VIPERS: Visual Prototyping Environment for Real-Time Imaging Systems

Frederic Jean
Genie Electrique et Informatique
Laval University
Quebec, Canada
fjean@gel.ulaval.ca

Alexandra Branzan Albu
Electrical and Computer Engineering
University of Victoria
BC, Canada
aalbu@uvic.ca

ABSTRACT
This paper presents a novel environment for the visual design and prototyping of computer vision systems. The proposed environment consists of a modular architecture, where each module instance performs a pre-specified task. This is well suited for the implementation of computer vision systems, which are typically highly modular. A graphical user interface allows for the visualization and comparison of intermediate results, as well as for the design of complex systems by interconnecting modules.

Categories and Subject Descriptors
I.4 [**Image Processing and Computer Vision**]: Miscellaneous

General Terms
Algorithms, Design

Keywords
computer vision, real-time imaging systems, visual prototyping

1. INTRODUCTION
Computer Vision systems are needed for a variety of applications, ranging from medical imaging to intelligent vehicles, video surveillance, and perceptual user interfaces. During the last two decades, significant progress has been made in major areas of computer vision, with numerous robust algorithms being developed for image enhancement, segmentation, motion tracking and object recognition. Implementations of such algorithms are available through the MATLAB Image Processing Toolbox, the OpenCV library [1], or through academic repositories of code such as RAVL [2]. Therefore, we witness a significant change of focus in computer vision research, from algorithm creation towards software systems design and prototyping. The task of integrating existing algorithms into a functional system is not trivial, since one needs to program the 'glue' code to link these algorithms. Moreover, experimenting with different algorithms designed for the same task is not straightforward.

Few attempts were made about the fast development and prototyping of computer vision systems. The Imalab [3] framework provides only a minimal graphical user interface (GUI), and most of the development and the integration still need to be done at the code level. By contrast, the iceWing architecture [4] provides a complete but complex GUI. The modules in this architecture put their labeled outputs data in a "data pool" so that any other modules can directly use them as inputs. This data pool method does not provide an easy way to visualize intermediate results of the system and to modify its components.

This paper addresses limitations of previous work by proposing VIPERS (Visual Prototyping Environment for Real-time Imaging Systems). VIPERS was built for facilitating the design, visualization, and prototyping of computer vision systems. This environment works well for any system that can be described by a modular architecture. Each algorithm is represented by a module, which requires a certain number of inputs, outputs, and parameters. Module instances are created and interconnected graphically. Their parameters can be modified while the system is running, and their output can be visualized in real-time. This is consistent to the WYSIWIG paradigm (what you see is what you get), and it is extremely helpful in comparing the performances of different algorithms on the same input data, as well as in evaluating the impact of parametric values on a given module's output. The remainder of the paper is organized as follows. The proposed system is described in section 2. Results are presented in section 3. Finally, section 4 draws conclusions and outlines the main directions of future work.

2. PROPOSED APPROACH
An overview of the proposed environment is presented in Fig. 1. The environment consists in four components: the modules, a library, a graphical user interface and a command line interface. The environment and its tools are open source [5] and are released under the GNU General Public License (GPL) version 3. VIPERS can be run under Linux™, Windows™, and MacOS X™.

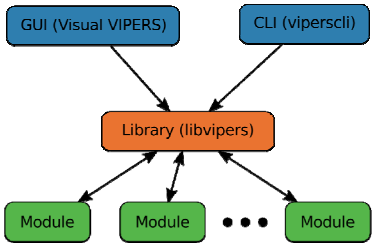


Figure 1: Overview of the VIPERS environment

2.1 Modules
A module represents an implemented computer vision algorithm that requires image data as input and that generates one or more output images. It can also represent any process that takes image data as input (e.g. a video file) or provides images as output (e.g.

Copyright is held by the author/owner(s).
SOFTVIS'10, October 25–26, 2010, Salt Lake City, Utah, USA.
ACM 978-1-4503-0028-5/10/10.

camera, video file, etc.). Custom modules can be created as standard dynamic libraries simply by deriving as base C++ class. A module may also require a set of parameters that can be specified by the user in the graphical interface.

2.2 Library

The library is a core component of the framework since it provides all functionalities. It is responsible for the module instance creation and deletion as well as for controlling the execution flow while the system is running. The execution flow is handled by a kernel, which first determines the order for asking the modules to process their inputs and generate their outputs. The kernel achieves this by analyzing the graph of the module interconnections. The modules are then called in the determined other and the kernel will loop until the image input is stopped or until the user pauses or stops the execution flow.

The library also enables the user to load and save a system design in an XML file format. The VIPERS XML files contain information about the instantiated modules, their interconnection graph, and the value of their parameters.

2.3 Graphical User Interface (GUI)

The GUI enables the user to design and prototype the system, as well as to visualize its dataflow. A screenshot of the GUI is shown in Fig. 2a. The left part of the GUI provides the list of available modules, the middle part (design area) shows the instantiated modules and their interconnections, and the right part shows the list of module parameters. A module is instantiated by dragging its name from the list on the left and dropping it in the design area. The output of a module can be connected to the input of another module by the same drag-and-drop action. The GUI also provides menu and buttons for loading/saving the design and for controlling the execution flow (start, stop, pause).

A novelty of the GUI consists in the presence of monitor windows (see Fig. 2 b, c). Monitor windows can be created at any time for visualizing the output of a module in real time. The usefulness of monitor windows relates to the ability to visualize intermediate results, as well as to compare results generated by algorithms in the same class running simultaneously on identical input data.

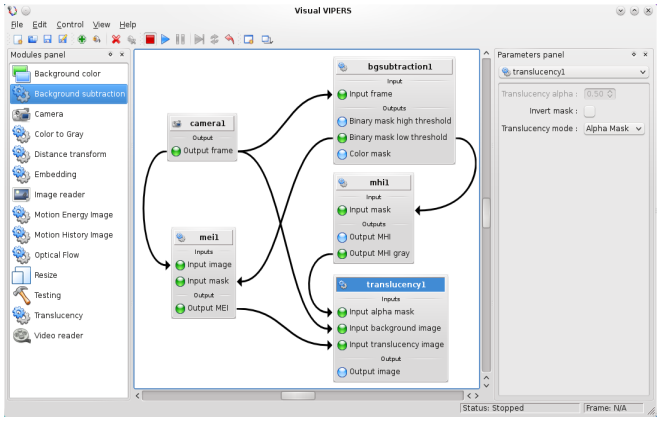
2.4 Command Line Interface

For efficiency of interaction, a command line interface (CLI) is provided as well. The CLI takes a VIPERS XML file as input and starts the execution of the designed system. The CLI is useful in scripts to perform batch processing with the designed systems.

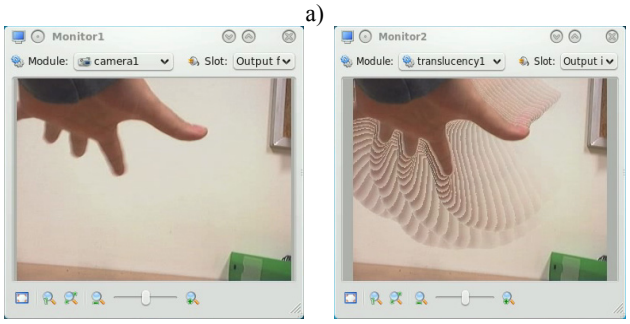
2.5 Results

While the VIPERS environment is fully functional and stable, its evaluation is not completed yet. As a qualitative partial evaluation, we present an example of the use of VIPERS for the design of a simple computer vision task. The system takes colour images (Fig. 2b) from a camera, finds the foreground pixels (background subtraction), and then, with the help of motion energy (MEI) and motion history images (MHI), shows a history of the previous foreground pixels using transparency cues (Fig. 2c). Implementing this in C++ using OpenCV would not be a rapid task, but with the proposed framework it is easy and fast.

It is important to note that in the proposed framework, no copy of the images are performed between the modules, that is, the module works with references to the images. Given that the



a)



b) c)

Figure 2: a) Visual VIPERS (GUI); b) monitor window showing the output of the camera module; c) monitor window showing the output of the translucency module

modules themselves are built in C++ with libraries like OpenCV (which is fast and optimised), it is then possible for the proposed framework to process images in real-time.

3. CONCLUSIONS

This paper presents an environment for the rapid visual prototyping of computer vision systems. We believe that this environment will benefit both novice and experienced computer vision scientists. Future work will focus on a comprehensive evaluation of the proposed system design tool, as well as on the expansion of VIPERS by allowing other types of input/output in the modules and by exploiting parallelism for module processing. Bindings with interpreted languages such as Python will be also considered.

4. REFERENCES

[1] OpenCV (0pen source computer vision)

http://opencv.willowgarage.com, 2010.

[2] RAVL (Recognition and vision library). University of Surrey, UK. http://www.ee.surrey.ac.uk/CVSSP/Ravl/, 2010.

[3] F. Lomker, S. Wrede, M. Hanheide, and J. Fritsch. Building modular vision systems with a graphical plugin environment. In Proc. of the Int. Conf. on Computer Vision Systems, 2006.

[4] A. Lux. The imalab method for vision systems. In Proc. of the Int. Conf. on Computer Vision Systems, 2003.

[5] F. Jean and A. B. Albu. VIPERS: Visual Prototyping Environment for Real-Time Imaging Systems. http://vipers.googlecode.com, 2010.