

# 3D Vision: Developing an Embedded Stereo Vision System

By John Iselin Woodfill, Ron Buck, Dave Jurasek, Gaile Gordon and Terrance Brown, TYZX

## ***Stereo vision systems can provide accurate real-time data in a variety of applications***

Many products would benefit greatly from the ability to see people and objects and respond to events in real time. Cars, for example, could detect pedestrians or objects in harm's way and brake automatically. Agricultural equipment could navigate fields autonomously, avoiding stray objects, people, and wildlife. Security systems could track people moving through a building.

Any vision system deployed in the real world must be able to operate in a broad range of conditions. Lighting can vary dramatically. Rain, snow, and ice can occlude objects or alter their appearance. People and objects can be stationary or moving, and people can move in different ways and at different speeds. Detecting a darkly clad pedestrian racing across an unlit alley in a thunderstorm requires visual sophistication far beyond the simple identification of stationary objects in a well-lit laboratory.

## **STEREO VISION**

The most promising solution for promptly and effectively interpreting visual data in these challenging conditions is 3D vision, and the most effective way of realizing 3D vision is stereo vision. Using two cameras side by side, stereo vision produces a virtually instantaneous estimate of the distances to elements in a scene. Detecting distances serves as a primary cue for detecting things that stand out in depth from a background. In addition to quickly gauging depth, stereo vision is highly effective for segmenting objects and gauging their size and shape. Ultimately, stereo vision simplifies data interpretation.

Accurate, real-time responsiveness would be difficult, if not impossible, to achieve cost-effectively using 2D systems, which rely on a single imager. Besides being more error-prone at gauging distances, 2D-vision systems can be easily confounded by changing lighting conditions. Now that low-cost complementary metal-oxide semiconductor (CMOS) imagers have come on the market, the opportunity to develop a general-purpose 3D embedded-vision system has never been better.

The DeepSea G2 Stereo Vision System that TYZX developed features an embedded stereo camera consisting of two CMOS imagers, a TYZX DeepSea 2 stereo application-specific integrated circuit

**Copyright © 2007 IEEE. Reprinted from IEEE Computer, May 2007.**

***This material is posted here with permission of the IEEE. Such permission of the IEEE does not in any way imply IEEE endorsement of any TYZX products or services. Internal or personal use of this material is permitted; however, permission to reprint/republish this material for advertising or promotional purposes or for creating new collective works for resale or redistribution must be obtained from the IEEE by writing to [pubs-permissions@ieee.org](mailto:pubs-permissions@ieee.org).***

*By choosing to view this document, you agree to all provisions of the copyright laws protecting it.*

(ASIC), a field-programmable gate array (FPGA), a DSP/coprocessor, a PowerPC running Linux, and an Ethernet connection. About the size of a hardback book, the G2 delivers real-time—30 frames per second (fps)—interpretations of visual data even in challenging environments, and has been deployed in a variety of applications, including people-tracking and autonomous-vehicle navigation.

### EFFICIENT PROCESSING

The richer and more immediately useful results that 3D vision systems produce come with a price. Processing data from stereo-imager inputs increases the system's computational requirements—and potentially its cost. Our biggest design challenge was to address the computational needs of stereo vision in a platform that would remain affordable, compact, and capable of running for long periods on little power.

One way to minimize the embedded computational requirements would be to collect image data in the vision system, then offload the data processing to another system or series of systems on a local network. We quickly realized, however, that the bandwidth requirements for multicamera applications would make this approach untenable.

Consider, for example, a wide-area tracking system based on stereo imagers (one color and one monochrome), producing 26 bits (10 bits Y, 8 bits U and V) and 10 bits per pixel respectively. For  $640 \times 480 \times 30$  fps, the imagers will output ~330 Mbits of data per second. If we feed this data into a stereo processor and down-sample the color image, we might end up with as little as 200 Mbits of data per second. By processing the range and color data to detect and locate people, we can further reduce the information to a few bytes per person tracked per frame, resulting in perhaps only 80 Kbits per second. For just one tracking camera, a conventional LAN could accommodate the resulting data at any stage in the pipeline.

However, for 100 tracking cameras, the left and right source data becomes 33 Gbits per second, the color and range 20 Gbits, and the segmented track data might be about 8 Mbits per second—in total, an amount of data far beyond the capacity of most networks and sufficiently large to call into question the system's real-time capabilities.

Image processing, therefore, is most practically performed within the system itself. A self-contained system is also the most practical for applications such as automotive safety and consumer appliances, where offloading to another processor would be impractical.

### FUNCTIONAL DECOMPOSITION

Instead of offloading the processing, we solved the computational problem by applying functional decomposition, breaking down the computational work into smaller tasks, and then putting as much processing as possible into hardware primitives.

We decomposed the data-processing work into these steps:

- Rectification—aligning image data to account for imperfections and misalignments in the imagers
- Stereo correlation—correlating data from both imagers to create one master set of data
- Background modeling—applying a background model and differentiating foreground images from background images
- Mapping data into a 3D quantized projection space
- Completing other general-purpose processing tasks

At the simplest level, the computation required for a vision system takes input from left and right imagers, processes data, and outputs results for use by an application.

### First-Generation Product

In our first-generation embedded product, we programmed an FPGA to perform image rectification and developed an ASIC, the DeepSea 2 chip, to perform stereo correlation. The remaining processing was performed on a 600-megahertz Pentium III chip, which also ran the user application. The FPGA, ASIC and PIII communicated over a peripheral component interconnect (PCI) bus. The system ran at 15 hertz and processed images that were 400 pixels wide by 300 pixels high.

For applications in which a vision system must make rapid decisions about complex, changing data, it's best to have many frames of data to confirm the system's interpretation of a scene. While our first-generation product represented a milestone in visual-system integration, its 15 fps performance wasn't sufficiently fast for some real-time applications, such as automotive safety. The system also lacked the computing power required for tracking objects—a common requirement in many vision applications.

In addition to increasing the system's performance, we also wanted to lower its overall processing and power requirements to make it easier to deploy where space is limited.

### Second-Generation Product

After considering various tradeoffs, we invested in more integration up-front for our second-generation product, the DeepSea G2. By applying additional functional decomposition in the G2, we increased the system's performance while decreasing CPU and power requirements.

The G2 passes even more work to hardware primitives in the FPGA and ASIC. For example, an FPGA now performs background-modeling primitives previously performed on a PIII. This tighter integration and hardware processing enabled the system to achieve process images at 30 fps in a smaller package.

**Table 1** compares the functional decomposition of the two products. Whereas the first-generation product took advantage of an off-the-shelf CPU, the second-generation product relies on an FPGA for background modeling and individuation.

**Table 1: Comparing Functional Decomposition in Two TYZX Products**

Task	First-generation product	DeepSea G2 Stereo Vision
Rectification	FPGA	FPGA
Stereo correlation	DeepSea 2 ASIC	DeepSea 2 ASIC
Background model	Pentium III	FPGA
Projection space	Pentium III	FPGA
DSP coprocessing	NA	BlackFin DSP coprocessor
User application	Pentium III	PowerPC

We chose hardware-accelerated visual primitives that would be useful for many tasks, account for a large part of the computational load, and be suitable for hardware acceleration. In the G2, some primitives process the incoming streams of left and right images. Other primitives process the output of other primitives.

Since the G2 is an embedded stereo camera, the visual primitives are aimed at producing and interpreting range data. Since stereo correlation is computationally intensive and a common subcomponent of vision algorithms, it's an obvious visual primitive to support. Background modeling based upon range and color requires huge amounts of memory bandwidth and finds a place in many tracking algorithms—hence it's our second visual primitive. A third primitive, ProjectionSpace, produces 2D and 3D quantized representations or projections of the range data. This is a computationally intensive, but widely applicable, visual primitive. Lastly, a programmable DSP is included in the G2 as a generalized visual primitive—an additional resource for doing expensive, regular image operations.

**Stereo correlation.** The primary visual primitive is the stereo-correlation processor. It takes as input the left and right images, and creates a range image as output. The TYZX DeepSea 2 ASIC accelerates the performance of this primitive.

**Background modeling.** The background-modeling primitive takes as input range and intensity data and generates a pixel-by-pixel foreground/background map of the image based on the nature of the pixel's change from previous frames. This task may require roughly 400 bits per pixel to be read from memory, and roughly 200 bits per pixel be written to memory. For 400 x 300 images at 30 frames per second, background modeling requires over 2 Gbits of memory bandwidth per second.

**Matching and updating each pixel involves several comparisons and updates.** Offloading this task to dedicated hardware reduces the CPU's workload and is a key factor enabling the use of a smaller embedded CPU. The output is a binary foreground mask image.

**ProjectionSpace.** The ProjectionSpace primitive transforms a full 3D cloud of points into either 3D data projected into a Euclidean 3D quantized volume or a 2D array. These 3D volumes or 2D arrays are useful for many applications, such as people tracking. The ProjectionSpace primitives transform vast amounts of 3D data into forms that make sense for applications tracking people and objects in a 3D space.

## OTHER DESIGN CHALLENGES

The technical hurdles for such a vision system also include size, portability, and power requirements. The design goal, in most cases, is to unobtrusively add a vision system to an existing product or environment. That means deploying a solution that's compact and probably lightweight, and makes few power and cooling demands.

**Table 2. TYZX DeepSea 2 Specifications.**

TYZX DeepSea 2 Stereo-Correlation Chip	
Specification	Unit of measure
Input image size (max)	512 × 2048 (10 bit) pixels
Stereo range	
Search window	52 disparities
Subpixel localization	5 bits
Z output	16 bits
Max frame rate	200 fps (512 × 480)
Power	< 1 watt
Pixel disparities/second	2.6 billion

**Table 2** details the G2's specifications. The system consumes about 15 watts; its camera includes a 100BaseT Ethernet interface that accepts Power-over-Ethernet. A compact flash memory card stores a Linux kernel and a root file system, so that the G2 can boot into Linux on power-up.

Increased hardware integration and acceleration in the G2 has paid off. The system can track people and objects in changing lighting conditions, using 512 × 380 images at 30 fps—a rate fast enough to support rapid decision-making and deliver real-time responsiveness. Electroland, a company that creates interactive environments, has installed four G2s running our person-tracking application in their interactive installation for Target in Rockefeller Center's new observation deck. Another partner is developing an autonomous urban-reconnaissance vehicle that uses the G2 to run obstacle-detection and path-planning algorithms.

In the future, we expect to integrate even more functionality in hardware to create embedded stereo systems that are smaller, faster, and easier to integrate in products working in the real world.

**Ron Buck** is president and CEO of TYZX.

**John Iselin Woodfill** is chief technology officer of TYZX.

**Gaile Gordon** is vice president, advanced development, at TYZX.

**Dave Jurasek** is vice president, hardware engineering, at TYZX.

**Terrance Brown** is senior hardware design engineer at TYZX.

Contact the authors at [info@TYZX.com](mailto:info@TYZX.com).

Editor: Wayne Wolf, Dept. of Electrical Engineering, Princeton University, Princeton, NJ;  
[wolf@princeton.edu](mailto:wolf@princeton.edu)