Real-time Frustum Normalization for Video Projectors

"Stay Square"

Anthony Kuntz, Adam Pinson & Daniel Stiffler

Outline

1. Scope
2. Requirements
3. Implementation
4. Applied Skills
5. Milestone Progression

Scope

Our project, real-time frustum normalization, is based around the task of re-squaring video being output from a projector. When consumer projectors are not properly leveled, their throws appear distorted, rotated, or stretched; this is because their cone of light (frustum) does not travel a uniform distance before impinging upon the screen. Current fixes essentially involve adjusting the lens or the device itself, but some premium projectors use software to correct the images. Unfortunately, these tend to be specialized or beyond the means of most people. We propose an embedded pass-through system that sits between a video source and a projector, applying shape correction to the stream in real time. An FPGA would make an ideal hardware accelerator. The end product should be a system which can be inserted between any laptop and projector such that the final image remains properly oriented and keystoned regardless of the moderate (or severe) changes in the projector alignment.

Requirements

There are a multitude of technical challenges with this project. Foremost is the issue of processing high-definition video in real time with an FPGA. Second, we would need to warp an image such that it is properly shaped based on a set of initial conditions, such as the screen distance and relative pitch, roll, and yaw to the plane. Third is interfacing with sensors that will assist us in determining potential changes in distance from projection surface as well as changes in orientation of the projector and then using these new values to create a newly warped image. Fourth, we will need to implement bounding and resizing tools to prevent the image from being cut off when the projector is rotated. These aspects serve as formidable challenges since they are both difficult to

implement and must also be implemented in a manner efficient and fast enough to service real-time video projection with minimal delay.

Implementation

To perform the real-time video processing we plan to use an FPGA, likely the Vertex 7. This board is ideal since it has an abundance of logic slices and MACs which can support multiple concurrent pipelines. In addition, the Virtex 7 has an HDMI peripheral which we would likely utilize for video output. We have already begun investigating video input decoders and believe we should be able to use an off-the-shelf product for decoding VGA or HDMI streams via a digital interface we could connect to the Virtex 7's IO pins (FMC breakout). To determine the distance of the projector from the projection surface we thought about using an ultrasonic or LIDAR sensor, but it warrants more investigation. To determine the pitch, roll, and yaw of the projector we plan on using a high-accuracy inertial measurement unit. Both the distance and IMU sensors might interface to the FPGA using a Raspberry Pi.

Applied Skills

The various sensors used by the project will require a strong embedded skillset. Fortunately, Adam has industry experience in systems integration and we have a solid baseline from 18-349. Daniel's experience TAing 18-240 and 18-341, as well as industry experience with FPGAs, means he is well-equipped to deal with setting up, programming, and interfacing with the FPGA that is used. Anthony has had industry experience with RTL designs, which will aid in the FPGA datapath creation. All three group members have taken 18-447 and 18-341, which will allow for basic architectural understanding and experience in RTL design.

Milestone Progression

> **Disclaimer**: We basically put down random times for this section. Some might be incredibly unrealistic.

1. HDMI pass-through (Video I/O) - **2 weeks**
2. Sensor integration - **1 week**
3. Simple transformation working on one frame buffer - **2 weeks**
4. Simple transformation pipeline working on multiple frame buffers - **1 week**
5. Vertical keystone correction - **1 week**
6. Horizontal keystone correction - **1 week**
7. Rotation correction - **1 week**
8. Bump mitigation - **2 weeks**