Challenges to Embedding Computer Vision

J. Scott Gardner
General Manager and Editor-in-Chief
Embedded Vision Alliance (www.embedded-vision.com)

May 16, 2011

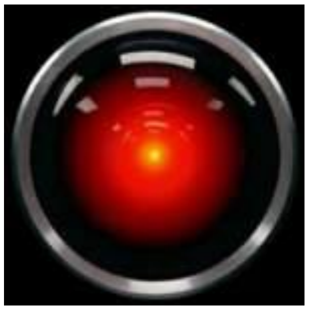

Figure 1 HAL 9000 – a machine that sees. Source: Wiki Commons

For many of us, the idea of computer vision was first imagined as the unblinking red lens through which a computer named HAL spied on the world around itself in 2001: A Space Odyssey (Arthur C. Clark and Stanley Kubrick, 1968). The computers and robots in science fiction are endowed with vision and computing capabilities that often exceed those of mere humans. Researchers in computer vision understand the truth: computer vision is very difficult, even when implemented on high-performance computer systems. Arthur C. Clarke underestimated these challenges when he wrote that the HAL 9000 became operational on January 12, 1997. As just one example, HAL was able to read the lips of the astronauts, something not yet possible with modern, high-performance computers. While the HAL 9000 was comprised of hundreds of circuit boards, this article takes the computer vision challenges to the world of embedded systems. The fictional Dr. Chandra would have been severely constrained if trying to design HAL into a modern embedded system. Yet, embedded vision systems are being built today, and these real-world successors of HAL meet the unique design constraints that are critical for creating successful embedded products.

Why is Computer Vision so Difficult?

Computer vision has been described as "inverse 3D graphics", but vision is orders of magnitude more difficult than rendering a single 2D representation of a 3D scene database viewed through a virtual camera. In 3D graphics, the computer already knows about the objects in the environment and implements a "feed-forward" computation to render a camera view. In computer vision, an observed scene must be analyzed to build the equivalent of a scene database which must correctly identify the characteristics of the objects in the scene. An embedded computer needs to enhance the image and then make inferences to identify the objects and correctly interpret the scene.

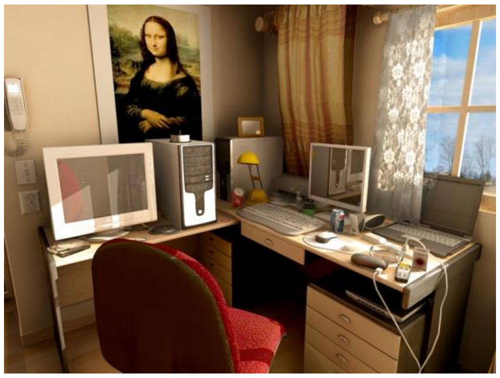

Figure 2 Rendered 3D Scene with occluded objects. Source: Wiki Commons

However, vision systems don't have perfect information about a scene, since the camera field of view will often include occluded objects. It would be very difficult to computationally recreate a 3D scene database with the same detail as the source data for a 3D graphics renderer.

To simplify the problem, a vision system usually works with assumptions about the type of objects in a scene, and complex algorithms handle the cases where moving objects become occluded and perhaps disappear from the frame. While real-time, pixel-level processing of streaming image data is obviously compute-intensive, the complexities caused by occlusion (or other depth of field issues) require high-level processing of data that may span several frames. To make the task even more difficult, the high-level processing often must identify and track a moving object of interest in a scene that has background motion, such as trees blowing in the wind.

Embedded Systems Constrain the Compute Resources

Decades of research in computer vision algorithms have finally advanced computer vision technology to the point where it can solve real-world problems. However, researchers developed these algorithms on high-performance computer systems with fast, general-purpose CPUs and copious amounts of memory. Many of these algorithms are not practical to implement on an embedded system, since embedded system design is all about designing with constraints. Embedded systems deal with a large number of technical design constraints that often vary for each project. Examples of technical constraints include cost, power consumption, size, weight, acoustic noise, etc. Moreover, sometimes a project is constrained by issues that aren't in a data sheet. Examples of these design constraints include design time, testing cost, software development environment, availability of libraries, design reuse issues, etc.

Constraints Lead to a Host of Unique Design Issues

To implement computer vision in an embedded system, designers need to identify the constraints that will have a dramatic impact on algorithm selection and hardware requirements to meet the performance goals. Experienced embedded system designers already understand these design constraint issues, but many may not have experience with modern computer vision algorithms. Computer vision experts who have only developed software on general-purpose computers will need to adapt to the constrained world of embedded computing. Most of this discussion assumes the reader is new to embedded system design. The following brief overview covers some of the issues to be considered when developing a computer vision solution for an embedded system.

1. *Specialized computing hardware*

Figure 3 Pentium 3 CPU. Source: Wiki Commons.

Few embedded systems have a power budget for implementing the entire computer vision workload on a general-purpose processor. While very flexible, a purely general-purpose software approach isn't the most energy-efficient solution, and many computer vision applications would require a high-end, workstation-class processor to run these workloads. The raw computing hardware in a high-end CPU requires the support of millions of transistors that deal with reading instructions and moving data to support a broad range of workloads. A system with specially-built, dedicated hardware can consume less than 100th of the power when compared to a general-purpose CPU performing the same math. However, the general-purpose CPU is easy to program and can efficiently process the high-level operations in the computer vision pipeline. For pixel-level processing, specialized programmable devices efficiently process streaming data and algorithms with large amounts of parallelism. Examples include DSPs, GPUs, and FPGAs. Most of these devices also integrate a general-purpose embedded CPU. This is only a high-level snapshot of a topic that is covered in detail by the Embedded Vision Alliance website.

2. *Smaller memory footprint*

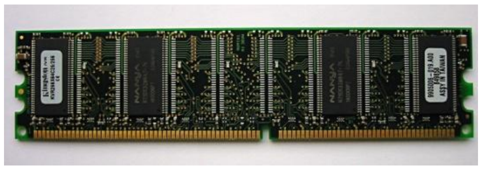

Figure 4 DDR Memory. Source: Wiki Commons.

As memory chip densities continue to increase, it would be easy to discount memory size limitations as a long-term problem for embedded vision systems.
However, the memory constraint isn't going away, since adding memory always incurs a cost – to power, weight, size, level of integration, reliability, etc. In many embedded vision systems, the memory system isn't a single bank of DRAM, easily upgraded by dropping in another memory stick. There are memory storage elements at various places in the vision pipeline, each supporting a different role in the processing workflow and incurring a different cost for adding more memory. Often these memory storage elements are integrated with other computing resources on a single chip, so there is a trade-off in silicon area as memory space competes for real estate with other integrated devices. There are also additional complexities related to memory latency when adding extra devices. These memory issues are difficult to spot when developing algorithms on a workstation, but they become major issues when developing an embedded vision system.

3. *Bandwidth issues*

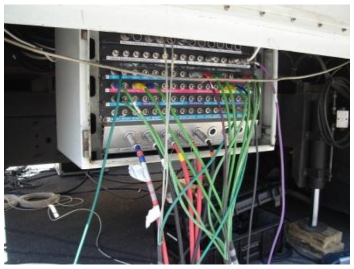

Figure 5 Video panel. Source: Wiki Commons.

One of the high-level design issues in computer vision relates to the age-old question of distributed versus centralized computing models. In many vision systems, multiple imaging sensors are remotely collecting real-time video data for analysis. The question quickly surfaces of whether to process the data near the camera or to ship the data to a centralized computer for processing. Once again, the best choice will depend on the constraints of the system. How much bandwidth is available to get multiple camera feeds back to the central processing unit? Is this the most cost-effective approach if a small amount of processing at the camera could reduce the data rate substantially?

4. Security issues

Figure 6 Predator drone. Source: Wiki Commons.

The question of security for an embedded system relates to the issue of distributed computing, since remote imaging devices require detection of tampering and perhaps even need secure communication of high-speed video data. Adding security features incurs a cost that may not fit within the system constraints. An interesting example of this issue was the discovery that the US military did not encrypt video data from some of its early drones.

5. New algorithms and optimizations for embedded systems

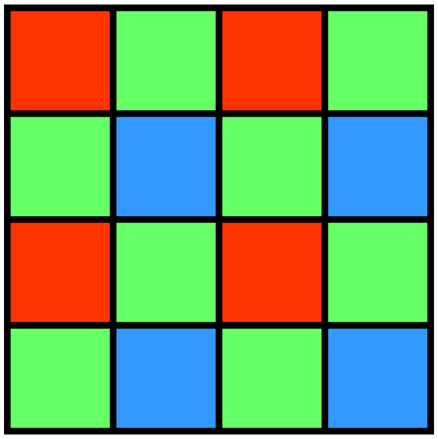

Figure 7 5x5 Spatial Filter. Source: Wiki Commons.

The focus on design constraints leads to a question about whether classical computer vision algorithms, developed on workstations or even mainframes, represent the best solution for an embedded vision system. On the other hand, some of the early computers had similar constraints. The circular queue is an example of a processing technique for spatial filtering in a system that is memory-constrained. This technique dropped out of fashion when computer systems had plenty of memory to store an entire image. A spatial filter traverses across an image while only accessing data from a number of image rows that is equal to the filter size. For example, a 5x5 filter window only accesses 5 rows. After the spatial filter processes all 5 rows, the filter loads the next row and removes the top row (changes the circular pointer to load the new row in the oldest row memory). In this fashion, only the data being processed needs to occupy scarce memory. Any of these techniques add complexity for the software designer, but that is the constant trade-off for embedded system designers. Dealing with hardware constraints is just part of life for the software team working on embedded computers.

The EVA Community Addresses these Design Issues

Science fiction has once again become a reality, and our machines have started to see and understand the world around them. This article started with a look at computer vision as it was envisioned in 1968 to bring HAL to life with the ability to see and communicate. It has been over 40 years since Clarke and Kubrick anthropomorphized a machine by giving it these features. While most of the world believes these innovations are just like every other technology advance that catches up to science fiction, hopefully this article helps highlight the huge amount of effort behind these embedded vision systems.

The good news is that embedded vision is incredibly fascinating, and the visual aspect of these embedded products gives a tangible sense of accomplishment when you see it working. Whether you're an experienced embedded system designer wanting to learn about computer vision or a computer vision expert wanting to design for embedded systems, welcome to the world of embedded vision. Please join the conversation at www.embedded-vision.com and share your expertise with others in this rapidly-growing field.