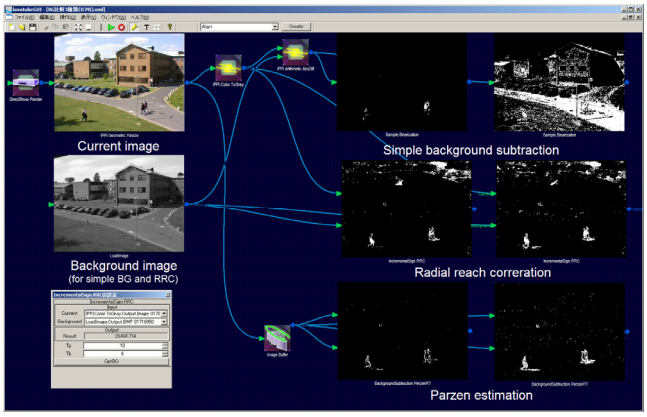


Figure 9. Implementation of background subtraction

want to improve the user interface by integrating multiple processing using a macro.

This software is released under GNU General Public License (GPL). We hope the software will be of help to researchers and educators in related fields.

References

[1] D. A. Forsyth and J. Ponce, *Computer Vision: A Modern Approach*, Prentice Hall, 2003

[2] K. Konstantinides and J. R. Rasure, *The Khoros Software Development Environment for Image and Signal Processing*, IEEE Trans. on Image Processing, Volume 3, Issue 3, pp.243–252, May 1994

[3] O. Milvang and T. Lonnestad. *An Object Oriented Image Display System*, Proc. of 11th ICPR, pp. 218-221, October 1992

[4] Max/MSP, http://www.cycling74.com

[5] MATLAB/Simulink, http://www.mathworks.com

[6] AVS/Express, http://www.avs.com

[7] P. Viola and M. Jones. *Rapid Object Detection Using a Boosted Cascade of Simple Features*, Proc. of CVPR2001, vol.1, pp.511-518, December 2001

[8] Y. Satoh, H. Tanahashi, C. Wang, S. Kaneko, Y. Niwa and K.Yamamoto, *Robust Event Detection by Radial Reach Filter (RRF)*, Proc. of 16th ICPR, vol.2, pp.623-626, August 2002

[9] T. Tanaka, T. Shimada, A. Arita and D. Taniguchi, *A Fast Algorithm for Adaptive Background Model Construction Using Parzen Density Estimation*, Advanced Video and Signal Based Surveillance 2007 (AVSS 2007), pp. 528-533, September 2007