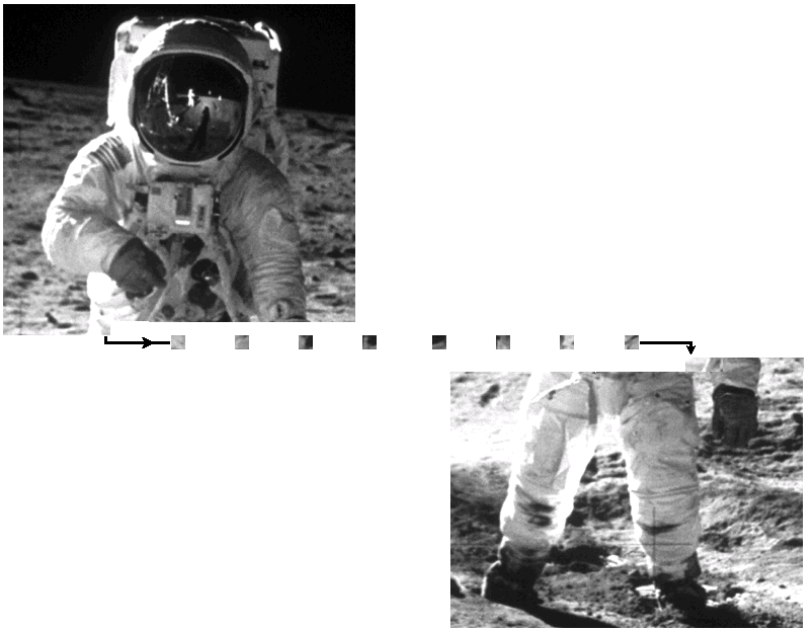
ImageFlow Technical Description

Pipeline Image Processing with ImageFlow



Datacube, Inc.
300 Rosewood Drive
Danvers, MA 01923
TEL:978-777-4200
FAX: 978-777-3117

Document No. MS0001-2.1
November, 1998

Notice

This document is provided “as is,” for informational purposes only, and without warranties as to the validity or usefulness of the information herein. This document is provided without any expressed or implied warranties. Because of the diversity of conditions under which this information could be used, no warranty of fitness for a particular purpose is offered.

A request for and/or receipt of this document is assumed by Datacube, Inc. to be an acceptance of the terms and conditions stated herein.

Datacube, ImageFlow, MAXbus, and MaxVideo are trademarks of Datacube, Inc.

Datacube, Inc.
300 Rosewood Drive
Danvers, MA 01923

TEL: 978-777-4200
FAX: 978-777-3117

info@datacube.com
http:\\www.datacube.com

Inside. . .

Introduction

At one time, image processing was too complex and too expensive to be supported by any but the most sophisticated computer systems. Its roots are in expensive research & development projects like those supported by the U.S. space program and in defense industries including military surveillance and guidance systems. Just a few decades ago, these high performance imaging applications necessitated the computational capabilities and speed of the world's most powerful supercomputers. But the growth of the computer industry over the last few decades made it possible for technologies like image processing to become both more widespread and affordable.

Today, image processing is finding its way into an ever increasing variety of applications – from assembly line inspection systems in auto parts plants, to motion detection in surveillance and security systems, to computer-aided animation in the entertainment industry, and more. The technology successfully made the migration from supercomputer to personal workstation. But with more widespread use come more demands on the technology – demands for larger bandwidths with faster processing speeds, and more flexibility yet tighter control.

To meet the needs of its customers in the R&D and defense communities, Datacube adopted a computer architecture known as "pipeline processing" in the early 1980's. Capable of providing exceptionally fast, highly precise manipulation of large images, this complex technology enabled the automation of a wide range of inspection and visualization tasks. In 1990, Datacube introduced the first of its single-board image processing sub-systems and ImageFlow, a library of C-callable functions for the configuration and control of multiple, parallel pipelines.

As a broader audience of application developers come to understand the features and benefits of Datacube's pipeline image processing, the technology is spreading into a growing number of manufacturing, medical, and other private-sector applications. This document provides a conceptual overview of a pipeline processing computer architecture and explains the features and benefits of ImageFlow that make it and exceptional development tool for high performance image processing applications.

The Evolution of an Image Processing Architecture

Most image processing systems use general-purpose architectures composed of a central processing unit (CPU), memory, and a single data bus. The central processor fetches data from memory, processes it, and stores it in a buffer, from which it can either be fetched for a second (or third or fourth or…) process, or output in one form or another (see *Figure 1*).

These CPU-based computer architectures have been around for a number of years, developing a strong market base and a broad variety of application development tools. They're inexpensive due in large part to high market demand, and developers find them reliable and relatively easy to understand.

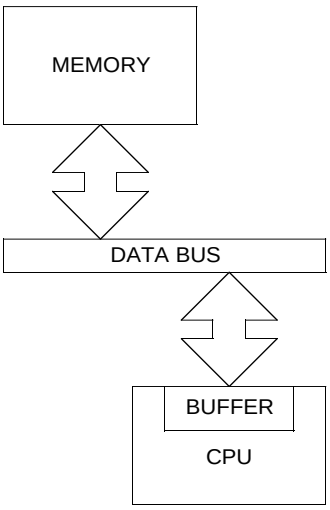


Figure1: CPU-based Architecture

The CPU-centered processing model is limited, however, in how quickly it can perform complex processes on large amounts of data. Also referred to as “Single Instruction, Single Data” (SISD), this type of architecture is capable of completing only one computation per cycle or clock tick. The more computations or operations required by a given algorithm, the greater the delay between the input of data and the output of a result. The problem is compounded by the volume of data required by applications like image processing.

In an attempt to decrease this bottleneck, some systems offer variations of the SISD architecture. For example, the Single Instruction, Multiple Data (SIMD) architecture includes multiple processors under the control of one central or master processing unit. Data is broken up and sent to separate processors which, acting under the direction of a single control unit, each perform the same operation at the same time on a subset of data. SIMD architectures offer improved data throughput over SISD systems, but are still limited in the number or complexity of operations they are capable of performing.

Multiple Instruction, Multiple Data (MIMD) architectures allow multiple computations to be simultaneously executed on subsets of input data. They subdivide processing power across a complex network of interconnected nodes. Input data is segmented and directed to each of the various nodes, boosting overall data throughput. Each node has the authority to execute a separate instruction on its own subset of data, increasing the number of computations that can be executed at a time. While highly functioning, the drawbacks of such an architecture are obvious: there is a tremendous amount of overhead associated with segmenting the data and synchronizing operations.

Pipeline processing has proven to be more effective than any of these architectures handling high volume, high performance image processing applications and other data intensive operations that must be executed in real time. It provides distinct advantages of speed, data throughput, and flexibility over the more primitive CPU-based architectures.

Pipeline Image Processing

Pipeline processing subdivides processing control differently — and more effectively — than MIMD systems. Pipeline processing uses a collection of sequentially connected, specialized computational elements. These elements perform operations upon data as it passes through the pipeline — a process that is fundamentally different from the fetch/process/output cycle employed for each byte handled by general purpose CPUs (see *Figure 2*).

Datacube hardware and software applies the pipeline architecture to image processing. A steady stream of image data is input into the system and piped through a series of highly specialized, easily reconfigured computational elements capable of a wide variety of image processing functions. Because all of these elements function simultaneously, data never stops flowing through the system. Output is continuous from the time the first pixel completes its journey to the last.

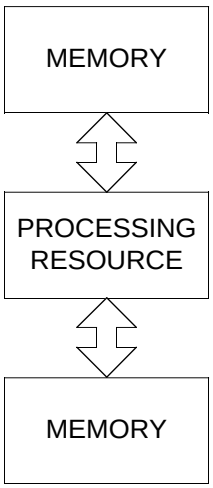


Figure 2: Pipeline Architecture

Speed and Efficiency

Consider again the simple CPU-based processing architectures. CPU-based architectures are capable of issuing only one instruction on one set of data at a time — a fact that severely limits the architecture's efficiency when dealing with complex algorithms. In a highly simplified example, consider what it would take to process an image four pixels high by four pixels wide. If processing was limited to a single instruction, it would take a total of 16 steps or ticks of the clock — 1 instruction * 16 pixels — to process the entire image. If processing required the execution of four separate instructions on each pixel, it would take a total of 64 steps — 4 instructions * 16 pixels. In complex imaging applications, that can lead to a significant delay in the display or availability of important data.

Because different instructions can be executed on different sets of data simultaneously, complex algorithms are handled more quickly and efficiently by pipeline processors. *Figure 3* shows how the 16 pixels in our example image move through a pipeline set up to execute four separate instructions. The pipeline fills, one pixel at a time, during the first four steps or ticks of the clock. By the fifth step, however, the first pixel has been completely processed and is available for display. Each step after that produces another pixel, and with the 20th step, the entire image has passed through the system — in less than one third the number of steps required by the CPU-based architecture. Add to this the fact that all Datacube hardware has been optimized to provide the fastest, most accurate performance possible for its particular image processing task.

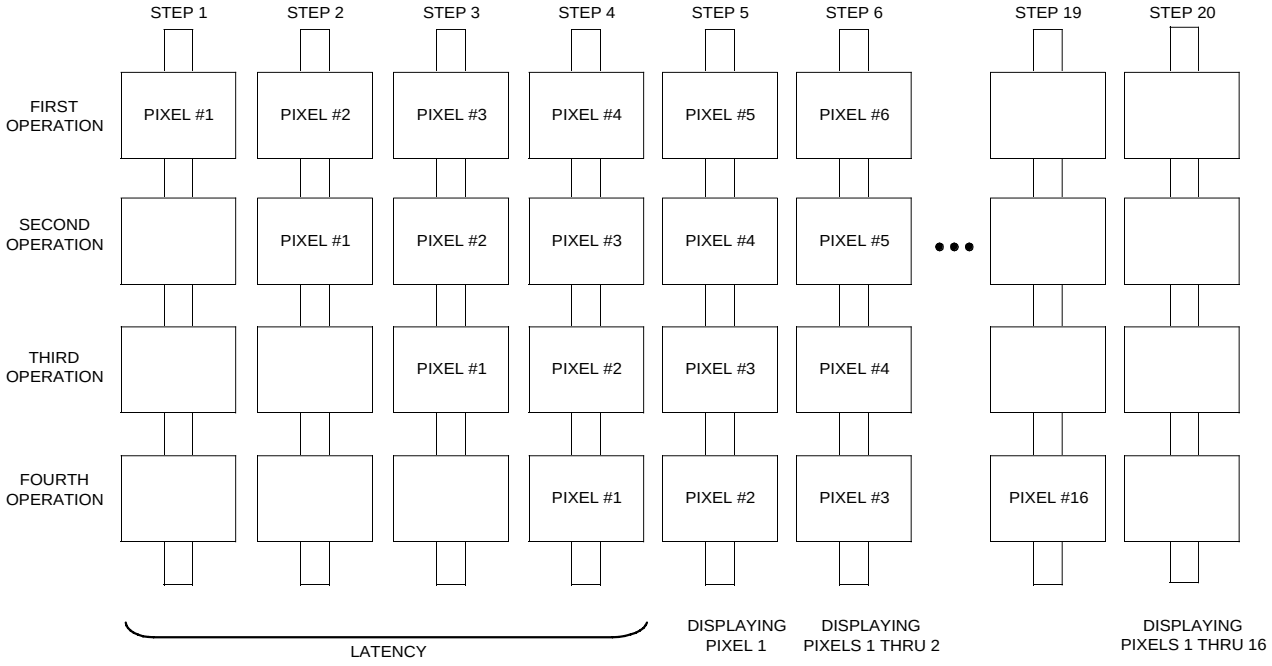


Figure 3: Efficient handling of a complex algorithm by pipeline processing

Flexibility

Though the details behind pipeline processing's flexibility are very complex, the concept is not. As data enters the system, it flows through a series of memory devices and processing resources, much like water flowing through a household plumbing system. The flow is controlled by a series of valve-like gateway elements. Between operations, data is stored in elements called surface stores, which can be likened to holding tanks.

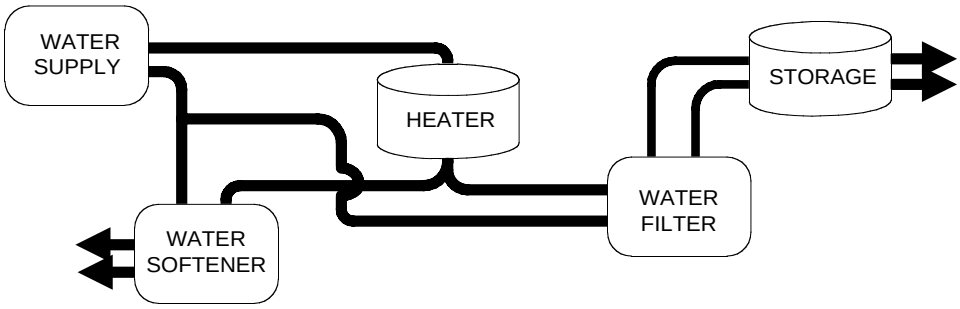


Figure 4: Household Plumbing Analogy

Just as water can be re-directed with the opening or closing of a valve, the elements that work together to form a pipeline can be configured and reconfigured in a seemingly infinite number of combinations. Within each processing device are multiplexers and crosspoint switches which allow data to be sent through a wide variety of paths. The crosspoint switch is able to make and/or break these connections "on the fly," making the pipeline processor a highly flexible computational architecture well-suited to image processing operations requiring high throughput. Configured alone or in parallel with other pipelines, this flexible design provides applications with exactly the resources they need.

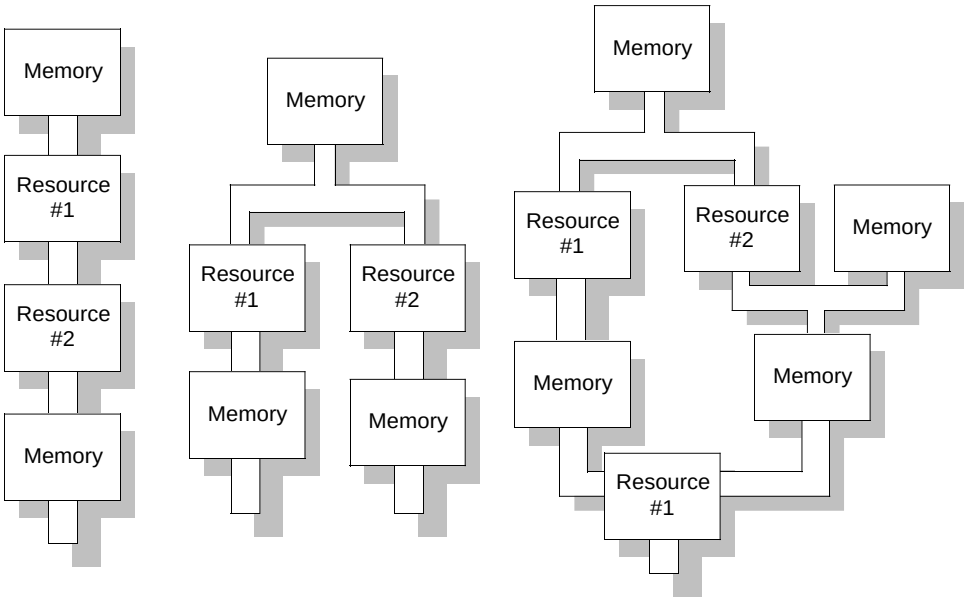


Figure 5: High bandwidth crosspoint switches allow "on the fly" reconfiguration

Building Powerful Pipes

How does an application direct data through such a maze of specialized processors? Like a household plumbing system, pipeline processing sends a flow of something (data, rather than water) through a set of preconfigured "pipes." Both systems need a set of controls to govern the moment-by-moment activation and flow rate of the system. Household plumbing systems have storage tanks, valves, and faucets. What controls do far more complex image processing systems have?

The first generation of Datacube's pipeline processor cards (1986) was programmed at the hardware register level. This was a challenging task, but it was possible because back then, a system typically had only a few hundred programmable registers. Datacube proceeded to develop several custom image processing ASICs to implement its second generation of pipeline processors (1990). The new devices' thousands of hardware registers created a clear need for higher-level programming tools. Datacube developed ImageFlow to support application development on this highly advanced second-generation hardware.

ImageFlow is Datacube's library of C-callable functions used to set up and control the flow of image data through software configurable "pipes." These pipes are created by connecting some of the many hardware based elements on Datacube hardware — acquisition devices, processing elements like multipliers, storage devices, display devices, and more. These devices act like the valves, tanks, and faucets found in a plumbing system, and code written in ImageFlow dictates how they're connected.

While the household plumbing analogy is a useful introduction to pipeline processing topologies, there are significant differences between plumbing and the flow of image data through a pipeline system. First, software gateways must be more intelligent than water faucets, because they interpret the data coming into the system as well as control the flow. They define the format and transfer parameters of the pixel data, such as the height and width of the image, timing, what kind of transfer should be used (continuous stream or one frame), and other attributes of the conversion between a two-dimensional image and a one-dimensional stream of data. ImageFlow allows the developer to specify each of these characteristics and many others, providing complete control over the application.

Another significant difference is that the upstream valve in a plumbing system has no "knowledge" of the flow requirements at the downstream end, but instead lets through all the water it can. On the other hand, the upstream valve or gateway in an ImageFlow-based pipeline processing system is fully aware of the flow requirements demanded up by the downstream gateway (or gateways) and adjusts its data flow accordingly. This prevents a back-up of image data — which could result in the loss of important information.

Architectural Components

ImageFlow recognizes five basic architectural building blocks which can be used to configure an endless number of pipes, working alone or in parallel, to solve a wide variety of image processing problems. These basic building blocks are: surface objects, gateway elements, device objects, pipe objects, and system objects (see *Figure 6*).

The ***system object*** represents the entire image processing system and includes all the pipe objects, device objects, gateway elements, and surface objects used by a particular image processing application. Note that it does not include a CPU — the system object is composed entirely of image processing-specific elements. Exactly which elements are included in a given

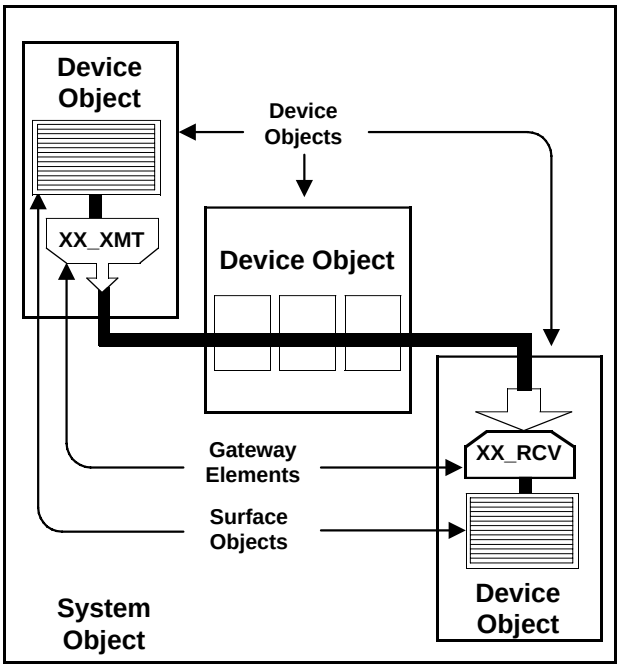


Figure 6: Basic components recognized by ImageFlow

system object is defined in the ImageFlow configuration file.

Device objects are image processing modules designed to handle a particular task which might include acquisition or an arithmetic function. Devices don't have to be on the same board to be part of a single system object — devices may be part of the image processing motherboard (for example, the AU or AM device on a MaxVideo 250), daughtercards attached to the motherboard (Datacube's MaxACQ acquisition modules), or separate boards connected by data bus cables to the main processing board. Each device object is made up of one or more image processing elements (which may include surface objects, gateway elements, or other specialized processing elements).

Surface objects are the two-dimensional "canvases" on which pixel data is laid for temporary storage. They are software defined structures that define what type of information is stored in memory and how.

Gateway elements act like valves or faucets controlling the flow of pixel data into and out of any pipeline. The gateway is the only one of the five basic ImageFlow components that is not an object but an element; it does not have to be explicitly declared and created in an application like the objects.

Pipe objects are one-dimensional pipelines which are put together to form an image processing application. Pipes include the individual image processing elements that actually process pixel data. They frequently pass through more than one device, and can have multiple sources and/or destinations.

Structuring ImageFlow Applications

Think again of the household plumbing analogy, and the many paths water might flow through a typical house. As described previously, gateways are similar to valves, and data pipes are analogous to water pipes. Surfaces provide beginning and ending points for each pipe, so the analogy must be expanded to add water tanks representing surfaces in the plumbing system. Image processing devices are represented by individual rooms in the house, and the system object is represented by the entire house.

Figure 7 depicts an ImageFlow application represented as a house, using the plumbing analogy for all five of the building blocks described thus far. This household plumbing model uses many more water tanks than a conventional system would, because in ImageFlow every pipe must begin and end with a surface (tank). Intermediate buffering tanks/surfaces appear throughout the data path and each one is attached to a valve or a pair of valves. The tanks which have a "V" in them represent surfaces on virtual memory stores, and these have only one valve.

The incoming analog video data is like the water service. The water flows into the house through the water main and is immediately fed into the

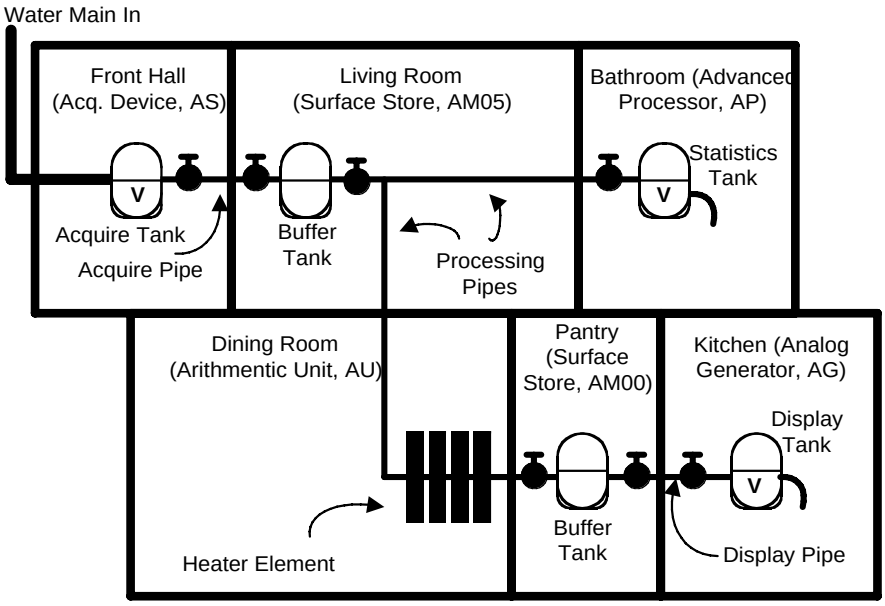


Figure 7: Expanded Household Plumbing Analogy

acquire tank in the front hall, which is analogous to a surface on the Analog Scanner (AS) device that converts analog data to digital pixel data.

The first pipe in the system runs from the acquire tank to a buffer tank which is in the living room. This is how a pipe crosses device boundaries in ImageFlow. This first pipe object is usually called the “acquire pipe.”

The second pipe is a multi-destination pipe that feeds two different tanks in two different rooms. It forks after it leaves the buffer tank, with one fork going to another tank in the bathroom, analogous to the view surface on the Advanced Pipeline Processor (AP) device (a virtual surface used to extract statistics). The pipe's other fork runs into and through the dining room — analogous to the Arithmetic Unit (AU) device — but never enters a tank/surface in that room/device. Instead, it passes through heating elements which raise the temperature of the water, and then crosses a second device boundary and enters the tank/surface in the pantry (analogous to a surface store), ending the second pipe. By going through the dining room, the pipe accesses all the processing elements in that room/device. This whole multi-destination pipe could be called the “processing pipe,” since it handles most of the actual image processing. It outputs statistics at the end of one fork and a processed image at the end of the other fork. An ImageFlow application can have many more processing pipes, configured in serial or parallel.

A third pipe goes from the pantry buffer tank to the display tank in the kitchen, analogous to the DAC surface on an Analog Generator (AG) device, used to convert the digital pixel data back to analog video. In ImageFlow, this pipe is typically called the “display pipe.”

There is one point in this model where the plumbing analogy breaks down: the processing pipe fork in the living room. Water molecules can't be duplicated. Therefore, when the flow of water comes to a fork in the pipe, some of the water molecules flow into one side and the remaining water molecules flow into the other side. In an ImageFlow application, however, pixel data can

be and is duplicated in order to allow all of the image data to flow through both processing pipes.

Image data is also duplicated between devices, because a surface object can not be written to and read from at the same time. A duplicate of the surface object (a software defined structure) is created on the same image memory store (a hardware device that may hold many surface objects). This duplicate surface object is then used as the beginning of the next pipe. A duplicate image does not use extra memory resources for its image data, though it does independently store other attributes such as the surface's processing rectangle and overlay information. These two surface objects must share the same memory store on the same device. Both objects must also have their own handles and creation statements.

Getting Started

ImageFlow configures and manages data transfers and processing elements on Datacube's family of MaxVideo pipeline processing devices using a variety of guidelines. For example:

- Pipe objects always end with a gateway that has been attached to a surface, and usually (but not always) begin with a gateway attached to a surface.
- Surface objects are used to move data into and out of memory.
- Each surface must be attached to a gateway before it can be used.
- Exactly which elements are included in a given system object is defined in the ImageFlow configuration file.
- Collections of elements are given device names or “handles” by which ImageFlow refers to them.

Computational elements can be quickly reconfigured to perform different sets of imaging operations. These operations are set up prior to the flow of data through the pipes. Once the pipeline processor begins operations, reconfiguration can occur “on-the-fly” between frames, providing exceptional flexibility and power. ImageFlow significantly simplifies pipeline processing control, handling complex synchronization and timing issues for the programmer.

The following sections apply these and other basic ImageFlow concepts and guidelines to a real application.

Acquisition and Display

The acquire and display portion of this application uses the following components:

- One system object
- Four device objects: Analog Sensor (AS), motherboard (AB), Surface Store (AM), and Analog Generator (AG)
- Four surface objects, named by the user
- Four gateway elements: AS_XMT, AM_RCV, AM_XMT, and AG_RCV
- Two pipe objects: the acquire pipe and the display pipe

Depicted in *Figure 8*, the acquire pipe brings the data through the crosspoint to an AM memory surface, which is duplicated to make a new surface. This surface is the beginning of the display pipe, which comes out of the AM, through the AB crosspoint, and into the AG for display.

This application uses no particular AM memory, because it demonstrates how both pipes use the AB crosspoint device. But if it were to use memory 0 (AM00) to hold the final buffer memory surfaces before going out to the AG device, it would not have to go through the cross-point because AM00 is "hardwired" to the AG device. Notice how the display pipe appears in the next example application, which explicitly uses AM00 as the last device before the AG.

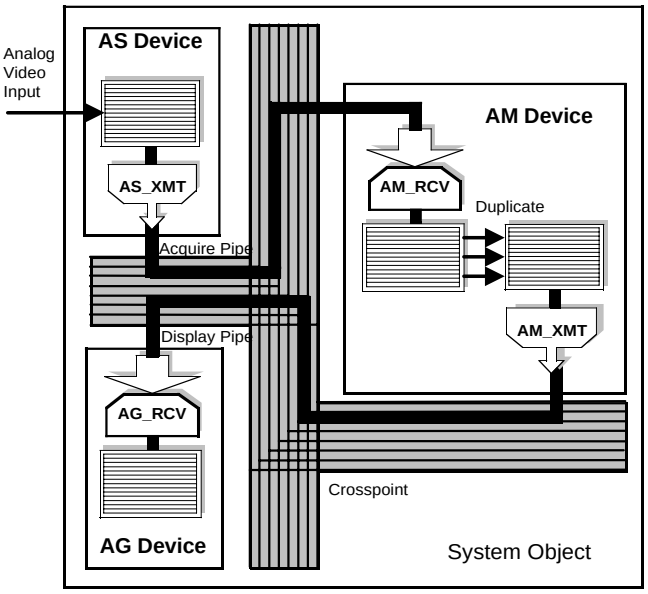


Figure 8: Acquire/Display Application

Adding Processing to the Application

To add a processing pipe to the acquire/display application, the configuration needs another memory device, AM05, used as the end of the acquire pipe (see *Figure 9*). Then pixel data can be routed from AM05 out through the AB crosspoint and into the AU device, where some arithmetic processing is performed. This processing pipe does not end in the AU device, but passes through it and out to AM00. From there, the display pipe goes straight to the AG for display.

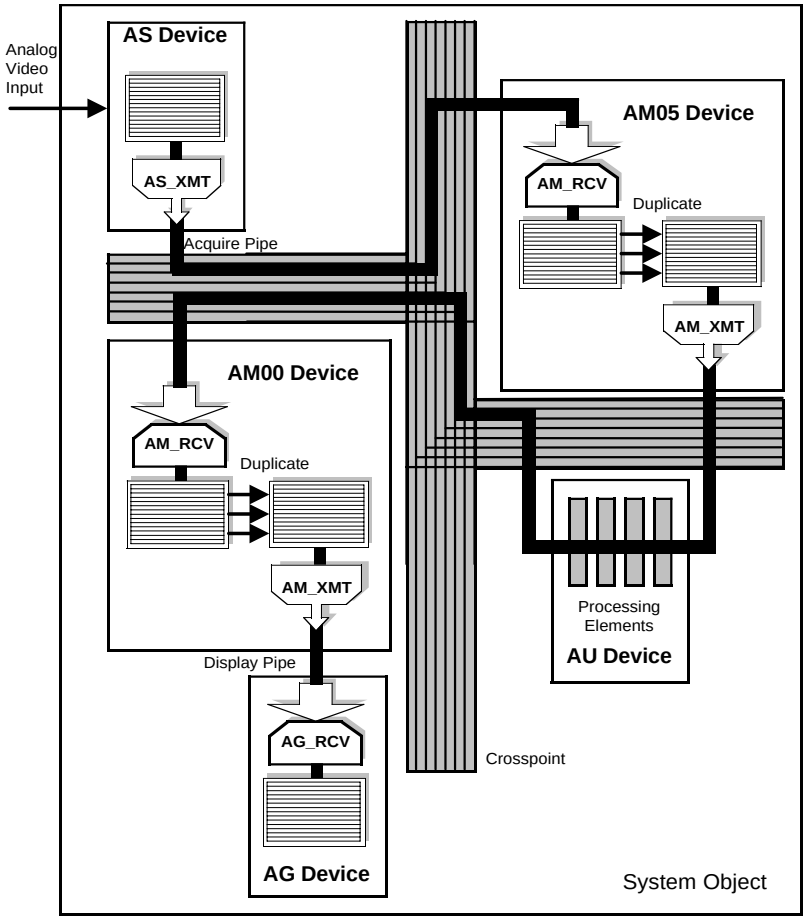


Figure 9: Processing Added to Acquire/Display Application

With processing added, the application uses the following ImageFlow components:

- One system object
- Six device objects: AS, AB, AM05, AM00, AU, and AG
- Six surface objects, named by the user
- Six gateway elements: AS_XMT, AM_RCV and AM_XMT on AM05, AM_RCV and AM_XMT on AM00, and AG_RCV
- Three pipe objects: the acquire pipe, the processing pipe, and the display pipe

In this application, the acquire pipe brings the data through the AB crosspoint to a surface created on AM05, which is duplicated to make a new surface. The duplicated surface is the beginning of the processing pipe, which comes out of AM05, through the AB crosspoint, and into the AU for processing, then back into the AB crosspoint and

into a surface created on AM00, where the processing pipe ends. That surface is duplicated, and serves as the beginning of the display pipe, which does not need to go through the AB crosspoint, but goes directly to the AG for display.

Adding Statistics-Extraction to the Application

With the addition of statistics-extraction, the application is finally an ImageFlow representation of the household plumbing example shown in *Figure 7*. It is similar to the example shown in *Figure 9*, but the processing pipe has become a single-source, multi-destination pipe which forks and goes to both the AU and the AP devices. The end of this fork of the pipe on the AP is a statistics surface used to generate a histogram or extract and list features of the image. The application uses the following ImageFlow components, all shown in *Figure 10*:

- One system object
- Seven device objects: AS, AB, AM05, AM00, AU, AP, and AG
- Seven surface objects, named by the user
- Seven gateway elements: AS_XMT, AM_RCV and AM_XMT on AM05, AM_RCV and AM_XMT on AM00, AP_RCV on AP00, and AG_RCV
- Three pipe objects: acquire pipe, multi-destination processing pipe, and display pipe

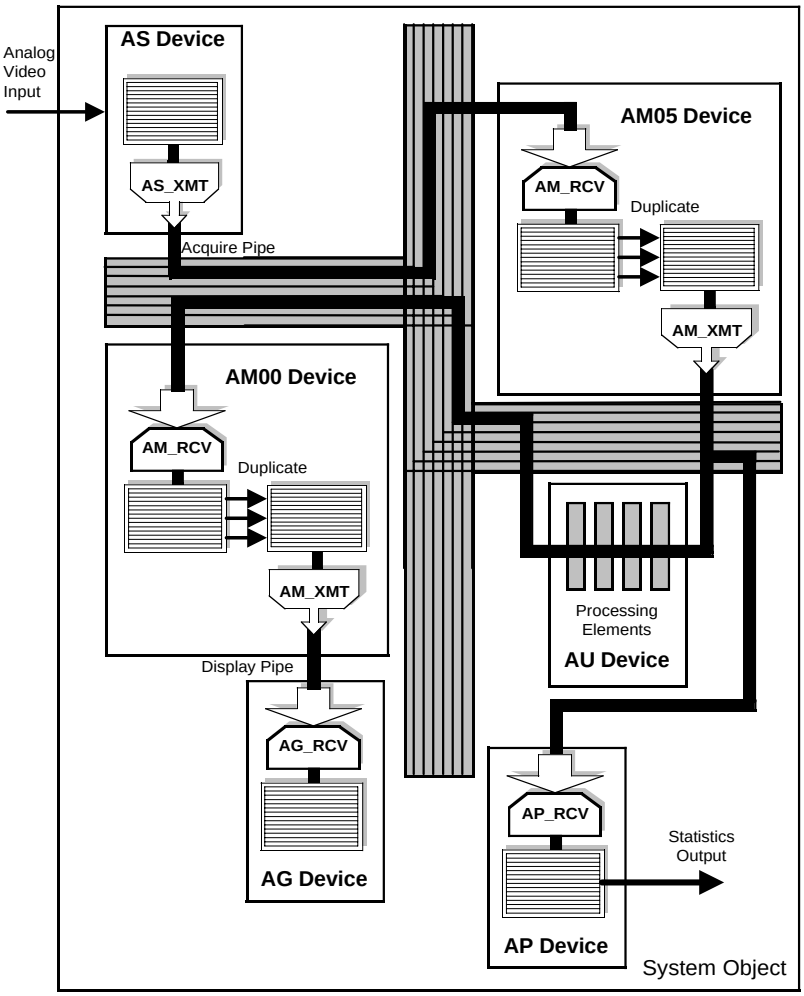


Figure 10: ImageFlow Application with Acquisition, Processing, Switching, Display, and Statistics Output

With these additions, the acquire pipe brings the data through the AB crosspoint to a surface created on AM05, which is duplicated to make a new surface. This surface is the beginning of the multi-destination processing pipe, which comes out of AM05, into the AB crosspoint, and then forks. One fork goes into the AP device, where it goes to a view surface used to extract the statistics data. This data is then passed to a statistics data surface which is not shown.

The second fork of the processing pipe goes into the AU for processing, then flows back into the AB crosspoint and finally into a surface created on AM00, where this fork of the processing pipe ends. That surface is duplicated, and serves as the beginning of the display pipe, which does not need to go through the AB crosspoint but goes directly to the AG for display via a hardwired connection.

Conclusion

The highly advanced pipeline processing architecture provides unmatched power to meet the challenge of processing digital images at frame rates. CPU-based architectures can't compete with the speed and flexibility provided by specialized image processing hardware from Datacube. ImageFlow gives developers complete control over this powerful hardware, while at the same time shielding them from the complexities of synchronization and timing. Capable of managing all the details involved with data transfers, managing events (interrupts), and reporting errors, ImageFlow keeps application development time to a minimum and simplifies migration of application code from one Datacube device to another.

The examples in this document provide a conceptual overview of pipeline image processing using Datacube's ImageFlow software. Considerably more detailed information is available from Datacube, including the Datacube *ImageFlow Programmer's Manual*, part of the standard ImageFlow software documentation set. Datacube also offers technical training on a variety of levels:

- *Introduction to Image Processing* is a non-Datacube specific course that covers the basics of digital image processing.
- Newcomers to Datacube and/or pipeline image processing are encourage to get hands-on experience by attending *Introduction to ImageFlow Using the MaxVideo 200/250*.
- *Advanced Topics in ImageFlow Using the MaxVideo 200/250* provides experienced ImageFlow programmers with the advanced information they need take advantage of ImageFlow's more sophisticated capabilities.

Contact a Datacube Sales Representative at 978-777-4200 to enroll in a course or to request more information.

D A T A C U B E

Datacube, Inc. • Rosewood Drive • Danvers, MA 01923 • Voice 978-777-4200 • Fax 978-777-3117
http://www.datacube.com • info@datacube.com