TINE VIDEO SYSTEM – A MODULAR, WELL-DEFINED, COMPONENT-BASED AND INTEROPERABLE TV SYSTEM UNDERGOING A REDESIGN

Stefan Weisse (DV) [#] DESY, Germany

Abstract

In recent years, the usage of TV systems and optical readout at accelerator facilities has constantly been increasing. At the same time, the pace of vendor upgrades of industrial vision hardware has hardly slackened. Because image readout hardware is required to meet special criteria in accelerator physics, vastly different hardware systems are frequently used side by side. Given such circumstances it is not surprising that the imaging software needs to be changed, adapted and updated on a semi-permanent basis. Current TV systems cannot cope very well with rapid software and hardware changes. To improve this, a redesign of the current TINE Video System, initiated at PITZ, was undertaken. Efforts are focused on an abstract, modular grabbing interface, dedicated software components, a well-defined Video Transport Layer and use of standard file formats where possible.

This paper will show current, planned and possible software architectures as well as hardware support and outlines perspectives for near and far future. Although the current implementation is integrated into TINE control system, it is modular enough so that integration into other control systems can be considered.

INTRODUCTION

The origin of the outlined Video System and its predecessor is the **P**hoto **I**njector **T**est Facility **Z**euthen (PITZ). PITZ is a test facility at DESY for research and development on laser driven electron sources for Free Electron Lasers (FEL) and linear colliders [1, 2, 3]. The optimisation of an electron gun is only possible based on an extended diagnostic system including a video system. The goal is to measure the electron beam position and the profile of the beam at different places and by different diagnostic tools along the beam line [4, 5].

The currently installed albeit deprecated Video System grew to its current powerful and versatile state by being modified by constant adaptation to changed conditions [6]. Due to this high rate of evolution, weak points of the original design become more and more evident. Because the world of IT and industrial vision is changing rapidly and thus, constant software adaptation is necessary, work is focused on modularity of the whole system and to rework proprietary developments by using industry standards or well-documented in-house developments.

Emphasis was especially put on

- an abstract, modular grabbing interface
- native JAVA support for client side
- flexible, open documented transport layer, support of colour image transport
- component-based design
- use of standard image formats for permanent data storage
- widely useable Application Programming Interface (API) to support and encourage users of writing their own clients and components

ABSTRACT, MODULAR GRABBING INTERFACE

Over the years it was found out that support for only one camera model cannot provide the various demands regarding image quality, speed of acquisition or cost of readout. For very special measurement demands very few camera types are available on the world market at all, at an exceptional price. A non-standard API is provided by the manufacturer of the hardware. In contrast to that, for simple monitoring consumer webcams are usually hardware of choice. A standard API is used to access such consumer hardware. Generally this does not allow fine-tuning of the hardware but easy and quick integration, sometimes finished within minutes.

Past showed that a variety of different camera types are necessary to be integrated. To provide easy integration of different type or models of camera (image readout) hardware, an abstract software interface was designed. This software interface is called Small Grabber Part (SGP). The way to integrate a new camera is to change / rewrite only certain source code parts of this SGP. All other elements of a complex Video System were and are designed in a way that also this newly created SGP can interact with them instantly. This scheme eases enhancement and adaptation to future requirements that one cannot estimate by now.

VIDEO TRANSPORT LAYER

In the past a proprietary interface was used for image data transfer between server and client parts. This was documented, but limited (only greyscale images, very limited metadata). For the future this interface will only be kept alive for interconnection to old clients.

For new developments a collaboration between DOOCS and TINE [7] control systems at DESY was initiated to make it easier to exchange image data. The result of this is on TINE side the data type CF_IMAGE that consist of a fixed, well-defined header and variable image bits. This data type is used as a standard method of exchanging image data across components of a complex

# stefan.weisse@desy.de

Video System. The perspective is to improve reuse of software work. Parallel developments or reinvention of the wheel are possible to avoid.

NATIVE JAVA SUPPORT AT CLIENT SIDE

For future accelerator consoles and GUI panels Java has already been selected to be used as the programming language and technology for client-side. This applies to viewing as well as processing of image data. Java, in contrast to native binary code, significantly reduces execution speed (and thus performance) of software. This is a compromise to provide platform independence. Live video viewing and processing naturally requires strong performance, difficult to provide under native Java.

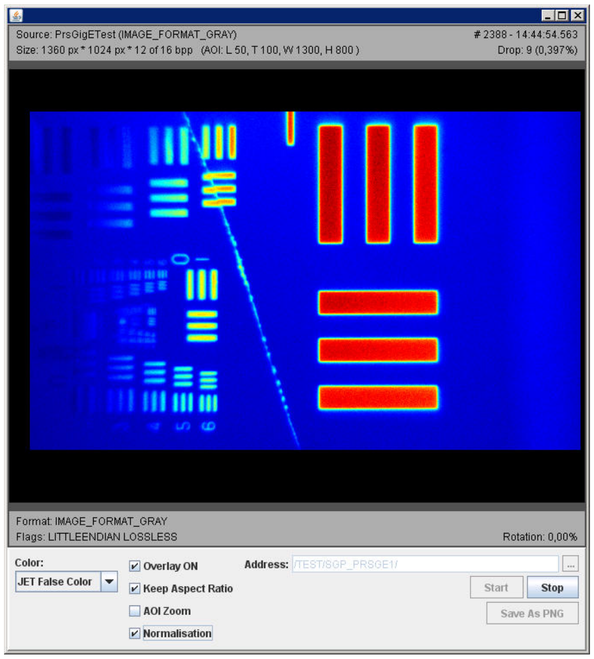
Source: PrsGigETest (IMAGE_FORMAT_GRAY) # 2388 - 14:44:54.563
Size: 1360 px * 1024 px * 12 of 16 bpp (AOI: L 50, T 100, W 1300, H 800) Drop: 9 (0,397%)
Format: IMAGE_FORMAT_GRAY
Flags: LITTLEENDIAN LOSSLESS Rotation: 0,00%
Color: JET False Color
[✓] Overlay ON
[✓] Keep Aspect Ratio
[ ] AOI Zoom
[✓] Normalisation
Address: /TEST/SGP_PRSGE1/

Fig. 1: VideoApplication including TINE VideoBean

The TINE VideoBean, based on TINE AcopBean [8] framework, is a software component for viewing of image data (live as well as single images) and was designed and created with performance in mind.

The results are satisfactory. Nevertheless, a native client can use system resources much more efficiently – with the trade-off of no longer being platform independent. It must be noted that no JNI was used and the whole component was written in 100% pure Java. It provides:

- live image view, receiving video frames via TINE
- still image view
- image display enhancements like false colour modes for luminosity data, histogram equalisation and maintaining of aspect ratio
- optional so-called „on-screen-display“ of image metadata
- image types supported: RGB colour images (24 bits per pixel), greyscale images (8 up to 24 bits per pixel), JPEG colour images, Huffman-compressed greyscale images

LAYERED, COMPONENT-BASED DESIGN

The past has shown that a heterogeneous video system can be very complex. To tear down complexity it was decided to split necessary functionality into smaller pieces that are interconnected using the video transport layer mentioned above. Under ideal circumstances, each component should only fulfil a single operation on the data and then pass it along.

One possible consideration is a breakdown of the Video System in the already mentioned Small Grabber Part, which does all hardware-dependent tasks of video readout, and a CoreProvider component, which takes the raw data that is delivered by the Small Grabber Part and performs software pre-processing of image data such as orientation change, scaling, compression, software binning etc. Other imaginable components could be a former Video System compatibility layer to provide video images for already existing software, a component in order to save short mpeg movies to disk, an HTTP access component that could deliver video images to web browsers, a DAQ storage transformer component…

Layer-based design is planned to be used for complex control and readout schematics. The aim is to keep core (ground bricks) components easy and add more complex functionality by upper layer(s). Examples are the setting of defined readout setups for dedicated measurement purposes and parallel matching setup of interchangeable lenses (remote zoom factor change) and to apply in parallel the matching scale factor (changes with zoom) to each image.

STANDARD IMAGE FORMATS

As default formats for permanent data storage of images and image sequences PNG (for lossless storage) as well as JPEG (for lossy storage) were preselected. Both formats are capable of storing greyscale as well as colour images and are widely used. For storing of image sequences it is planned to store single images in formats JPEG or PNG plus an additional XML metadata file which encodes the sequence order of these individual files. Single files as well as XML metafile might be encapsulated into a ZIP archive to keep directories tidy.

USER LIBRARIES, API

Experience shows that even a designed-to-be-versatile video system can never be flexible enough. In order to give users the opportunity of enhancing the core system to their very special needs, a multi-platform interface library will be provided. It is foreseen that central functionality and reusable algorithms on the inside are outsourced to this library. Functionality moved there will still be referenced in central parts of the Video System while seamlessly being available in source and binary form to users. In addition, an API will be created that provides interaction with the Video System in a simple way.

CURRENT STATUS

Certain Small-Grabber-Parts are nearly finished:

- PCVision analogue PCI framegrabber card
- Prosilica GigE/Vision Gigabit Ethernet cameras
- National Instruments IMAQ interface
- Microsoft Directshow API
- Animation file playback from disk

All implementations are (mainly due to API constraints) only available on Windows XP. The effort to change the platform is considered to be little, source code is written in a platform and compiler-independent fashion. Support depends on a proper API and/or driver that must be provided by a third party. In addition, implementation has been started for JaiPulnix Gigabit Ethernet cameras using JAI API.

The initial revision of the TINE VideoBean has been finished. Small modifications might be necessary once the final set of components will be released. Future efforts are focused on implementing second-level functionality.

The CoreProvider component, which does software-based adjustment of raw video images delivered by SGP components, is under planning.

The VideoService component, an upper layer plus central and very important component is designed at the moment.

NETWORK TRANSPORT, PROTOTYPE IMPLEMENTATION RESULTS

In recent weeks, measurements were performed in order to understand weak performance on network transport level. In spite of a fast switched Gigabit Ethernet connection between two test PCs special care had to be taken to provide high bandwidth data transfer. This incorporates special TINE protocol configuration, special network cards, special configuration of network stack etc. If everything is working like a precisely adjusted clockwork, network transfer speeds of up to 100 MB/s can be reached.

If compatibility on the network site is necessary, an easy to provide general speed is about 15 MB/s. This performance is certainly limited and thus, has a strong influence on component architecture as well as the possibility of sending out live, uncompressed high resolution video images.

In addition, estimates of possible frame rates were done. If the 15 MB/s boundary was not crossed, there was no problem of transferring up to 30 images per second from one PC to the other. This is sufficient for the near future, because for live view hardly more than 30 frames per second are necessary.

A test setup to measure throughput of two Video System components on the same machine via shared memory showed better results. A data rate of 80 MB/s or more can easily be reached. The interconnection between Video System components on the same machine exchanging huge data sets is possible.

All measurements were done using Windows XP Professional. Two PCs equipped with PCI-Express Gigabit Ethernet network cards (1x Intel, 1x Broadcom) were connected together using a switched environment.

PERSPECTIVE

The current effort focuses on getting the main components up to production status in order to satisfy demands from physics still due and to get feedback from real operation. Afterwards there are already two camera models of JAI/Pulnix awaiting proper integration. The implementation of standard formats for permanent data storage will be started afterwards in connection with the creation of general APIs for user demands. It is planned that algorithms used in the components will be exported 1:1 to C/C++ libraries so that the users can also benefit from internal development and vice versa. The philosophy of supporting an API on all used development environments that users use will result in many secondary applications and is meant to remove the workload from the core developer(s).

REFERENCES

[1] F. Stephan et al., “Photo Injector Test Facility under Construction at DESY”, FEL 2000, Durham, NC, USA

[2] S. Rimjaem et al., “Status and Perspectives of the PITZ Facility Upgrade”, FEL 2007, Novosibirsk, Russia

[3] C. Boulware et al., “Latest Results at the Upgraded PITZ Facility”, FEL 2008, Gyeongju, S. Korea

[4] Juergen Baehr et al., “Development of a TV Diagnostic System for the Photo Injector Test Facility at DESY Zeuthen”, FEL 2001, Darmstadt, Germany

[5] Juergen Baehr et al., “Diagnostics for the Photo Injector Test Facility at DESY Zeuthen”, DIPAC 2001, Grenoble, France

[6] S. Weisse et al., „Status of a versatile Video System at PITZ, DESY-2 and EMBL Hamburg“, ICALEPCS 2007, Knoxville, TN, USA

[7] TINE (Three-fold Integrated Network Environment) website http://tine.desy.de

[8] J.Bobnar et al. "The ACOP Family of Beans: A Framework Independent Approach", ICALEPCS 2007, Knoxville, TN, USA