Technology Review

Low-cost Image Processing Architecture for Airborne Manned and Unmanned Vision Systems

Group 4

The HawkEyed Crew:

Ryan Kitchen

Hailey Palmiter

Scott Griffy

Sponsored by Carlo Tiana and Weston Lahr at Rockwell Collins

Table of Contents

Abstract:

The goal of this project is to create a proof of concept for a low cost real time video processor system for airborne vehicles. This system will combine multiple camera views into one complete picture for remote display on both manned and unmanned commercial and military aircraft. It should be low cost, low power, lightweight, and the code base should be easy to maintain and improve. Rockwell Collins currently develops and manufactures avionic camera systems with integrated video processors. These systems are extremely high performance, but modifications and improvements to the systems are extremely difficult to implement and require a substantial amount of time and money to develop. Therefore our project will be a proof of concept to explore alternative technologies that may be able to provide similar performance, while reducing development and hardware costs.

Key Components:

1. Camera Interface (Hardware Level)

There are three different camera interfaces available on the Jetson development board. Each one offers a different set of benefits and challenges.

1.1. USB 3.0

USB 3.0 is a high-speed plug and play interface that offers a theoretical 384 MB/s data throughput. One USB 3.0 port is available on the Jetson, but additional ports can be added using the mini PCI Express expansion slot.

Benefits:

- Availability of USB 3.0 cameras
- Plug and play, with no camera configuration

Drawbacks:

- Jetson onboard USB 3.0 does not meet the USB 3.0 specification for bandwidth
- Some cameras do not have USB 3.0 drivers for ARM
- USB 3.0 only supports one camera per port due to bandwidth limits
- Increased cost due to board expansion

1.2. Gigabit Ethernet

The Jetson has one-gigabit Ethernet port available. Gigabit ports can handle 125 MB/s, however like USB 3.0 there are expansion cards available that add two additional ports.

Benefits:

- All gigabit cameras use a standardized interface and require no driver
- More high quality cameras available than the USB 3.0 offers
- Simple configuration with video4linux

Drawbacks:

- Most cameras require external power supplies
- Gigabit cameras typically cost more than the USB 3.0 alternatives

1.3. MIPI Camera Interface

The Tegra K1 processor on the Jetson has a direct camera interface, which is designed for onboard, low power camera modules. There are two MIPI interfaces, one 4 channel and one 1 channel interface, but any additional interfaces are unavailable.

Benefits:

- High speed, built in interface
- Low power consumption
- Lower latency because the video processing can be done before buffering

Drawbacks:

- Very limited number of cameras supported
- Specific to the Tegra K1, making it not portable
- Increased hardware complexity
- Increased software complexity due to lack of standardized drivers

Decision:

We decided to use the USB 3.0 cameras because of the availability of low-cost high-quality cameras. Also, Rockwell Collins has several USB cameras they would like us to use. For our customer, it will be best to base the system around the USB 3.0 option. This may require adding a USB expansion card, but the other advantages of using USB 3.0 make it the first choice for camera connection.

2. Video Subsystem (Software Level)

We will have a framework for streaming the video from different components (camera, image processing unit, display).

2.1. OpenCV

We could use OpenCV to stream the video and setup processing of the video stream. This seems like the easiest approach, but would constrict us to the functions given by OpenCV, which might be enough for a proof-of-concept.

Benefits:

- Able to write scripts using python
- Can use OpenCL to take advantage of the GPU

Drawbacks:

- Hides functionality
- Questionable performance

2.2. Direct Memory Handling

We could handle the video stream ourselves on top of v4l provided by the Jetson, but would require some low-level knowledge of the Jetson's hardware. It would allow for more control over the video stream and give better performance.

Benefits:

- Fast performance
- Full control of operations

Drawbacks:

- Requires low-level knowledge of the Jetson architecture
- Complex, which make it much easier to develop bugs
- Not portable to other boards

2.3. Gstreamer

We could use Gstreamer, a free library with video functionality. Gstreamer is widely used as an industry standard, and works on different architectures. If we choose to use Gstreamer we are limited to the API it provides, which is pretty limited because it's meant for video conversion.

Benefits:

- Lightweight
- Portable
- Scriptable
- Easy pipeline-design

Drawbacks:

- Questionable performance
- May not be able to make use of hardware components

Decision:

We will start developing with OpenCV, but if the performance is to slow we may switch to a different option. Video4Linux may be enough to easily move the stream to different components of the Jetson.

3. Image Processing

In order to demonstrate the processing power of the Jetson Tk1, we are going to implement some video processing algorithms.

3.1. On-board GPU

We could make use of the GPU on the Jetson to do image processing to test the performance. GPUs are commonly used for this type of processing and there are many ways to access the functionality of it.

Benefits:

- Easy to use
- Easy to maintain
- Easy to update

Drawbacks:

- Best performance (compared to the ISP)

3.2. Integrated Stream Processor

We could use the two ISPs on the Jetson to do image processing. ISPs are meant for video processing and would provide the best performance. It would allow us to bypass the memory if we connect the camera directly to the camera port. This would require tuning the video processing operations to use the ISPs interface.

Benefits:

- Fast
- Can bypass memory buffers

Drawbacks:

- Not-entirely portable
- Only one camera port

3.3. PCIe GPU

The Tegra has a PCIe port that could be used to install a more powerful GPU for video processing. This could be a good option if the Jetson doesn’t have the required video processing power out of the box.

Benefits:

- Could potentially be very powerful

Drawbacks:

- Uses more power

Decision:

We are going to use the GPU for our image processing. Use of the GPU is widely supported by applications and is easy to code and maintain. Also OpenCV on the Tegra Tk1 directly supports it, and many other choices have support for GPU processing. It will also be able to handle our requirement of 30 frame rates per second processing.

4. Demo Interface

We will need a framework in order to display the video and user interface.

4.1. QT Framework

The QT framework is a stable library that provides user interface.
It is cross-platform and has a simple API, but it is not a lightweight framework. Its ability to display video properly is undetermined

Benefits:

- Easy to use
- Many features

Drawbacks:

- Bulky library
- Not specifically for video streaming

4.2. OpenCV GUI

OpenCV has various UI elements built in allowing for video display and user interaction. It is built into OpenCV and since we have decided to use OpenCV this makes it very simple.

Benefits:

- Part of the OpenCV library
- Easy to code

Drawbacks:

- Slow
- Limited API
- Limited functionality to control the look and feel

4.3. OpenGL

We could use OpenGL to display the video and interface. This would give us full control over the look and feel and it would be very responsive.

Benefits:

- Fast
- Portable
- Powerful (full control of display)

Drawbacks:

- Long development time
- Hard to maintain

Decision:

We will use the OpenCV built-in GUI for the interface, as it is simple to use if we are already using it for the memory manipulation and video processing. It could even be used in combination with other choices, and would still be a good choice as the API is easy to use and is large enough to have all the features we want.