Chapter 10
Towards OpenVL: Improving Real-Time Performance of Computer Vision Applications

Changsong Shen, James J. Little, and Sidney Fels

Abstract Meeting constraints for real-time performance is a main issue for computer vision, especially for embedded computer vision systems. This chapter presents our progress on our open vision library (OpenVL), a novel software architecture to address efficiency through facilitating hardware acceleration, reusability, and scalability for computer vision systems. A logical image understanding pipeline is introduced to allow parallel processing. We also discuss progress on our middleware—vision library utility toolkit (VLUT)—that enables applications to operate transparently over a heterogeneous collection of hardware implementations. OpenVL works as a state machine, with an event-driven mechanism to provide users with application-level interaction. Various explicit or implicit synchronization and communication methods are supported among distributed processes in the logical pipelines. The intent of OpenVL is to allow users to quickly and easily recover useful information from multiple scenes, in a cross-platform, cross-language manner across various software environments and hardware platforms. To validate the critical underlying concepts of OpenVL, a human tracking system and a local positioning system are implemented and described. The novel architecture separates the specification of algorithmic details from the underlying implementation, allowing for different components to be implemented on an embedded system without recompiling code.

10.1 Introduction

Computer vision technology is profoundly changing a number of areas, such as human-computer interaction and robotics, through its interpretation of real-world scenes from two-dimensional projections. However, building computer vision
systems remains difficult because of software engineering issues such as efficiency, reusability, and scalability. Especially when computer vision technology is applied in embedded systems, in which real-time performance is emphasized, these issues become critical. In a field with as rich a theoretical history as computer vision, software engineering issues, like system implementation, are often regarded as outside the mainstream and secondary to the pure theoretical research. Nevertheless, system implementations can dramatically promote the progress and mainstream applicability of a field, just like the success of OpenGL promoted the development of hardware acceleration coupled with significant theoretical progress in computer graphics.

In current computer vision, there are three main system implementation issues. The first issue is efficiency. Most video operations are computationally intensive tasks that are difficult to accomplish using traditional processors. For example, for a single camera with a sequence of 24-bit RGB color images at a typical resolution (640×480 pixels) and frame rate (30 fps), the overall data volume to be processed is 27 MB/s. Moreover, even for a very low-level process such as edge detection, hundreds or even thousands of elementary operations per pixel are needed [7]. However, many computer vision applications, such as nearly all surveillance systems, require real-time performance, which means that the systems must interact with their environments under response-time constraints. Improving efficiency of the algorithms helps to meet these constraints.

The second issue is reusability. Hardware designers have developed various dedicated computer vision processing platforms [7, 9] to overcome the problem of intensive computation. However, these solutions have created another problem: heterogeneous hardware platforms have made it time-consuming and difficult (sometimes even impossible) for software developers to port their applications from one hardware platform to another.

The third issue is scalability. Recently, multi-camera systems have generated growing interest, especially because systems relying on a single video camera tend to restrict visual coverage. Moreover, significant decreases in camera prices have made multi-camera systems possible in practical applications. Thus, we need to provide mechanisms to maintain correspondence among separate but related video streams at the architectural level.

The open vision library (OpenVL) and its utility toolkits (VLUT) are designed to address efficiency, reusability and scalability to facilitate progress in computer vision. OpenVL, discussed in Section 10.3, provides an abstraction layer for applications developers to specify the image processing they want performed rather than how they want it performed. VLUT, discussed in Section 10.3.7, is created as a middleware layer to separate camera details, events management, and operating details from the specification of the image processing. By providing a hardware development middleware that supports different hardware architectures for acceleration, OpenVL allows code reuse without compromising performance. The novel software architecture separates the specification of algorithmic details from the underlying implementation, allowing for different components to be implemented on an embedded system without recompiling code. Further, when the embedded system’s functionality changes, it is possible to change the drivers without changing