OpenVX Integration Into the Visual Development Environment

Alexey Syschikov, Saint Petersburg State University of Aerospace Instrumentation, Russia
Boris Sedov, Saint Petersburg State University of Aerospace Instrumentation, Russia
Konstantin Nedovodeev, Saint Petersburg State University of Aerospace Instrumentation, Russia
Vera Ivanova, Saint Petersburg State University of Aerospace Instrumentation, Russia

ABSTRACT

The OpenVX standard has appeared as an answer from the computer vision community to the challenge of accelerating vision applications on embedded heterogeneous platforms. It is designed to leverage the computer vision hardware potential with functional and performance portability. As long as VIPE has a powerful model of computation, it can incorporate various other models. This allows to extend facilities of a language or framework that is based on the model to be incorporated with visual programming support and provide access to the existing performance analysis and deployment tools. The authors present OpenVX integration into the VIPE IDE. VIPE addresses the need to design OpenVX graphs in a natural visual form with automatic generation of a full-fledged program, shielding a programmer from writing a bunch of boilerplate code. To the best of the authors’ knowledge, this is the first use of a graphical notation for OpenVX programming. Using VIPE to develop OpenVX programs also enables the performance analysis tools.

KEYWORDS
AGP-Model, Computer Vision, Model of Computation, OpenVX, Performance Analysis, Visual Programming

1. INTRODUCTION

Development of parallel programs, which should be efficiently executed on heterogeneous manycore platforms, is a hard challenge for embedded system developers. Such platforms are targeted to the domains like ADAS, cryptography, video surveillance, aerospace etc. Even today, there are many heterogeneous manycore platforms on the market from NVIDIA (NVIDIA, 2014), Qualcomm (Cheng, 2017), Imagination (Voica, 2016), AllWinner (Allwinner Technologies, n.d.), Samsung (Samsung, n.d.), Mediatek (MediaTek, n.d.) and other vendors. Tomorrow most of embedded systems will be heterogeneous (Joshi, 2016).

Nowadays develop teams create complex computing embedded systems (Ghosh, 2011; Olenev, et al., 2011; Evans, 2004; Balandin & Gillet, 2010). Teams often include many experts from various domains. For an efficient problem solving such teams desperately need a common language for their project. According to the researches, a visual graph notation is a natural representation of an operations sequence (Mellor, Balcer, & Jacoboson, 2002). Each member of a developer team explicitly

DOI: 10.4018/IJERTCS.2018010102

Copyright © 2018, IGI Global. Copying or distributing in print or electronic forms without written permission of IGI Global is prohibited.
or implicitly uses some kind of a graphical flow chart for his projects. It is better to have a single “big picture” of a whole project, to which all the members have a simultaneous access.

In addition, developers face extremely complex and contradictory requirements; for example, they need to produce a high-quality embedded solution for some task within a tight time frame. Meanwhile, the volume of code vastly increases. When companies describe existing situation they compare it to the shift from writing programs solely in an assembly language to writing them on a high-level language. It is more comprehensible and productive and let teams cope with large projects. Modern projects are so huge and sophisticated that the high-level text-based language fall into state of an assembler. It is not a coincidence that source-to-source compilers are used there (Castillo, 2008; Puschel, 2005; Ayguadé, 2009).

Computer vision experts, involved in many of the aforementioned application domains, most frequently require performance for their tasks, so effective use of platform resources is crucial for the success. Responding to the industry demand, Khronos Group developed the OpenVX standard. OpenVX (Khronos Vision Working Group, 2017) is a low-level programming framework for efficient access to computer vision hardware acceleration with both functional and performance portability. OpenVX was designed for diverse hardware platforms, providing a computer vision framework that efficiently addresses current and future hardware architectures with minimal impact on applications.

VIPE IDE (Syschikov, Sheynin, Sedov, & Ivanova, 2014) allows designing parallel algorithms in a visual graph notation and transforming them to parallel programs, which are ready for porting to the required hardware multicore heterogeneous platform.

Our task was to make it possible for developers of computer vision applications to gain all the benefits provided both by OpenVX and by visual programming approach (Syschikov, Sedov, Nedovodeev, & Pakharev, 2017). We used the VIPE IDE as a platform for visual OpenVX. We successfully added support of OpenVX (spec. 1.0.1) in a VIPE IDE. To the best of our knowledge, this is the first visual development interface for the OpenVX programming.

2. STATE OF THE ART

OpenVX is intended to increase performance and reduce power consumption of machine vision applications. It is focused on embedded systems with real-time use cases such as face, body and gesture tracking, video surveillance, advanced driver assistance systems (ADAS), object and scene reconstruction, augmented reality, visual inspection etc.

The using of OpenVX standard functions is a way to ensure functional portability of the developed software to all hardware platforms that support OpenVX.

Since the OpenVX API is based on opaque data types, client-code need not be recompiled, when used with various OpenVX implementations. That is because such machine-specific details as memory alignment, byte packing etc. are hidden inside the vendor library, which is linked to the main program.

OpenVX uses a graph-based execution model (Khronos Vision Working Group, 2017) and incorporates data and task-level parallelism. This model allows OpenVX to solve a number of issues relevant to programs parallelization.

The graph program representation (Figure 1) is crucial to OpenVX efficiency. Developers describe a graph of image processing operations using nodes (functions). Graph nodes could target any hardware computational unit. The graph model enables OpenVX implementations to optimize for power and performance. The host processor can set up a graph, which later can be executed almost autonomously (Syschikov, Sheynin, Sedov, & Ivanova, 2014). Several nodes may be fused by the implementation to eliminate memory transfers (Rainey, et al., 2014). Image processing can be tiled to fit data into a local scratchpad memory or cache. Host interaction during frame-rate graph execution can be minimized.

The use of a visual graph notation for a software development gradually broadens. The following tool suits already use visual graph notation for the computer vision domain:
Toward Green Evolution of Cellular Networks by High Order Sectorisation and Small Cell Densification
www.igi-global.com/chapter/toward-green-evolution-of-cellular-networks-by-high-order-sectorisation-and-small-cell-densification/172194?camid=4v1a

A Mobility Model for Crowd Sensing Simulation
www.igi-global.com/article/a-mobility-model-for-crowd-sensing-simulation/169873?camid=4v1a