BENCHMARKS FOR SMARTCAM DEVELOPMENT

[1,2]W. Caarls, [2]P.P. Jonker, and [3]H. Corporaal
[1]wcaarls@ph.tn.tudelft.nl
[2]Department of Imaging Science and Technology, Delft University of Technology
[3]Department of Electrical Engineering, Eindhoven University of Technology

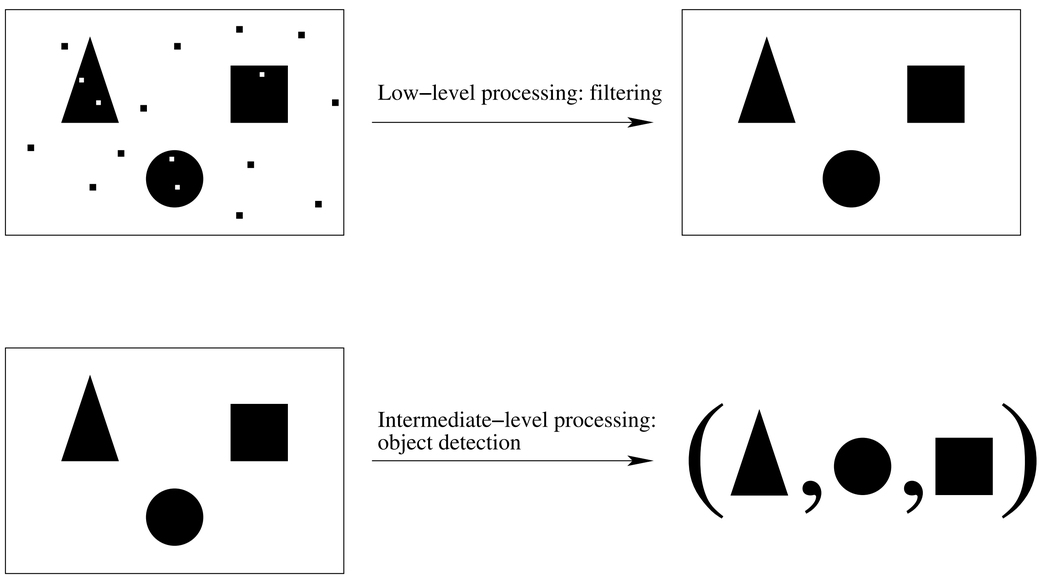
Smart cameras

The continuing miniaturization of microchips makes it possible to integrate increasing amounts of processing power with sensing hardware to create *intelligent sensors*. A typical example of this integration, Smart Cameras are surveillance-camera sized devices with on-board image processing logic. This allows them to be used in stand-alone applications such as robotics, industrial inspection, and security systems. The added intelligence enables functions like positioning, fault detection and face recognition.

The Inca+ Smart Camera by Philips CFT and Philips Research uses an SIMD processor for low-level filtering operations, and a VLIW processor for object processing.

Quantified design

We want to quantify the design of smart cameras by creating a general smart camera architecture template consisting of a sensor and various types of processors, and by instantiating this template for specific applications. The former requires representative benchmarks to guide and test the design of the template, while for the latter we need real-world applications to verify the mapping of the application to and the instantiation of the template.



The three levels of image processing.

Real-world applications

Because real-world applications are rarely as well defined as a set of kernels, we are using actual application code to verify the automatic instantiation of the template (using Design Space Exploration), and the mapping of the user program onto the various processors in the instantiation. Because the automatic parallelization of legacy code – necessary if the application is to run on more than one processor – is an unsolved problem, we rely on compiler directives and code restructuring based on *algorithmic skeletons* to provide information about dependencies and data distribution.

Representative kernels

Image processing applications are usually divided into three stages: low-level, intermediate level, and high-level. Low-level algorithms transform images into other images using filters, geometric transformations, segmentation, etc. The intermediate level extracts objects and features from the transformed images, and the high level stage uses these features to make decisions or present the information. Because each level has its own processing and communication needs, we use representative kernels from all three to guide our algorithmic template.

Skin-tone detection on a smart camera, part of a larger face recognition system: original, skin-colored pixels, and final skin-tone regions. These regions are fed into a neural net classifier running on the same camera [1].

[1] H. Fatemi, R. Kleihorst, H. Corporaal, and P.P. Jonker, Real-Time Face Recognition on a Smart Camera, in: Proceedings of Acivs 2003 (Advanced Concepts for Intelligent Vision Systems) (Ghent, Sept. 2-5), Ghent University, Ghent, 2003