# Interactive 3D Processing Framework

Adam Gleitman
Andrew Shum
Tim Balbakov

6.111 Fall 2011

## 1 Introduction

The 21st century has seen an abundance of advancements in the field of 3D human-computer interaction (HCI), whereby users are able to interact with 3D models either through virtual space or physical space. The advent of technologies such as the Nintendo 3DS$^{TM}$ and Oblong g-speak$^{TM}$ is a testament to current developments in stereoscopy and 3D motion capture / gesture-based interaction. Such developments have led to the use of interactive simulation software to train soldiers in hand-to-hand military combat or to train surgeons in performing complicated operations.

3D HCI technology can be dichotomized by the way virtual and physical spaces are used in the interaction. When the physical space is used for data input, users send commands to the machine using an input device for detecting the 3D position of the user action. When it is used for data output, the simulated 3D rendering is projected onto the physical space through an output device.

In this project, we develop an FPGA-based interactive 3D processing framework to explore both of these modes of interaction. In stereoscopy mode, our system renders a stereoscopic anaglyph image from 2D images captured using two NTSC cameras that are positioned a few inches apart to mimic a pair of human eyes. This 3D rendering is then viewable through an output device, namely red and cyan 3D glasses. In gestural interaction mode, we use the same camera setup as a motion capture input device. Our system captures simple gestures and interprets them to control the viewing position of a virtual rendering of a polyhedron.

# 2 Functionality

## 2.1 Overview

The 3D processing framework works in two modes: stereoscopy (or anaglyph) mode and gestural interaction mode.

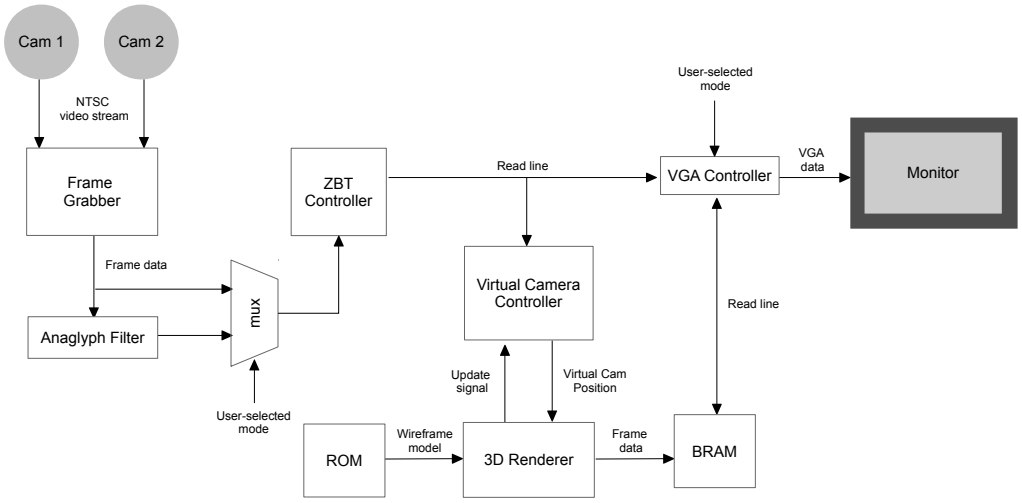


Figure 1: A high-level block diagram of the processing pipeline.

In the first mode, a stereoscopic image is generated that can be viewed with 3D glasses. The frame grabber outputs deinterlaced video frames to an anaglyph filter. This filter uses information from the left and right frames and generates an anaglyph image. This image is saved to a frame buffer by the ZBT controller. The VGA controller renders the resulting anaglyph to the monitor.

In the second mode, a specially marked object is used as a gestural input controller for a virtual camera viewing a 3D model. The frame grabber outputs a deinterlaced video frame directly to the ZBT controller, which is saved to a frame buffer. The virtual camera controller module reads data from the frame buffer to detect the position and orientation of a specially marked object. It determines the location of calibration dots on an object, and uses that information to calculate a virtual camera position. This virtual camera position determines the perspective from which the 3D renderer will render a pre-configured wire-frame stored in ROM. The resulting image is saved to a BRAM frame buffer, from which the VGA controller renders an image on the screen.

## 2.2 Module Descriptions

### 2.2.1 Frame Grabber

The frame grabber interfaced with the ADV7185 video capture chip and simultaneously records two video feeds to memory.

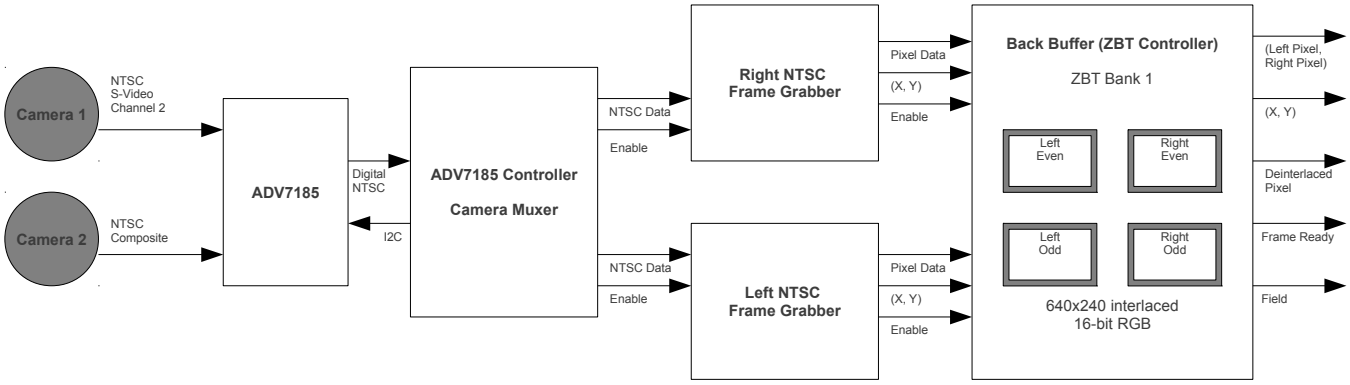


Figure 2: Block diagram for the Frame Grabber module.

**NTSC Decoder Controller.** The NTSC decoder module generates control signals for the onboard video decoder chip. This module is based on the staff provided NTSC capture code, but is modified to capture two video feeds simultaneously. This is possible by processing image data from one camera at a time, and switching between cameras at the expense of frame rate. There are three analog video input lines that are physically accessible on the Labkit: composite, S-Video chrominance, and S-Video luminance. The composite video signal of the second camera will be connected to the chrominance input of the S-Video connector, and the appropriate configuration bits will be set in the video decoder. A register internal to the decoder chip selects which video stream is selected. By modifying the contents of this register, we can configure the video decoder to selectively convert one of these channels at a time.

The controller captures two streams at a reduced frame rate by alternating the incoming video stream. The NTSC standard specifies a 24 frame per second (FPS) frame rate for the interlaced signal that is generated by the cameras. Since the video decoder is shared by two video feeds, it is not possible to capture video at the full frame rate. The subsequent image processing modules do calculations on a whole frame, so the input video stream must be deinterlaced. De-interlacing (which is performed by the frame grabber modules) halves the

frame rate, and switching the input feed discards an input frame. The resulting frame rate is 6 FPS at the original 640x480 resolution.

**Left and Right Frame Grabber.** The left and right frame grabbers will take the interlaced frame data as input and store frames in the back buffer. The frame grabbers convert from the input color space to an internal 18-bit RGB palette.The NTSC decoder controller module enables these modules alternately, so only one of these modules is operational at a time. To deinterlace the input signal, the frame grabber records even and odd video fields into the back buffer. After a full frame is recorded, control switches to the inactive frame grabber module. The active frame grabber outputs a pulse on the ready line to signal subsequent modules to begin processing.

This module and the preceding NTSC decoder will be tested in a testjig that outputs the contents of the front buffer to a VGA monitor, with a user selectable video feed switch.

### 2.2.2 ZBT Controllers (front and back buffers)

There are two one-port ZBT memory chips that allow for 36 bits to be written or read in one clock cycle. Internally, two 18-bit RGB pixels are stored per line of memory in both buffers. Since memory access is time-critical, the back buffer ZBT bank is reserved to the frame grabber modules. The back buffer contains four interlaced video frames at any time: odd and even frames for each of the two cameras. The front buffer is stored on the second ZBT chip, and access to the chip is shared by the virtual camera controller and the VGA controller. Since this chip implements a one-port memory, memory contention will be handled internally by the ZBT controller module. Memory access will be completed in order, and a ready line will be asserted once the memory access is complete. The ZBT memory controller will operate at a faster phase-locked clock frequency than the rest of the modules to accommodate for timing constraints.

If the maximum synthesized clock speed does not allow live video feed, we have several options to optimize the design. We can implement a read/write BRAM cache in non-timing critical modules (the frame grabber and VGA renderer have absolute memory access priority to the back and front ZBT memories respectively). Alternatively, we can slow down the target frame rate until timing constraints are met by periodically grounding the frame grabber enable lines.

The ZBT controller will be tested using the logic analyzer as a standalone module.

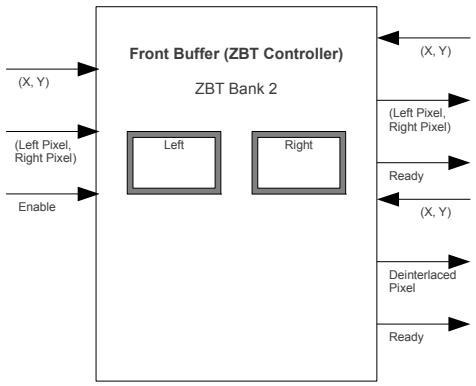


Figure 3: Block diagram for the front buffer ZBT controller.

### 2.2.3 Virtual Camera Controller

The virtual camera controller is responsible for controlling the position of a virtual camera viewing a wireframe object using camera data. The user controls the position of the virtual camera by moving a colored dot. This module tracks the location of the dot on the two cameras, calculates the dot's location in three-dimensional space, and translates its position into change in the virtual camera's position.

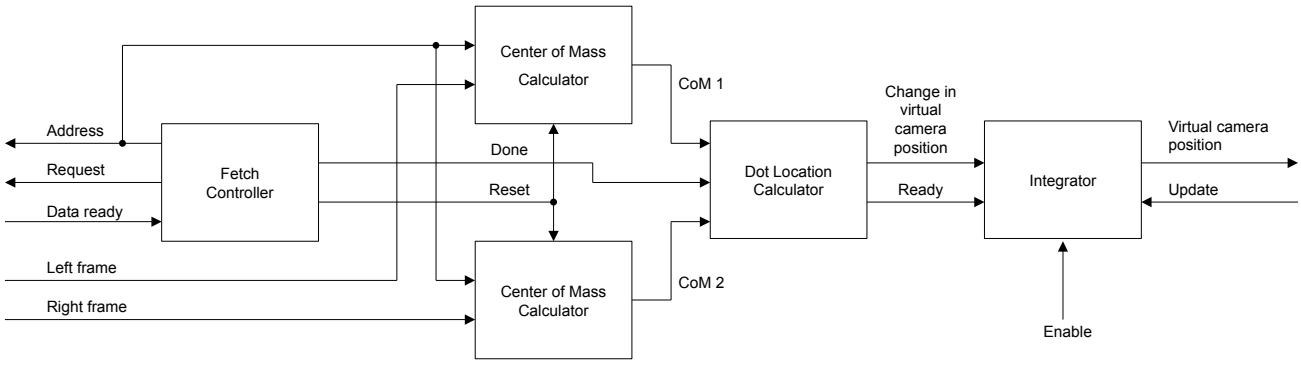


Figure 4: Block diagram for the Virtual Camera Controller module.

**Fetch Controller.** The fetch controller is responsible for coordinating the actions of all the modules in the virtual camera controller. When it is not processing data, the fetch controller holds its request line high, telling the frame buffer that it may write data to it.

When the data ready line goes high, the fetch controller brings the request line low, resets the center of mass calculators, and starts calling addresses so the center of mass calculators can receive frame data. After going through every possible address, this module pulses the done line and brings its request line high again.

**Center of Mass Controller.** The center of mass calculator modules are responsible for locating the centers of the dots on each of the camera frames. There are two of these modules, one for the left frame and one for the right frame. Each module keeps track of a weighted sum of the pixel values, which is used to calculate the center of the dot on each frame. When reset, this sum is set to zero and any additional information used to perform the calculations is discarded.

Each module outputs what it calculates to be the center of the dot on the frame it analyzed. The outputs are not necessarily valid until the modules have received all the pixel data from the frame buffers.

This module will first be constructed in software so it is easier to experiment with different filters and ways to calculate the center of the dot. Once a reliable method has been found, it will be implemented in Verilog.

**Dot Location Calculator.** The dot location calculator is responsible for taking the centers of the dots on each camera frame and calculating the location of the dot in three-dimensional space. Given the properties of the cameras, such as their viewing angle, the distance between the two cameras, and the angle between them, this module performs the necessary math to determine the location of the dot. It then translates this location into a change in the virtual camera position. When this module is told by the fetch controller that the center of mass modules have received all the pixel data, it performs this calculation and pulses its ready line to tell the integrator to update the change in the virtual camera's position.

This module will first be tested by figuring out the necessary mathematical formula, which will then be translated into Verilog.

**Integrator.** The integrator is responsible for keeping track of the virtual camera's position. It takes as an input the rate at which the virtual camera moves and an enable line. When the enable line is low, the integrator does not change the position of the camera, which prevents this module from changing the position of the virtual camera when the input from the dot line calculator may not be valid, such as when the system is in anaglyph mode.

This module keeps track of the virtual camera's position in an internal register. It outputs

the contents of this register only when the update line is pulsed high. This allows the 3D Renderer to read the virtual camera position without worrying about it changing while drawing a single frame.

### 2.2.4 Anaglyph Filter

Rendering an anaglyphic image involves applying separate color filters to two input images. The left image, which is viewed through a red filter, has its blue and green color components are reduced, while the right image, which is viewed through a cyan filter, has its red component reduced. The composite image is then composed by combining the remaining color components into a single color image. When viewed through the red-cyan 3D glasses, the visual cortex of the brain fuses this into perception of a three dimensional image.

As mentioned in Section 2.2.1, the output of the Frame Grabber module is a stream of frame pixel data that is eventually passed into the ZBT controller and stored in the front buffer ZBT. The user-selected mode dictates (through the mux shown in Figure 1) whether or not the stream of frame pixel data being stored is anaglyph-filtered.

Therefore, our Anaglyph Filter module acts as a processing unit that takes a left and right pixel for a given $(x, y)$-coordinate and outputs a single RGB pixel, transformed according to the Optimized Anaglyphs method of [1].

This method works as follows. For a pixel on the left frame with RGB components $v_L = [r_L, g_L, b_L]$ and the corresponding pixel on the right frame with RGB components $v_R = [r_R, g_R, b_R]$, the resulting pixel that is rendered has RGB components

$$v_{anaglyph} = [0.7 \cdot g_L + 0.3 \cdot b_L, \quad g_R, \quad b_R],$$

which can be implemented in Verilog as two 64-row lookup tables and an adder, since $g_L$ and $b_L$ are each 6 bits.

Upon completion of the VGA controller, the Anaglyph Filter module can be tested by simulating a stream of frame pixel data and attempting to filter and display the frames on the screen.

### 2.2.5 3D Renderer

The 3D Renderer module is responsible for pulling the hard-coded polyhedron, specified by a set of vertex coordinates and edge list in ROM, as well as the virtual camera position from the Virtual Camera Controller, and performing all the necessary matrix operations to output a 2D wire-frame image of how the polyhedron appears as seen from the virtual camera position. The module coordinates with the Virtual Camera Controller module by

sending it update signals to specify when the 3D Renderer is done processing data and ready for the new virtual camera position.

This module can be tested by hard-coding both polyhedron data and the virtual camera position and attempting to store correctly transformed renderings onto the BRAM.

#### 2.2.6 VGA Controller

The VGA controller module is responsible for providing a constant feed of either the anaglyph-filtered image we are capturing or a wire-frame rendering of a polyhedron as seen from a virtual camera position. Thus, depending on the user-selected mode, the VGA controller reads either anaglyph frames from the front buffer ZBT or polyhedron renderings from the BRAM and sends them to the VGA at 60Hz refresh.

The VGA Controller can be tested by loading hard-coded synthetic data on the BRAM and ZBT controller, and attempting to render images on the screen based on a selected mode.

# 3 Organization

## 3.1 Division of Work

The work will be assigned according to the three conceptual modules. Tim will work on the Frame Grabber, Adam will work on the Virtual Camera Controller and Andrew will work on the Anaglyph Filterer and 3D Renderer.

## 3.2 Milestone Timeline

By the end of week 4, we plan to finish the video capturing and anaglyph processing work. This entails the completion of Tim's Frame Grabber module and the front buffer ZBT Controller working in anaglyph mode. This also entails the completion of Andrew's VGA Controller module and the Anaglyph Filter successfully integrated within Tim's modules.

By the end of week 6, we plan to finish the 3D rendering mode, which requires Adam's Virtual Camera Controller to be functional and communicating with Andrew's 3D Renderer. During this time, Tim will be in charge of integrating these modules together with his Frame Grabber.

### 3.3 External Components

Two NTSC compatible cameras with composite video output are required. These are supplied by the 6.111 staff. Additionally, red-cyan glasses are needed to view the generated anaglyphs. These are relatively inexpensive and will be purchased online.

## References

[1] Anaglyph Methods Comparison [Online]. Available from: `http://3dtv.at/Knowhow/AnaglyphComparison_en.aspx`. Accessed 2011 Nov 1.