Stay Square

Real-time Frustum Normalization for Video Projection

Anthony Kuntz, Adam Pinson, and Daniel Stiffler
Group 5A, Bench 12

Outline:

Problem

In education, business, and collaboration, projected video is becoming increasingly more important as a means of conveniently sharing information. When consumer projectors are not properly leveled, however, the outputted image or video ends up distorted, rotated, or stretched. This unappealing consequence occurs when the cone of light, or frustum, emitted from the projector does not travel a uniform distance before meeting the screen. While current solutions to the distorted image output by a tilted projector, also known as the keystone effect, do exist, they are less than ideal. Many projectors include the ability to move, refocus, or adjust the lens to manually correct for the keystone effect. However, this mechanical solution varies from device to device, requires manual input, and cannot adapt to later changes in positioning without additional manual input. Software solutions, on the other hand, exist only in very inaccessible, high-end projectors. Most importantly, existing solutions are device-specific. As such, educators, businesspersons, and presenters who use various types of projectors in wide varieties of settings cannot count on a reliable, consistent solution. A mobile, dynamic, self-contained solution to the keystone effect, however, would guarantee that an individual's presentation, movie, slides, or demonstration will remain viewer friendly regardless of the venue or setup used.

Scope

We propose an FPGA-based pass-through system that sits between a video source and a projector, applying shape correction to the video stream in real time. Such a solution would act as a mobile and modular fix to potentially any projector, without the need to re-adjust or recalibrate. Thus, 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 changes in the projector alignment. The FPGA will be used as a hardware accelerator to allow for the matrix multiply operations which keystone correction requires to occur swiftly and in parallel, thus allowing for real-time adjustment. Our projected "Minimum Viable Product" (MVP) will be able to apply keystone correction to still input images for vertical rotation (pitch) in a ~30 degree range, at a fixed distance from the wall, with decent throughput but possibly high latency. The FPGA will receive real-time sensor data from an Inertial Measurement Unit (IMU) which will report the pitch of the projector relative to ground, thus assuming a completely vertical projection screen. Additional goals past the MVP will be detailed in the Schedule and Milestones section and include: correction based on horizontal rotation (yaw), correction based on rotation about the axis of projection (roll), and integrating a distance sensor to determine the projector's throw, or distance from the screen being used. Due to the nature of the required digital datapath, necessary hardware, and algorithms to be used, the project will cover the digital design, embedded systems, and signal processing areas of Electrical and Computer Engineering.

Requirements

Our project can be broken down into a few different key requirements: HDMI input/output, sensor data collection and transmission, and frustum normalization video processing.

HDMI Input/Output: A key requirement of our project is that we must take video input via HDMI and output video via HDMI. Although we anticipate having an FPGA with HDMI output onboard, we will need to establish the parameters needed for this device to actually output HDMI. Understanding the interface by which we will provide the HDMI output system with the output framebuffer data will also be essential. All of the FPGAs we have looked at do not have HDMI input natively, and therefore we anticipate needing to use an external HDMI capture and decode interface card. Because of this, we will need to understand the card's interface and determine how to move video input data from the capture card into a frame buffer on the FPGA.

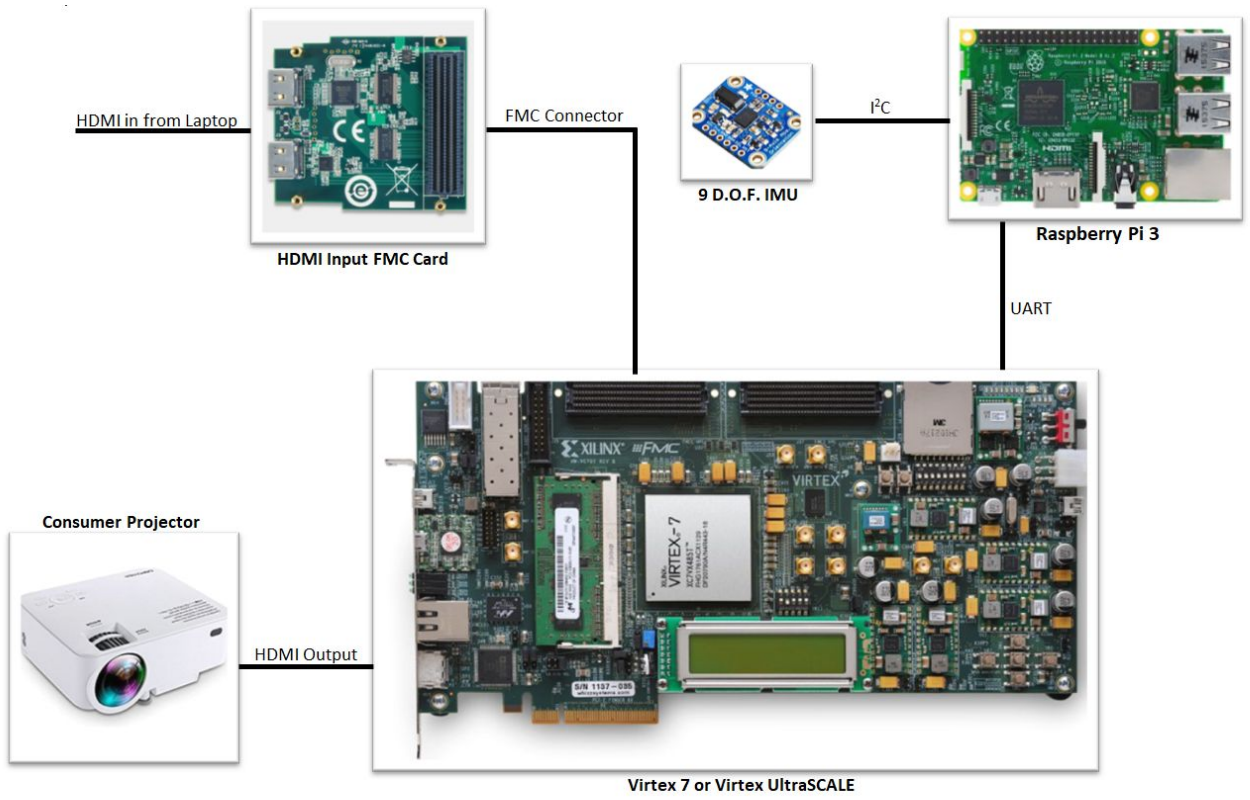
Sensors: In terms of sensor data collection and transmission we will need to capture data from our IMU and deliver that data to the FPGA board. We plan on using an embedded system between the sensor and the FPGA which will likely allow us to use a pre-existing software library to control and speak with the IMU, leaving us to implement communication of the important data between the embedded system and the FPGA.

Frustum normalization: Video processing is likely the most challenging component of this project. Given that we're working with video, we will need to read, write, and transform a very large quantity of data in or near real-time. To begin, the video transformation must be understood and implemented taking into account the sensor input from the IMU. To achieve high throughput, we will likely have to create a pipelined design and duplicate this design multiple times. Furthermore, we will likely run into issues with available memory and need to creatively manage our data to get around such bottlenecks while still being able to process a whole frame for output.

Implementation

The project will be implemented as an FPGA connected to one or more embedded sensors. We have considered multiple different FPGA boards and hope to maximize the amount of available on-chip storage as to prevent us from having to implement a more complicated and temporally costly memory manager. Given this, our team has identified two boards which we believe would suit our applications well, listed in order of preference:
1) Virtex UltraScale VCU108[1]
2) Virtex-7 VC707[2]

At the moment we believe our sensor setup will consist of a Raspberry Pi 3 interfaced via I2C with a 9 Degree of Freedom (D.O.F.) Inertial Measurement Unit (IMU). Based on the IMU sensor data, pre-processed parameters needed for the video transform will be passed to the FPGA over UART. An HDMI decoder board[3] attached to the FMC port of the FPGA will translate the video output of a laptop into a form readable by the FPGA datapath. Next, a triple buffer setup with an arbiter will ensure that the producer (the output of the HDMI decoder) and the consumer (the datapath which transforms input data and returns it) never have to pause to wait on the other. A memory controller will handle storing and retrieving data from the three buffers and an allocator will handle assigning regions of the input image to the different lanes of the datapath. Finally, the datapath will be composed of the multiplication necessary to perform the homography to map the points in one plane to the points in another. This multiplication pipeline will be instantiated several times in a parallel structure, depending on the number of logic elements available.



Team Skills

This project will be heavily leaning on the digital design knowledge of the team as well as our skills with integrating sensors, peripherals, and embedded systems. All three team members have taken the digital design courses 18-341 and 18-447, which will be incredibly useful as we embark on a project which will likely require pipelined video processing. Furthermore, all team members have some background in either video processing and/or embedded systems. We anticipate that all three of us will be working collaboratively on many of the main areas given our shared background. This said, we have a rough division of labor that should help distribute some tasks and make our project possible in the time provided. Adam will work on sensor and HDMI interfacing and decoding. Anthony will work on the datapath and visual output. Daniel will work on frame buffers, arbitration, and memory management.

Schedule and Milestones

There are a few key milestones to our project which will help us gauge our progress along the way. To properly examine these milestones it is helpful to divide the project into two tracks: video processing and video/sensor IO.

In terms of image processing, our first step will be to use a non-FPGA software implementation of keystone correction to verify viability and gain a better understanding of the parameters involved. This should take about a week. Our next step will be writing an FPGA implementation of this image processing for a static image, which should take about three weeks. After that, integration of the video input and the keystone processing will take place, which should take about two weeks. Next, the IMU sensor data will be integrated into the keystone processing. This should take about two weeks. Finally, the pipeline will be further optimized and tweaked to improve performance and throughput, potentially adding in reach goals, which will hopefully take about two weeks.

In terms of video and sensor IO, our first step will be to begin implementing HDMI output and to interface the IMU sensor with the FPGA. This will take about two weeks. Our next step will be finishing up HDMI output integration with the video processing pipeline and beginning HDMI input work, which will take about two weeks. After that, we will finish the HDMI input work and create a functional HDMI pass through (HDMI input and output both functioning and interfaced). This should take about a week. Finally both HDMI input and HDMI output systems will be fully integrated into the video processing pipeline, which should take about two weeks.

Our reach goals are:

- Distance from wall is measured and supported
- Horizontal rotation (yaw) is supported
- Rotation (roll) is supported
- Throughput is high enough to support higher frame rate video output (7.5/15/30 fps)
- Throughput is smooth and fast enough to support an actively moving projector
- Improved demo setup including visualization of correction severity

Our mid-semester milestone entails having working HDMI input and output with functional keystone correction applied to the video, given static and manually entered projector distance and angle parameters.

Given the scale of our project, we anticipate using most of the semester to implement our MVP and utilizing any remaining time to implement stretch goals and polish our presentation. A more detailed breakdown of our expected schedule is in the table below.

Week	Due Date	Goal, Feature, Progress
0	9/22	Present project proposal (Thursday) and submit project proposal (Friday).
1	9/29	Preliminary parts list submitted; keystone correction using Python library on a hard coded test image, without FPGA, implemented; Sensors have been explored.
2	10/6	FPGA can be used to output a static test image via on-board HDMI; Means for interfacing with sensors has been determined and tested.
3	10/13	Substantial progress has been made on FPGA datapath.
4	10/20	FPGA can be used to apply keystone correction to a static test image and output the result via on-board HDMI, given a distance from wall and degree of offset.
5	10/27	FPGA can function as a video pass-through device. Sensor interfacing works perfectly and can be used to input meaningful, well-structured data to the FPGA.
6	11/3	FPGA can receive video input of a test image, apply keystone correction given a manually input distance from wall and degree of offset, and output the result via on-board HDMI.
7	11/10	Substantial progress has been made towards integrating sensor input, video input, and FPGA image correction datapath. FPGA datapath has been polished and improved.
8	11/17	FPGA receives input from sensors, uses the degree of offset of the projector to apply keystone correction to an input test image, and outputs the result via on-board HDMI.
9	11/24	MVP implemented.
10	12/1	Datapath is optimized to allow for decent throughput for video projection. Early reach goals are attempted: distance sensor, yaw, and pitch support.
11	12/8	Means for demoing device are finished. Device is demo-friendly. Any casing, housing, or dress for the device has been polished, finished, and integrated.

List of References

1. Virtex UltraScale VCU108
   https://www.xilinx.com/products/boards-and-kits/ek-u1-vcu108-g.html#overview

2. Virtex-7 VC707
   https://www.xilinx.com/products/boards-and-kits/ek-v7-vc707-g.html

3. We are considering using the board at the link below for HDMI input via FMC to our FPGA. This card decodes HDMI and provides a large parallel IO interface to the image data which should allow us to interface well with the rest of the datapath.
   http://store.digilentinc.com/fmc-hdmi-dual-hdmi-input-expansion-card/