Design and Implementation of an *What*-Oriented Open Vision Library

Changsong Shen, Steve Oldridge, Gregor Miller, Amir Afrah, and Sidney Fels
Department of Electrical and Computer Engineering, University of British Columbia

MOTIVATION: HOW VERSUS WHAT

Typical vision API's expect programmers to specify **how** to solve a problem. This leads to:

1. Difficultly creating compile and run-time acceleration algorithms since details of the processes are already specified leaving little freedom for optimization.

2. Poorly exploited best-available technologies due to programmers not being aware of latest techniques.

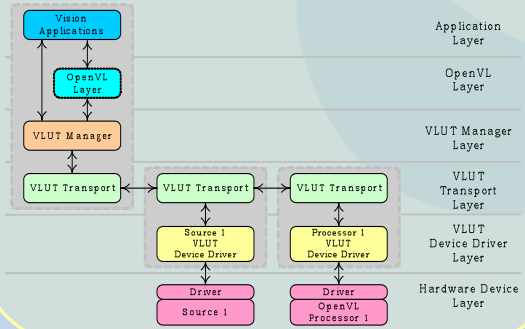
Effectively, programmers need an API to specify **what** needs to be done.

Under our ***what***-oriented methodology for vision processing, Open Vision Library (OpenVL) encapsulates interchangeable behaviors through specification of a state machine. Application developers decide what they want done by controlling the states leaving the compiler/run-time implementations determine **how** to do it.

OpenVL + VLUT =

A Portable Hardware-Accelerated Vision System

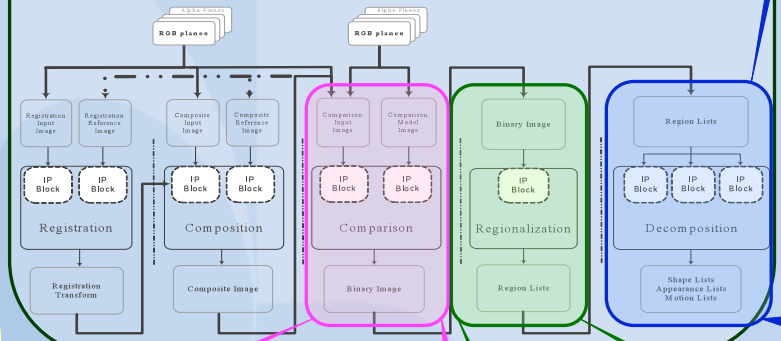
OpenVL provides an abstraction layer for application developers to specify the vision processing they want performed rather than how they want it performed. VLUT is created as a middleware layer to separate camera details, events management and operating specific details from the specification of the computer vision. By providing a hardware-oriented middleware that supports different hardware architectures for acceleration, OpenVL allows portability without compromising performance.



OpenVL Architectural Design

Input data to OpenVL is stored in an abstracted image block, which include RGB planes. Output data is stored in layers of various type, with each block potentially adding a new layer to the available set of data associated with the input image.

OpenVL consists of five main processing blocks. These are Registration, Composition, Comparison, Regionalization, and Decomposition. By connecting and configuring each of these components, a wide range of image processing and understanding tasks can be accomplished.



Decompositor

In the decomposition abstraction, an object can be represented by its shapes, appearances. and the motion of the objects can be represented by motion model. In User can specify properties of objects, and then the decompositer finds elements that match the description.

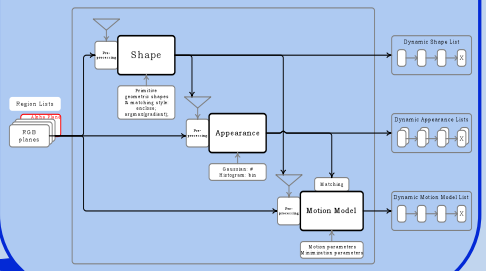
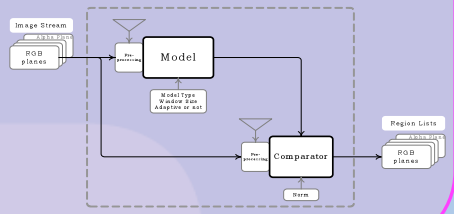


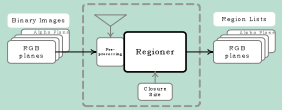
Image-Based Comparator (IBC)

In the image-based comparison abstraction, users are asserting whether or not pixels (tokens, etc.) belong to a particular user-defined model, such as a statistical model. Using this approach, users also need to declare some criteria for whether or not the pixel is a good fit.



Regioner

In a regionalization abstraction, pixels in binary input images are grouped into different regions according to user defined criteria.



Summary

Open Vision Library (OpenVL) consists of a language model to support computer vision applications that are reusable, scalable and can be accelerated by hardware by different vendors. The design and implementation of OpenVL has migrated from orienting towards the application programmer specifying **how** a computer vision algorithm is implemented to **what** they want done. Users benefit from these semantics with improved performance, reusability and scalability.

Some example algorithms have been developed as a proof-of-concept to demonstrate the OpenVL API syntax and some of our architecture's critical concepts.

Human Communication Technologies Laboratory

Acknowledgments: This research is funded by Precarn/IRIS, NSERC and Bell University Lab.