HoloPyramid

Author: Breyden Wood, Grace An, Jullia Tran: Electrical and Computer Engineering, Carnegie Mellon University

Abstract**—**Holograms and holographic illusions seem the realm of science fiction, but the technology is here, as easily encountered as a phone app and "do-it-yourself" hologram plastic pyramid. Currently, holographic illusions are currently largely used only for entertainment, gimmicks, and at a relatively small scale with premade renders. Our goal is to leverage this existing technology to create a highly useful and immersive presentation tool: a holographic pyramid that displays a scaled-up 3D illusion of a smaller object from a local studio. The presenter would be able to easily move the object and display moving objects with no overhead of preparation or rendering time. Thus, our product would be invaluable for archeologists, researchers, and scientists, who may need to show an artifact or small piece of equipment, small animals, or robots to a large audience.

Index Terms—Holographic Illusion, Presentational Tool, FPGA, Real-Time Image Processing

I. Introduction

The use case of our project is to provide an immersive and interactive presentation tool that displays small archaeological objects or small moving things in 3D with enlargement for easier viewing by a group of people. The goal of the project is to display an enlarged, 3D holographic illusion of a moving object in real time and offer ways to interact with this object for the presenter.

Because of this, our requirements will be focused on several criteria including usability, enlargement, timing, video frame quality, and illusion. We want to give the user an integrated, compact design that is easy to use. To better display the object to a larger audience, our project should enlarge a 3'' object at least four times over its real-world size. Because this is an interactive presentation tool, our design must maintain a stable real-time capture and output of all video feeds of a uniformly lit object for ease of showcasing actuated objects. To maintain the details in the object, our project needs to generate a crisp (see 2.E) video stream to the display. Lastly, this display needs to recreate a 3D projection of the local object under office lighting using a reflective, transparent pyramid to generate the Pepper's Ghost effect illusion.

Currently there are existing designs in the market that use pyramid displays with a smart-phone screen. However, these are generally designed for entertainment purposes and are very small (Fig. 1) -- typically using pre-baked video feeds. These videos would need to be pre-rendered and pre-recorded to be streamed through the smartphone to create the holographic effect. Because of these existing limitations, we want to expand this design from just an entertainment tool to a full-fledged presentation tool by providing a real-time video feed at a larger scale while still maintaining the holographic illusion.

II. Design Requirements

A. *Usability*

The design must allow the user to interact with the object while presenting. This requirement comes from the use case of the project, which is to provide an interactive presentation tool.

B. *Object Size*

Our holographic display must be able to replicate and enlarge objects up to 3" on the diagonal, without losing any edges to clipping. We will test our system with objects at or under 3" that are in a variety of differently shaped objects (small toys, small figurines, credit cards, a hamster, jewelry, etc.)

C. *Enlargement*

Our holographic display must enlarge any 3" object by a factor of at least four times. Thus, a 3" object should be enlarged to a virtual size of at least 12". To test this, we will place a variety of static objects of known sizes less than 3" and project it to the hologram, then measure the apparent size using a ruler held inside the middle of the hologram. Because the virtual object appears to be suspended in the pyramid, our virtual-image measurements must be taken planar with the virtual object to be accurate. We will compute the enlargement by dividing the virtual size as measured by the ruler by the actual size of the object in the studio and making sure that this factor is greater than or equal to four.

D. *Latency and Real-time Performance*

The holographic pyramid must display objects in real-time for there to be no perceivable delay between interaction with the object and its projection. Thus, the sum of all latencies in the system (end-to-end latency) be less than 250ms, which is roughly the limit of what humans have as reaction time [2].

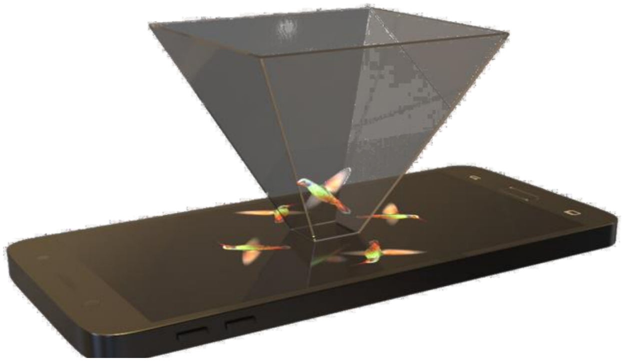


Fig. 1. Example of holographic illusion with a phone and pre-baked images [1]

Total system latency is defined as the delay between a change or interaction with the object and a change in the virtual projection of said object. This is tested by the following: Flash a light in the studio and measure the time it takes for the flash to appear in the hologram using a high-speed camera. The iPhone XS slo-mo camera (owned by one of our team members), or any other smartphones with slo-mo cameras, records at 240fps, allowing us to take latency measurements with a resolution of ~4.2ms. Thus, at 240fps, our total end-to-end latency must be less than 60 frames, which can be counted from the video.

E. Frame Rate Stability

Our holographic display must smoothly display the video. In video feeds, stutters and hitches that contribute to loss of smoothness occur when the time it takes for the video data to be processed and output exceeds the frame-time of the display. This is detrimental to the user experience, so we require that all processing must be done within one frame. The standard for display frame rates is 60Hz, thus all our processing is bounded to 1/60th of a second. Since all this processing occurs on the FPGA, we require that the FPGA processing for every frame be under this time. This can be done by counting the number of cycles taken for processing and ensuring it always remains under 1/60th of a second during normal operation. We plan to use the 7-segment displays to display the maximum number of cycles taken for all frames in operation and periodically checking it remains under the cycle count corresponding to 1/60th of a second over a period of several hours displaying a variety of images.

F. Frame Quality

Because of our project’s design goal as a presentation tool, it is critical that details are preserved on the holographic illusion. Thus, each video frame must be photographically crisp. This can be determined quantitatively through an MTF test [14], which measures how fine-grained details can be displayed in an image. A photographically crisp video frame is one in which separation and contrast between fine details is maximized as much as possible or limited solely by the resolution of capture. To evaluate our MTF score, we will capture identical images of our projected objects using a high-end digital camera owned by one of our team members and adjust the resolution to the same as that of our holographic display. Because the high-end camera’s MTF score is definitionally resolution-limited in this comparison, our hologram must match its score to pass this test.

Furthermore, we require that the display video feed must have minimal distortion of less than 5% change in angle throughout the image on all four sides of the pyramid. To evaluate distortion of the display, parallel lines can be projected onto the display as a test card image, where the result of the test can be measured using Lightroom.

G. Illusion

Our final requirement is the strength of the holographic illusion. To achieve a strong holographic illusion, the object needs to appear suspended in the display. To do this, we need to sufficiently remove the background behind the object in the studio while avoiding incidental object removal. We will measure this and define our criteria for success by capturing the display output and testing it in Lightroom to determine that we have removed >95% of the background and <5% of the object.

III. Architecture Overview

We want to begin this section by introducing the user experience of our project. The user will be able to place an object in the studio and see a holographic enlargement of the object being displayed on the pyramid, appearing to be floating in the center. The user can either interact with the object inside the studio or the object can actuate and these interactions will be captured inside the live studio through cameras, where the video stream would be processed inside an FPGA and outputted to the display in real-time. This flow of data corresponds directly to our system design, as shown in Fig. 2 (Simplified version of the system diagram for the purpose of showing data flow).

The cameras are positioned in a small live studio, where the object will also sit. This live studio consists of four cameras positioned at the center of four walls surrounding the object. This is so that the full size of the object can be captured in each frame of the video. The walls of the studio are designed in one solid color, chosen specifically for the purpose of background removal and minimizing glare. This design choice is made based on our frame quality and illusion requirements (See 2.E and 2.F). The size of the studio is calculated based on our object size requirement (2.A). The top part of the studio will be open, allowing the presenter to interact directly with the object with a tweezer, upholding our usability requirement. A more detailed description of the studio design can be found in section 5.F.

The pipeline diagram (Fig. 2) shows the data flow of the camera inputs, flowing into the FPGA. From here, the video stream will be decoded and processed using a background removal filter, more formally referred to as a *chroma-key* filter, which is an image filter that replaces a block of a particular color with another color or image (in this case, replacing the background color with black). This filter is designed so that the hologram effect would be enhanced, following our illusion requirement. Through the removal of background using our chroma-key filter, the object will appear floating, increasing the effect of the illusion. After being processed by this filter, the data will flow through an image combiner, where the processed data will be assembled into one frame before going to the VGA protocol controller that outputs to the display.

We introduce a simple 3D display based on the principle of Pepper's Ghost with ray optics, named after the creator John Pepper [3]. This is an old technique that causes the object to appear floating in air. This optical illusion involves a large plane of glass or other reflective surface, placed at an angle to project a person or an object from a room or a screen. This is the main principle on which the pyramid in our design functions. We expand on this principle: instead of just having one pane of reflective surface, we have four. This is so that the viewer does not only see one aspect but can see four sides of the object when they walk around. Hence, the illusion of a 3D object can be observed, giving an effect of a live hologram.

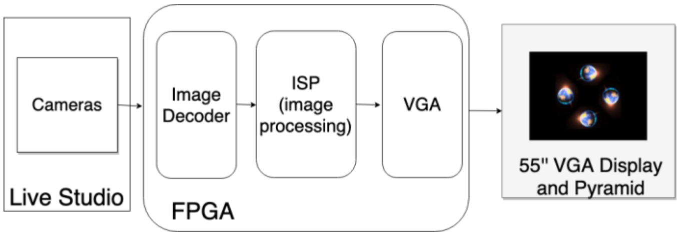


Fig. 2. Simplified system diagram showing data flow

IV. Design Trade Studies

In this section, we discuss the design trade-offs and design choices behind the components of our project.

A. *Choices of FPGA*

The FPGA is used for most of the computation in our system: receiving video frames from the OV7670 cameras, processing the video frames, and outputting each resultant combined video frame to the monitor over VGA. Thus, our FPGA also needs to be able to support the four input decoders, image processing unit, frame buffers, and VGA controller. As a result, we need a board that maximizes the number of logic elements, amount of memory, and number of I/O pins. We identified two choices of FPGAs in Table I. The DE0-Nano has a greater number of GPIO pins, but fewer logic elements and less memory, hence, we are selecting the DE2-115. While the DE2-115 does not readily have as many I/O pins as that of the DE0-Nano, it can be extended using a daughter card as needed. Memory-wise, our FPGA needs to at least store each camera's video frame (four 640x480 frames) and the output video frame (720p display). At the OV7670's RGB565 color scale, this comes out to 5.22MB. An estimate of how much memory is required is shown below.

Minimum of memory used by to store camera frames:

$$240 * 240 * 12 * 4 = 0.3456 \text{ MB} \quad (1)$$

Preferred estimate of memory used to store camera frames:

$$480 * 640 * 16 \text{ (bits per pixel)} * 4 = 2.4576 \text{ MB} \quad (2)$$

Memory used by display frame buffer:

$$720 * 1280 * 24 \text{ (bits per pixel)} = 2.7648 \text{ MB} \quad (3)$$

Furthermore, this board needs to have enough I/O pins to interact with the cameras in the studio. Our FPGA needs to interface with four OV7670 cameras, which each have 18 pins that need to be connected to the FPGA. Some of these pins can be tied together and do not necessarily connect to the FPGA's GPIO pins, such as power and ground lines. However, as a conservative estimate, our FPGA needs to have 72 pins to connect to the cameras. The DE2-115 does not have this many GPIO pins out of the box, however, it has an HSMC expansion header which we are using to add the necessary GPIO pins with an easily available daughter card. With this card, the Altera DE2-115 meets the memory and GPIO pinout requirements.

Importantly, the Altera DE2-115 also has general PLLs. The PLLs are especially important for the VGA output and for the cameras, as 720p output requires a pixel clock that is higher than the 50MHz internal clock and the OV7670 requires a pixel clock lower than the 50MHz internal clock. Using them, we can generate separate clock signals for both interfaces. The DE2-115 also has a VGA port and DAC that are capable of 720p video output to the display. We use VGA for our output because the protocol is easy to implement in hardware, can scale to different resolutions, and is supported by our FPGA without the need for an additional daughter board. Furthermore, this protocol is easy to adapt to other protocols (such as HDMI) allowing us to use our display.

TABLE I. FPGA Comparison

FPGA	DE2-115	DE0-Nano
Embedded RAM	3,888 Kbits	594 Kbits
SRAM/SDRAM	2MB SRAM 128 MB (4x32MB) SDRAM	32MB SDRAM
Number of logic elements	114,480	22,320
Number of GPIO pins	~40 + 80 (on HSMC expansion card)	153

B. *Choices of Camera*

Our design requires cameras that have at least 240x240 resolution and RGB565 color-scale for the following reasons: 240x240 resolution is needed because four video frames are combined and projected onto a 720p display, as shown in section 5.B. As shown in Fig. 3, about 1/9 of the cropped 720p display displays the processed output from each camera, so we need at least 240x240 resolution from each of the cameras. Any extra resolution available would also be preferable because it allows us to potentially scale up the resolution, zoom, pan, or use other image-enhancing techniques with the cameras. Finally, at least an RGB565 color scale is needed to provide sufficient color detail for the human eye.

There are several camera groups that are available commercially that operate using NTSC, USB, or VGA protocols. Through comparing these protocols, VGA provides ease of implementation and less overhead needed for the decoding unit when implemented on an FPGA board. Three VGA cameras that meet our budget and achieve our resolution and color-scale requirement: OV7670, OV7725, and OV5642. These are all viable options, but we chose the OV7670 because of its inexpensive cost and better documentation and reference support. The OV7670 supports 640x480 resolution and RGB565 color scale [11].

C. *Material of Pyramid*

We have four main requirements for the composition of the pyramid panels. This material must be:

1. Fully transparent and reflective - Pyramid panels must be able to reflect the display and easily transmit light to create the holographic illusion.
2. Stiff - Pyramid panels must be able to maintain shape precisely to avoid distortion in reflection.
3. Thin - Pyramid panels must be thin to avoid doubled reflections that distort the holographic illusion.
4. Durable - Our pyramid panel needs to be both shatter resistant and strong so that thin, large panels are not easily damaged during movement.

To create the Pepper's Ghost illusion, we need to pick a material that is transparent. Furthermore, because of the way

Pepper's Ghost illusion works, the pyramid needs to sit on top of the display screen. Because of this, we need to pick a material that is not too heavy. Because of this, we have two choices for the material: glass and plexiglass. Both glass and plexiglass have the property of being transparent and have properties of reflection and refraction of light that is needed to create the holographic illusion. To fulfill both the f illusion requirement and weight constraint, we will be looking at the reflective index, light transmission rate and specific gravity. We will also consider the factor of ease of construction because this panel will be constructed in-house, which can be measured through tensile modulus of elasticity.

A refractive index that is closer to 1.00 - that of air - would result in a more transparent material and less optical distortion [4] of the background, which is less preferable for the purpose of this project as we want to minimize distortion in the illusion to uphold our illusion requirement. As presented in Table II, plexiglass has a lower refractive index than that of glass, which would help to create a stronger illusion. Both materials have the required light transmission percentage needed for the Pepper's Ghost effect. Through comparing the specific gravity between glass and plexiglass, we can compare how heavy the two materials are when deciding on which one would better fulfill the weight constraint. With the property of being much lighter than glass as shown with smaller specific gravity, plexiglass is also lighter than glass which will make it easier to be mounted on top of the TV. Because of this, we will be using plexiglass for the pyramid's panel.

Another point we considered is also the thickness of the plexiglass needs to be kept at minimum. This is because we want to avoid reflections from the back surface of the material, which would create a "double reflections" effect, where the backside reflection has a slight offset from the front reflection. This can be avoided using thinner sheets so that the two beams would overlap, which further narrows our choice of plexiglass to be around ⅛'' thick, as this is the thickness that is commercially available.

A lower tensile modulus of elasticity means that the material is more prone to deformation when stress is applied, or less stiff and more flexible. With lower tensile modulus elasticity, plexiglass is more flexible and less stiff and so it is easier to be cut during construction of the pyramid.

Material	Glass	Plexiglass [6]
Refractive index	1.52 [7]	1.49
Specific gravity	2.4 [8]	1.19
Tensile Modulus of Elasticity	50 - 90 GPa (7.25e6 - 13.05e6 psi) [9]	450,000 psi
Light transmission	90% [10]	92%

TABLE II. GLASS AND PLEXIGLASS COMPARISON

D. *Lighting*

We require that the background of the studio must be uniformly lit on all four sides and the floor. This is necessary in order to ensure a uniform background color, so that our chroma-key algorithm can successfully remove all pixels of the background while leaving the foreground untouched. If there are shadows on the background or the background is unevenly lit, then the two following scenarios result: First, the chroma-key algorithm is unable to remove the background at the specified sensitivity. Second, the chroma-key algorithm removes part of the object because the "distance" metric is so coarse-grained, the algorithm removes both similar and farther away colors from the background.

We used LED strips on the four corners of the studio as well as felt to diffuse the light. The LED strips had a number of advantages: The multiple light sources provided more uniform lighting, and the adjustable remote for the LED strips also allowed us to dynamically change the brightness and/or colors to find the lighting that best uniformly lit the studio. We could also diffuse the lighting of the LEDS close to the floor (which tended to light the floor more brightly than the walls), due to the advantage of multiple, small light sources.

An alternative for lighting we also considered was point lighting. The sun represents an example of point lighting, so it

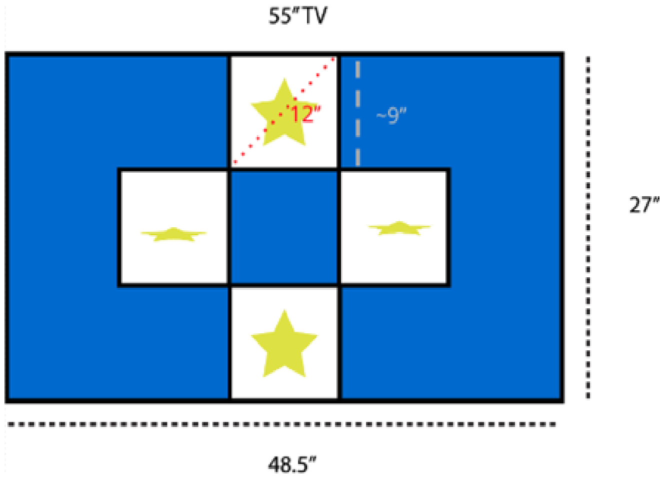


Fig. 3. Diagram showing the layout of images on the monitor

seemed reasonable to attempt it. Point lighting caused significant shadows and failed to be workable in comparison to lighting from LED strips.

The goal of our design is to create an interactive presentation tool. Because of this, the user needs a way to interact with the object such that the frame quality requirement of the object is maintained. To maintain frame quality, the studio will need to be uniformly lit. This can be done in two ways, either with point lighting or with LED strips running along the

E. Choices of Memory

One of our trade-offs is the choice of memory to use, specifically between embedded-RAM, SRAM and SDRAM. While choosing between these types of memories, we have to keep in mind the requirement of real-time performance and the available resources for each of these types. The real-time performance requirement provides a lower-bound for the bandwidth of the memory. This means that in order to fulfill this requirement, stores and reads from this type of memory needs to be less than 250ms. Because of this, BRAM seems to be the obvious choice since it can deliver stores and reads at a much higher frequency than that of SRAM and SDRAM. However, BRAM is bottlenecked by the resources available on the FPGA. BRAM only has 486KB (3,888KBits) while there are 2MB of SRAM and 128MB of SDRAM available. With the requirement of 720p image resolution, 486KB isn’t enough to store a frame. To solve this problem, we optimized memory storage so that the BRAM can handle a full frame, at 720p resolution. This is done by only storing the active frames video feed coming from the four OV7670 cameras at 240p by 240p pixel. We use the highest color depth settings available on this camera, which is RGB565 and store 12 bits at each pixel location in the buffer. Hence, as computed in section 4A, this comes out to be 0.3456 MB or 345.6KB, which embedded RAM can support. We then upsample the pixel to RGB888 later on in the pipeline to adhere to the VGA protocol by using bit extension.This choice allows us to fulfill both the requirement of real-time at 720p resolution, with a 83% reduction in memory. Timing becomes much simpler since BRAM operates at a much higher bandwidth than SRAM and SDRAM.

F. Studio Background Color and Materials

We used dark green felt for the background of the studio. We used felt because the matte material prevented reflections that might cause uneven lighting. As mentioned in Section IV.D, the chroma-key algorithm has a strong requirement for a uniform background color to ensure the most effective background removal possible. We also experimented with the color of felt, attempting black, dark green, bright green, and dark blue. These were selected because green, blue, and black are common backdrop colors for photography studios.

A brighter background color is better for the OV7670 cameras’ auto-exposure settings because the color of the background is more similar to the color of the object. If the background is too dark, such as with black felt, then the video frames of the object are “whited out”, with the displayed object appearing extremely bright with few visible details. However, brighter background colors also have more significant issues with uneven lighting because minute differences in bright background colors were easily picked up by the chroma-key algorithm. Bright green results in good camera auto exposure but varied background color. Thus, dark green was found to be the most successful compromise among the attempted colors in ensuring details and appearance of the central subject as well as success of background removal.

We also initially used construction paper as a matte background material that is easy to tape inside the live studio. Unfortunately, we found that construction paper is too reflective, causing a varied background color that is unsuitable for chroma-keying. Felt was less reflective and was ultimately used.

G. I2C Protocol Implementation

In order to adjust the camera settings for the four OV7670, I2C protocol needs to be used according to the documentation provided. We have two choices to implement this protocol, either on the FPGA or through the Arduino. We first attempted to implement the I2C protocol through the FPGA. However, this proves to be much more complicated. Because of this, we switched to implementing it on the Arduino, which already has some support for this protocol and this keeps our logic elements count for the FPGA design. This saving in logic elements count is helpful because the extra LEs are used towards building the embedded RAM buffers, which is needed to fulfill our Latency and Real-time Performance requirement.

H. Pyramid Shape

To fulfill our frame quality requirement, minimal distortion needs to be achieved. Along with this is also the crisp quality of the image (Section 2.E). To fulfill our crisp image quality requirement, we select the thinner plexiglass to avoid doubled reflections (Section 4.C). However, thinner plexiglass lacks the support to hold its shape at larger size. Sagging in the pyramid increases distortion, which violates the minimal distortion requirement we have. To overcome this problem and achieve a balance in the trade-off between distortion and doubled reflections, we added a top above the pyramid to maintain the shape, which helps the pyramid hold up its shape easier and reduce sagging while avoiding doubled reflections in thicker plexiglass. An advantage of this top is also that it darkens the area immediately beneath the pyramid, improving the strength of the holographic illusion.

I. Image Processing Alternatives

Because it is much easier to write software compared to hardware descriptions, a justification should be provided for the use of hardware. Alternative devices using software such as a Raspberry Pi or a CPU are unable to meet the timing requirements and the ease of use that our project requires.

The Raspberry Pi struggles to maintain a high FPS when handling multiple video streams with extensive image processing, and its image processing fails to scale for increased resolution and processing requirements. The Raspberry Pi also cannot meet real-time timing constraints as its performance dips when the OS handles background tasks.

A CPU can handle image processing of large, high-resolution images, but not in real-time. CPUs are designed to handle multiple tasks at once and cannot be relied on to provide processed images at 60FPS and with stable timing. Additionally, a full PC cannot be easily integrated into this setup and relies on the user to configure the hologram creates the potential for more user error in a complex setup.

In contrast, an FPGA can support real-time image retrieval from multiple cameras with image processing, all while maintaining stable and predictable frame timing. An FPGA can also be easily integrated into our presentation tool; our FPGA design also serves as a proof of concept for an ASIC to be created for a real-time holographic illusion display. Thus, our choice of hardware is most advantageous over possible alternatives.

Despite initially considering embedding image filters such as sharpness and brightness to the FPGA, we found that using the TV and cameras' in-built sharpness and brightness filters were preferable. The TV and cameras' filters were sufficient to meet our frame quality requirements. Avoiding implementing sharpness and brightness filters to the FPGA also allowed us to decrease our logic element count and also significantly decrease our memory usage, as an FPGA sharpness filter performs image convolution and requires significant memory usage.

V. System Description

A. Monitor

For the Pepper's Ghost effect to work, an image source must first be displayed and then reflected off the standing pyramid. The effect works by taking a square panel and dividing it into 9 sub-squares, four of which have the side images of the object projected. Each sub-image is reflected off one of the pyramid sides to form one side of the virtual image. To ensure the projected image is as large as possible, we have chosen a 55" flat screen HDTV (16:9) as the image source to be placed underneath the standing pyramid. This sets each sub image to ~9" square with a diagonal size of ~13", meeting our design requirement of enlargement. Furthermore, the HDTV has a flat panel with accurate colors, allowing us to display the image on the pyramid without color or image distortion. Our FPGA outputs video data over the VGA protocol, which unfortunately does not match the input of our TV as it only accepts HDMI. To solve this, we are using a simple VGA-to-HDMI converter which allows us to keep both the benefits of VGA output as well as the benefits of the HDTV.

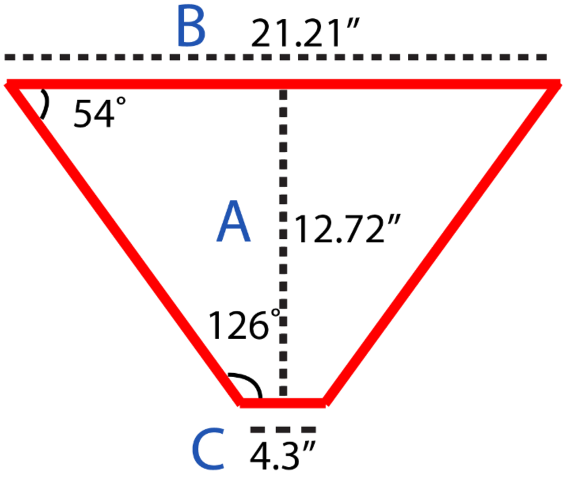


Fig. 4. Diagram showing the angles necessary for one panel of the pyramid

$$\text{Image side length} = (3" * 4) / \sqrt{2} = \sim 9" \quad (4)$$

B. Pyramid Design

The pyramid is where the holographic illusion appears. To create an illusion of a hologram that appears floating in the middle of the pyramid, the tilt angle on each pyramid panel needs to be at a 45˚ angle to the line of vision. This angle is to ensure that the light will travel from the monitor, reflecting about 10% and transmitting the remaining 90% of the incident light [11]. In Fig.5, the rays from the brightly lit monitor reflects from the pyramid panel and travels towards the viewer on the left. To a viewer who instinctively sees light as traveling in straight lines in open air, the object appears to be three-dimensional and occupying the middle of the pyramid.

To ensure the full illusion of the object is shown, each panel of the pyramid must be large enough to hold an entire image that is coming from the monitor. Because of this, the height of the pyramid needs to be at least the side length of each of the image's panels, as mentioned in section 5.1 to be 9'' (which is derived from our requirements). With this requirement, we use trigonometry to compute the rest of the dimensions as labeled A, B, and C in this picture. The most important aspect of this design is the inner angels of the isosceles trapezoid, which are computed to be 54˚ and 126˚, so that when all four panels come together, we will have the 45˚ tilt.

C. Hardware -- OV7670

Our design uses four OV7670s to capture side-views of the object at even, 90-degree angles. An Arduino Uno connects to all four cameras to control their color-scale using the SCCB protocol, which is compatible with the I2C protocol. Information about this interaction can be found in the following section on the Arduino Uno (5.D). The cameras also communicate video frame data to the Altera DE2-115 board using eight data pins and handshake signal lines, including

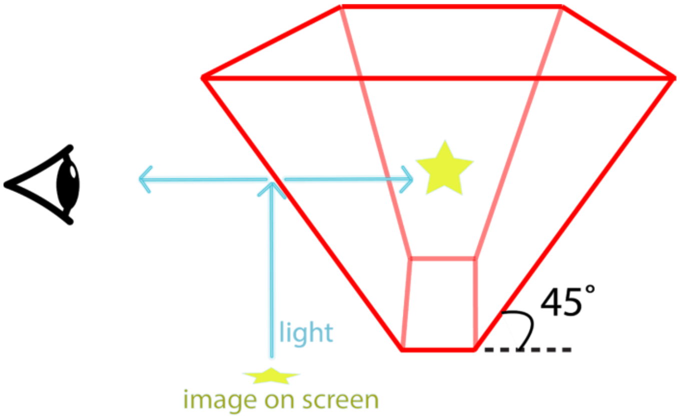


Fig. 5. Diagram showing how Peppers ghost illusion works in the pyramid

PLCK, HSYNC, HREF, and VSYNC. More information about these handshake signal lines can be found in Section 5.E.1.

D. Hardware -- Arduino Uno

An Arduino Uno is used to configure the OV7670 cameras. These cameras require setup and configuration with the I2C protocol to set the scale, color scale, and color matrix settings. The Arduino Uno connects to the camera's SCCB signals: the SCCB_E signal (serial chip select output), the SIO_C (serial bus control signal), SIO_D (serial bus data signal) and SCCB_E (serial bus enable/disable signal), and PWDN signal (power-down signal).

E. Hardware -- Altera DE2-115

The Altera DE2-115 is the center of our project, taking in the signals from the OV7670 camera, performing processing and filtering on those images, and outputting those signals over VGA to the holographic display. It has 6 internal modules that work together to handle this processing and handshaking as described below.

1) Image Decoder

Each OV7670 is connected to an image decoder, which handles the eight data pins, as well as the handshakes signals such as PLCLK, HSYNC, HREF, and VSYNC. Hence, there four decoders work in parallel to decode the data. This module will oversee retrieving data from the camera module. As can be seen in Fig. 6, signals VSYNC and HSYNC provide us with references about the location of a pixel data regarding a frame and the timing of its arrival. The image decoder will handle the different RGB formats used by the OV7670 since this camera supports RGB565, RGB555, and RGB444 with these 8 data pins. The image decoder then outputs serial streams of data corresponding to each camera to the first frame buffer, outputting enabling lines and completed lines as it goes to tell the FFB which cameras are having serial data written at any given time and when they are done.

2) Frame Buffer (FB)

The frame buffer technically consists of four buffers, one for each image signal processor module. This is to enable data pipelining and simultaneous processing of all four camera frames. Each frame buffer takes a serial stream of data from the image decoder and stores this data into BRAM to build the frame buffer. The FB module has 4 input data lines from the image decoder, 4 write enable lines from the decoder, and 4 output data lines to the image combiner.

3) Image Combiner

The role of the image combiner is to rotate images by appropriate multiples of 90 degrees. This will be done through selecting pixel data according to their relative position of the final frame from the RAM frame buffer to output a serial stream of bits. This data is being read and stored selectively through a FIFO queue protocol and handshake signals with combinational reads and synchronous writes. The image combiner will map consecutive bits of the final frame into pixel locations in the four camera frames, which are implicitly rotated in this process.

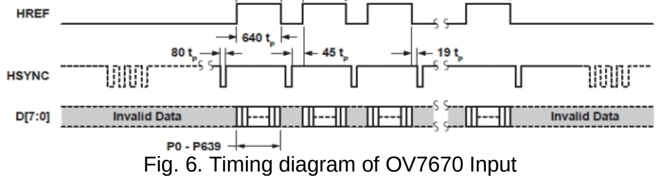


Fig. 6. Timing diagram of OV7670 Input

4) Image Signal Processor (ISP)

The ISP is designed to enhance the video quality going out to the display. It takes in signals from the Image Combiner for the serial data being read in. It will output a serial stream of data to the VGA protocol controller module. To sufficiently strengthen the holographic illusion and preserve details, we have one main image filter through which video frames are processed in the ISP module:

Chroma-key filter: Chroma keying is a technique to remove backgrounds from images by selectively removing all pixels that match the color of the background. Our background removal filter is designed to remove >95% of the background and <5% of the object. This will use a hard-coded background color value and threshold distance that determines how many colors close to the background color are filtered out. The background color will be the color used for the background of the live studio. This algorithm is shown below.

```
if(distance(pixel,target) < threshold) {
      pixel = 0;
}
```

The chroma-key filter also has corresponding hardware switches that allows the user to specify the target and threshold values, to determine which background color is removed and how sensitive the background removal is, respectively.

5) VGA Protocol Controller Module

The VGA Protocol oversees taking the pixel data and output from the ISP and outputs this pixel data along with the appropriate VGA timing protocol (HREF, VSYNC, HSYNC - Fig.6 and Fig.7). This data will be outputted directly into the VGA DAC, where the signals can be displayed on to the TV. Timing is synchronized between the ISP and the VGA Protocol Controller so that the output signal is delivered with the same rate that is needed for the display. It will take in a serial feed of data from the ISP as well as timing signals to let the combiner know when it needs new data to output.

6) Live Studio

A 28cm-by-28cm-base-and-30cm-tall (11”x11”x11.8”) live studio is used to capture side-view videos of the object. The size of the studio comes from the cameras: The OV7670 provides a 25˚ vertical field of view from the horizontal, so the camera must be placed at least 11cm (4.3”) away from the object to show the full size of the object. We want to leave a little room for the camera to stick through the cardboard size, hence we increased the distance between the object and the camera to be 14cm (5.5’’) to account for the length of the camera lens. This live studio is made of cardboard and uses a dark-green felt backdrop. At first, we thought that using a blue backdrop would reduce the color spill from the background onto the object. However, after experimenting with the different color backgrounds (mentioned in 4.F.), dark green is the best color for the camera's auto-exposure. This color background has the least color spill and improves detail preservation, which is critical to our presentation tool, fulfilling our frame quality requirement. LED lights are placed at the corners to ensure even lighting. The more uniformly lit the backdrop is, the better our chroma-keying filter can remove the background. Our live studio also includes accessories such as tweezers to enable seamless interaction with the object during presentations, without obscuring the object. Being the same color as the background, the tweezer minimizes obstruction of the object, while providing a nimble tool for the presenter to interact with the object.

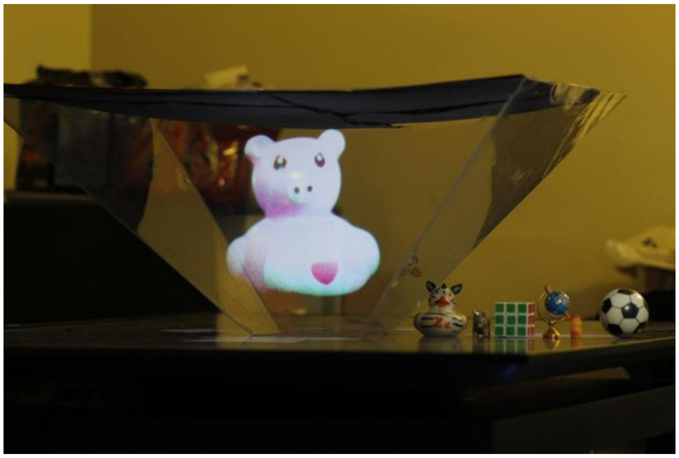


Fig.8. Enlargement test

VI. Testing and Validation

A. Enlargement

We required that our hologram enlarge objects by at least four times in our original requirements. After placing a variety of objects in the studio and measuring both their actual size and their apparent virtual size, we are seeing enlargements between three and five times, depending on where in the studio the object is placed. Specifically, if the object is placed at the center (where it is supposed to be placed for the illusion to work correctly), we observe an enlargement of four times (shown in Fig.8). While it is technically possible for our product to not quite meet this enlargement requirement if the object is placed too far from the camera (off-center), this also breaks the illusion as it is not how the product is intended to be used. We designed the studio such that the object should be located roughly in the middle, and our enlargement requirements are met under this stipulation. Thus, we consider this test to be a pass.

B. Real-time Latency

We required that our product have under 250ms of total latency from studio-to-display to ensure that any manipulations in the studio are quickly represented on the display (Fig.9). After repeating our latency test (outlined in II.C) multiple times with a high-speed camera, we were able to determine the average latency as 16 frames at 240fps, for a latency of 67ms. This is significantly lower than our latency requirement, and the lack of delay is very noticeable in our setup where changes feel instantaneous and responsive. Thus, because our actual latency is well below our requirement, we consider this test a strong pass.

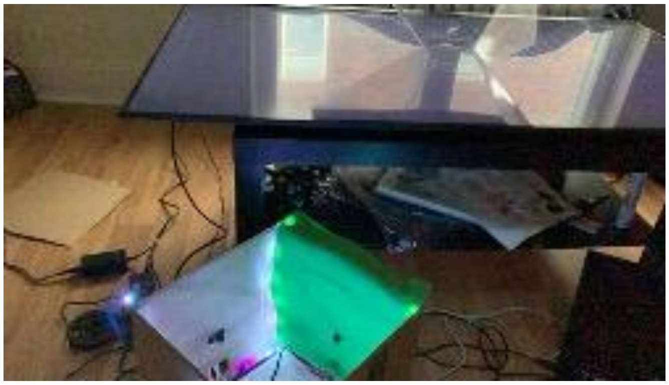


Fig.9. Real-time latency test with high-speed camera. This is taken from frame 3@240fps where the transition of the light turning on has not yet made it to the display

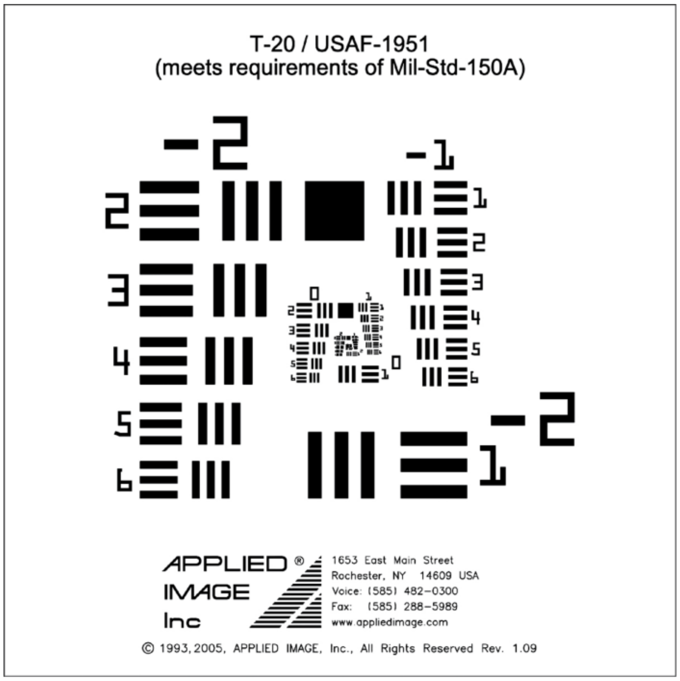


Fig. 10. MTF test card

C. Frame Rate Stability

In order for the frame rate to appear stable and no hitches or accumulating lag to arise over time in the use of our product, we required that our total processing delay be under 1/60th of a second, the industry-standard for framerate and the framerate we are using. When measured in synthesis, our product only requires one cycle per pixel processed by the FPGA as the filters, memory, and additional processing is all pipelined. This means that we have no accumulating lag whatsoever, and we are well below the 1/60th of a second of processing time requirement. We consider this test to also be a strong pass.

D. Image Sharpness

In order to ensure an optimal viewing experience for the user, we required that our final output image be photographically sharp (defined as detail-discernment being strictly resolution limited). This required that we pass an MTF test (test card shown in Fig.10) with the same score as a professional digital camera limited to the same resolution as our output. When this test was performed with an MTF test card (shown in Fig. 11), we were able to discern the lines in the “left-middle 6” at 8” away with the card printed on an 8.5”x11” sheet of paper. When we performed this test with a professional camera, we were able to discern the lines in the “right-middle 1” with the same distance and sheet size. Thus, the professional camera, when

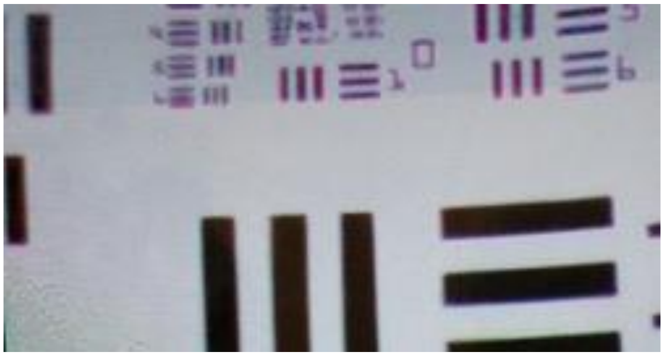


Fig.11. MTF test with a close-up crop of the test card showing the bottom-left 6 being the smallest part of the lines that can be discerned between.

resolution limited, was able to discern exactly one notch smaller on the test. While this does not technically meet our requirements, the 6 and the 1 are extremely close in size and the image does not appear to differ in quality between the two setups, this slight difference in detail-retention is only noticeable in side-by-side contrived tests. Thus, while the MTF test itself failed by one mark on the card, we consider the overall image sharpness test to be a pass.

E. Background Removal

To create the illusion of a floating object inside the pyramid, the background of the studio must be effectively removed from the images inside the FPGA. Furthermore, this must be done without removing the object itself, as this would create clipping around the object and break the illusion. Specifically, we required that, when the output was captured, at least 95% of the background must be removed and under 5% of the object must be removed to meet our requirements. When tested with a variety of objects, we found that between 95% and 99% of the background was being removed along with 1-5% of the object (Fig. 12). The reason for the variance noted here is that different objects have different colors, shades, and reflectiveness. Across all objects we were consistently meeting our requirements, however, in objects where we approached our 95%/5% threshold these imperfections did become noticeable in the output image, although they were minimal. We believe this could be improved by improving the studio lighting to have LEDs strictly across the top half of the studio with a greater spatial frequency than we built it to have. Overall, we consider this test to be a pass as we met our requirements and the output image has minimal issues, however improvements could be made to further our success here.

F. Lack of Distortion

In order to ensure that the final image was an accurate representation of the object in the studio, we required that the final image have minimal distortion. Specifically, it must have <5% change in angle throughout the image on all four sides of the pyramid. This was tested by projecting parallel lines inside the studio and measuring the distortion results in lightroom. This was the test we had the greatest difficulty with. When we first designed the full-scale pyramid, we were seeing extreme distortion as parts of the pyramid were sagging under its own weight due to the thin plexiglass used. We were able to mostly mitigate this by constructing a black top to the pyramid that held

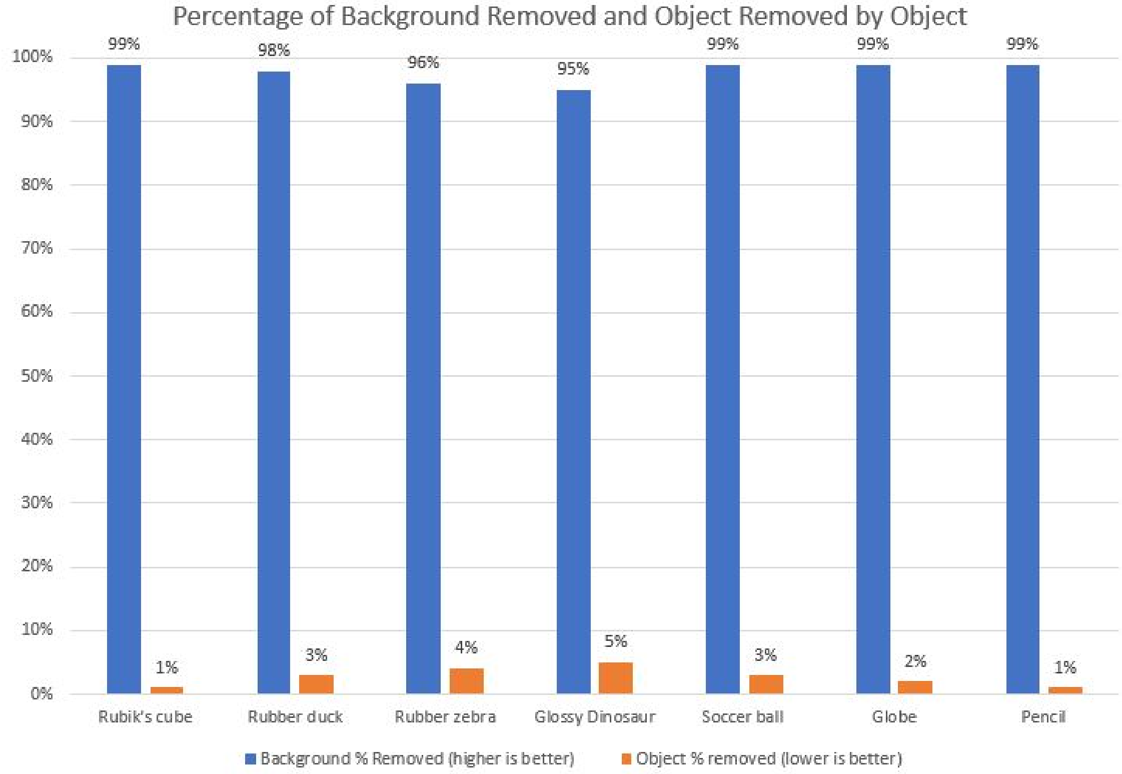


Fig. 12. Background removal test bar graph

all the sides together in place and took some of the strain off the plexiglass itself. Our results here diverge into two categories: results for the "average case" where the viewed object appears in the middle of the pyramid, and results for the "top case" where part of the object appears at the very top of the pyramid edges. In the average case, our results were ideal with distortion measurements consistently under 2%. This meets and exceeds our requirements, and results in no noticeable distortion for the typical viewing case. In the top case, however, distortion was visible with measurements ranging from 5-8% depending on the edge in question. This does not meet our requirement and does result in some minor distortion that is visible to the user if they view the product in this way. If the project were to be continued, we would have liked to put further effort into securing the pyramid to the top to reduce this top distortion. Unfortunately, we did not have time to implement this fix and our final product does have minor distortion when viewed in this way. Unfortunately, we do not consider this test to be a pass due to this distortion along the top, however, we would like to note that this is only visible when viewing certain objects from certain angles and does not significantly affect the image quality of the HoloPyramid as a whole, especially under typical viewing conditions.

VII. Project Management

A. Schedule

Our schedule is constructed so that the most important components were worked on at first, so that any issues that arises had sufficient time to be fixed. These critical tasks included the following: integrating the FPGAs with the camera, output video frames from the FPGA through VGA output, and image decoding and image combining, which includes movement of data into and out of SDRAM on the FPGA. We ordered our critical hardware components first (FPGA, cameras, display, pyramid materials) so that we could begin working on the MVP before getting to the image filtration.

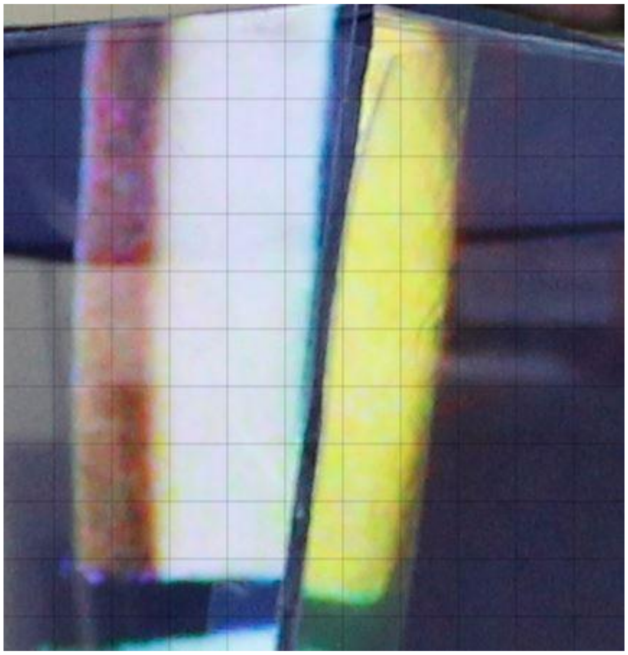


Fig.13. Distortion test, shown with the overlay grid from Lightroom so the percentage of distortion can be computed

From the design review report, we adjusted our schedule to give us additional time for implementing the FPGA modules such as the image combiner due to these tasks taking longer than expected. We also needed substantial time to debug color scheme issues with the camera input into the FPGA. We also spent more than one week on the plexiglass pyramid in order to construct multiple pyramids as well as prototypes.

Following the completion of our MVP by the interim demo on April 12th, we focused on improving our project and image quality, through writing and adapting the image filters as well as integrating and testing our FPGA.

B. Team Member Responsibilities

Breyden Wood

- Image rotation and combining
- PLL with VGA output
- Testing of image quality in Lightroom

Jullia Tran

- PLL with Camera interface with FPGA
- Image decoding and frame buffer
- Integration of FPGA

Grace An

- Building the pyramid and live studio
- Designing and building chroma-key filter on FPGA

C. Budget

As shown in Table of Cost Breakdown, our largest hardware components--the Altera DE2-115 and HDTV--are preexisting components. Added to the cost of the ~5$ cameras, the $55 HSMC expansion board, $20 VGA-to-HDMI adapter, and $35 for plexiglass and other building materials, our final product comes out to approximately $178.98 in price of materials used. Parts of the remaining portion of our $600 budget were used for issues that arose, such as the OV7670 cameras quality ranges (one of them did not turn on out of the box, some had better autoexposure sensors than others, etc). We also used some of the budget to prototype the plexiglass pyramid. We also experimented with different background colors and materials. With prototyping cost accounted for, we spent a total of $422.93 from our budget.

D. Risk Management

We had an extensive risk management plan focused on providing fallback plans to compensate for physical constraints such as number of logical elements or limited RAM on the FPGA board that we are using. We were able to:

1. *Reduce video frame quality*: We did not significantly struggle with the timing requirement, and/or logic element number on the Altera DE2-115 board, but we used some of this risk management plan. We struggled with the memory constraint but this issue was resolved by only storing active pixels, which significantly reduced the memory needed to be

stored (mentioned in section 4.E). We did not reduce color-scale [12].

We reduced the 480px640p camera input to a 240px240p camera input by only storing the active frames needed that covers the exact region that the object is present in. This reduction is simply a crop of the camera input, so that the four panels of the display would show up as a square. Hence, the resolution of the video frame is still maintained as 720p.. We did not reduce the VGA output from 1280px720p to 640px480p, as this was not needed. We then used a VGA-to-HDMI adapter and an HDMI upscaler to maintain video frame quality at this lower resolution. We did reduce the initial planned speed of 60 frames per second down to 30 by reducing the clock speed. However, this change does not affect our frame-rate stability requirement, as shown in section 5.C that our project passes the test for this requirement..

2. *Reduce pyramid size*: We did not need to reduce the pyramid size to meet the following requirements: The pyramid needs to be stiff (able to hold the pyramid shape at exact dimensions and angles), transparent (for high reflection quality), and thin (to avoid doubled illusions).

3. *Remove image filters*: We removed the sharpness and brightness image filters we initially planned to do. We did not remove the chroma-keying filter, which we could have by using a black background in the live studio, so that the background is automatically not projected onto the hologram. We removed the brightness and sharpness filters by relying on the in-built filters in the cameras and TV, which were sufficient for our frame quality requirement. This choice mitigated the complexity of sophisticated memory management. .

VIII. Ethical Issues

The main ethical issue is that the HoloPyramid is not an accessible presentation tool. By its nature, it requires a level of sight, and it excludes those with limited vision. Additionally, it is also stationary at a certain level of height. For wheelchair users and those otherwise impaired in movement, the HoloPyramid is especially exclusionary compared to other presentation tools. For people whose job is to report on presentations (such as reporters), the HoloPyramid could especially poorly impact them as a result of its exclusionary nature. To mitigate this ethical issue, if we were providing the HoloPyramid to users as a presentation tool, we would likewise provide the following advice: the presenter should provide additional accessibility tools for their presence to make up for the presence of an exclusionary presentation tool.

IX. Related Work

Similar products are available in the market. As stated in the introduction, the inspiration of our design comes from the hologram pyramid that is for smart-phones made out of plastic (Fig.1). Although this product is similar to our project, it is not meant for enlargement and use as an interactive presentation tool. Instead, it's meant for entertainment purposes and needs pre-baked or rendered videos to create the hologram illusion. Another product that is also in the market is a more professional hologram display meant for exhibition displays (Fig.11). While this product is at a scale similar to our project, this product only has three sides, doesn't allow the presenter to interact with the displayed object, and costs up to $5000 a piece [16]. Our product is less expensive and also allows a real-time video feed from the pyramid.

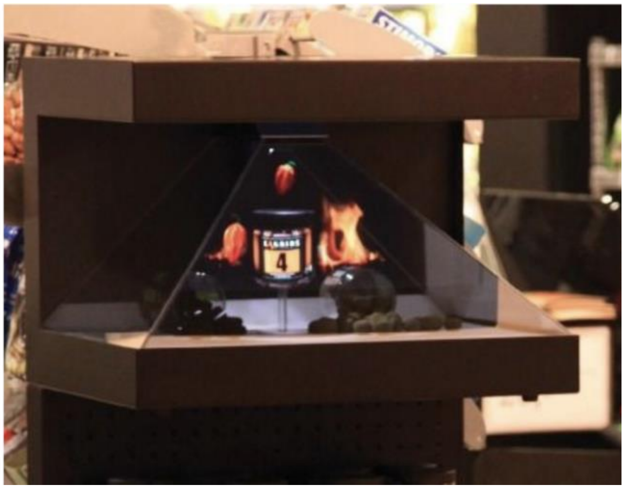


Fig.14. Hologram pyramid display for exhibitions

X. Summary

We achieved our initial design requirements as set out in our design review document, achieving the required enlargement, real-time latency, frame rate stability, image sharpness, and lack of distortion. Our project successfully displays a holographic illusion consisting of four real-time feeds that represent side views of an object in a live studio. Our project only displays objects that are non-reflective because the cameras do not capture good video feeds of reflective objects. Our project is also unable to display objects with the same color as the background, as is a common issue with chroma-key based background removal.

A. *Future work*

We do not plan to work on this project further following this semester, but we would have worked on and/or used the following if we had the time and/or budget:

1) *Improve chroma-key filter to be hue-based*

If there was more time for this project, we would like to change the current chroma-key filter from removing a specific color and some nearby colors within a range, to targeting the hue of that color. Currently, our algorithm computes the color distance across the three red, green, blue channels and removes this pixel if this color is within the target color's range. Our improvement would be to improve the removal of neighboring colors through having a more sophisticated algorithm to compute nearby hues. This would require significant computation in the FPGA as hue is more complex to calculate than color distance.

2) *Newer FPGA*

Budget-permitted, we would like to work with a newer FPGA. Newer versions of the FPGA have more available BRAM on the device. This would allow us to increase the resolution from 720p at 30fps to 1440p at 60fps.

In order to make this adjustment, we would also need to change our display output protocol from VGA to HDMI because of bandwidth bottlenecks from the VGA protocol. These changes would help improve frame quality, which would enhance the quality of the illusion and excel our current frame quality requirement.

3) *Rebuild pyramid*

If we had more time, we would build multiple full-size pyramids out of plexiglass of different thicknesses. This would help us determine which plexiglass thickness would best provide a solid shape and minimize double reflections and improve our project's distortion and illusion requirements.

B. *Lessons Learned*

One of the lessons we learned from this project was that documentation is often poorly written and inaccurate. We ran into this problem when dealing with the timing for the OV7670 cameras and when trying to implement the protocol that these cameras use. The issue we ran into was with the SCCB protocol. This protocol is similar to I2C, however, it is not a popular protocol that one could easily find the documentation for this protocol. Furthermore, the clock specified in the documentation for this protocol is supposed to be between 100kHz-400kHz. Yet, when the camera was given a clock input in this range, the SCCB protocol did not work, which broke the color scheme and made the decoding module not work. Instead, when supplied with a clock of around 5KHz, the camera settings immediately worked and the decoding parts worked out seamlessly. We were able to decode the video feed with a much better quality, along with being able to set other image settings such as auto exposure and scale.

Along with this, we also learned that with cheaper cameras, the quality control of this product is not always the best. When buying these cameras, some of them came out of the box broken and didn't turn on during initial boot up. Because of this, it was really helpful to devote a sizable amount of our budget into buying extra parts so that we can easily switch between the extras to find ones that were made with better quality. The sensors for autoexposure in each of these cameras weren't made with the same quality also. Hence, it was very useful to buy multiple cameras as a buffer during our build.

Another lesson we learned was how to use an oscilloscope as a debugging tool. This was especially useful when working with checking our inputs and outputs to the camera settings. When learning about Verilog in previous classes such as 18-240 and 18-341, we have tools such as waveforms to debug. However, when dealing with real signals, waveforms can't be used because this tool is only available for simulation. With synthesis, the oscilloscope serves as a replacement for this simulation waveform tool. It was extremely useful and aided our debugging process, helping us to debug color and camera settings issues faster.

Through creating this project, we also learned more information about photography: We learned how background removal works and how uneven lighting worsens background removal. We learned about how the MTF score provides a quantitative way to measure image sharpness. We also learned about how background colors can cause color spill, and how to manage auto-exposure on a camera.

Glossary of Acronyms

ASIC – Application-Specific Integrated Circuit
BRAM - Embedded Random Access Memory
CPU – Central processing unit
DAC – Digital-to-Analog Converter
FIFO – First-In, First-Out
FPGA – Field-Programmable Gate Array
FPS – Frame Per Second
GPIO – General-Purpose Input/Output
HDMI – High-Definition Multimedia Interface
HDTV – High-Definition Television
HSMC – High-Speed Mezzanine Connector
I2C – Inter-Integrated Circuit
LED – Light-Emitting Diode
MB – MegaByte
MTF – Modulation Transfer Function
MVP – Minimum Viable Product
NTSC – National Television Standards Committee
PC – Personal Computer
PLL – Phase-Locked Loop
RAM – Random Access Memory
SCCB – Serial Camera Control Bus
SDRAM – Synchronous Dynamic Random Access Memory
SRAM – Static Random Access Memory
USB – Universal Serial Bus
VGA – Video Graphics Array

References

[1] Artstation, https://www.artstation.com/artwork/lVwgAa

[2] Jain, A., Bansal, R., Kumar, A., & Singh, K. (2015, May). A comparative study of visual and auditory reaction times on the basis of gender and physical activity levels of medical first year students. Retrieved March 15, 2021, from https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4456887/

[3] Jeremy Brooker. 2007. The Polytechnic Ghost. Early Popular Visual Culture 5, 2 (2007), 189–206.

[4] Nick Schlüter, Franz Faul; Are optical distortions used as a cue for material properties of thick transparent objects?. *Journal of Vision* 2014;14(14):2. doi: https://doi.org/10.1167/14.14.2.

[5] Synowicki, Ron. "Transparent Substrates: How To Suppress Reflections from the Back Surface." *Https://Www.jawjapan.com/Document/Public/000/General_01_Transparent%20Substrates%20Suppressing%20Backside%20Reflections.Pdf*, Apr. 2010.

[6] Arkema Group. *Plexiglas Data Sheet*, Altuglas International, 2019, www.plexiglas.com/export/sites/plexiglas/.content/medias/downloads/sheet-docs/plexiglas-g.pdf.

[7] Apple Computer Corporation, *Index of Refraction*, 2000, hyperphysics.phy-astr.gsu.edu/hbase/Tables/indrf.html.

[8] Engineering ToolBox. *Solids and Metals - Specific Gravities*, 2003 https://www.engineeringtoolbox.com/specific-gravity-solids-metals-d_293.html

[9] Engineering ToolBox. *Solids and Metals - Specific Gravities*, 2003 https://www.engineeringtoolbox.com/young-modulus-d_417.html

[10] Li, Yan, and Shuxia Ren. "Plate Glass." *Plate Glass - an Overview | ScienceDirect Topics - Building Decorative Materials*, 2011, www.sciencedirect.com/topics/chemistry/plate-glass.

[11] Greenslade Jr, T. B. (2011). Pepper's Ghost. The Physics Teacher, 49(6), 338-339.

[12] Omnivision, "OV7670/OV7171 CMOS VGA (640x480) CameraChip Sensor with OmniPixel Technology", Aug. 2006

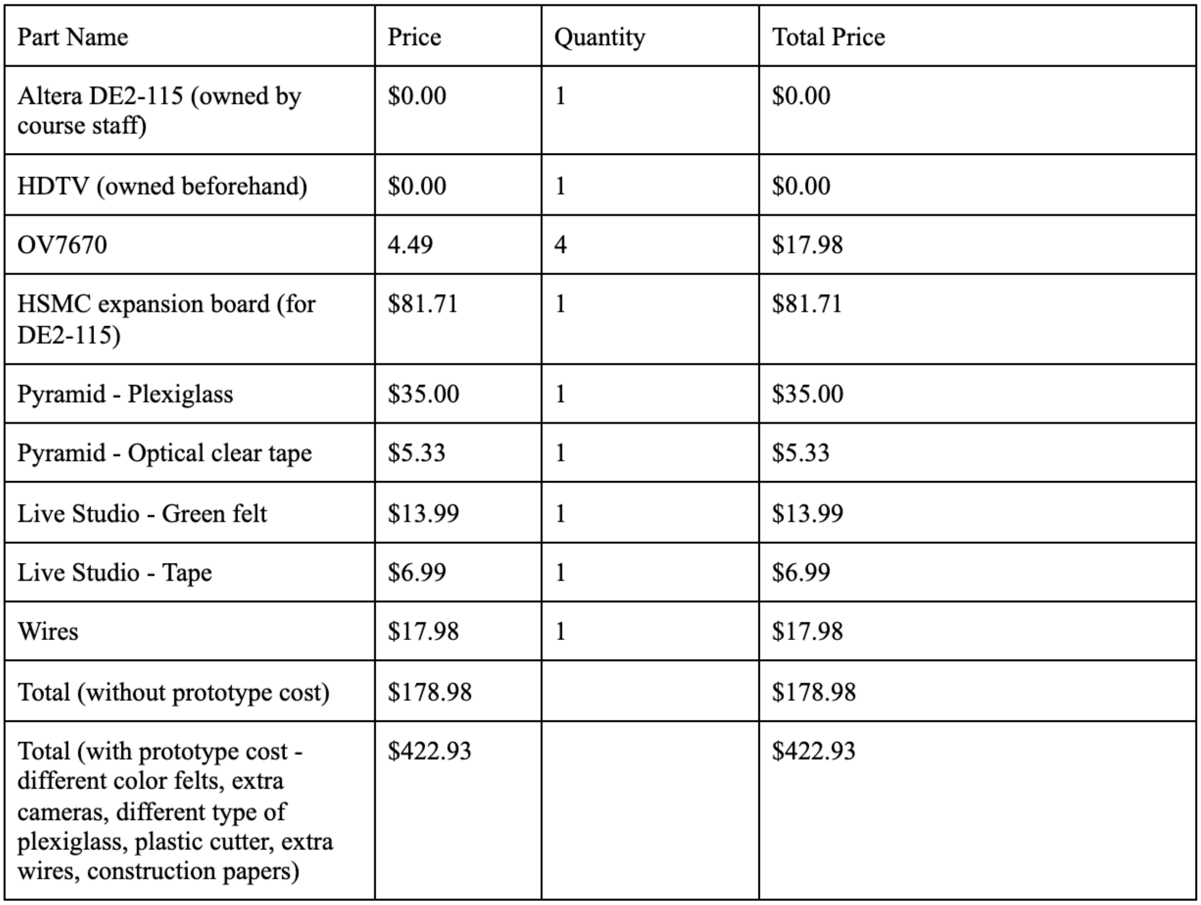
[13] Abtahi, Tahmid & Kulkarni, Amey & Mohsenin, Tinoosh. (2017). Accelerating convolutional neural network with FFT on tiny cores. 1-4. 10.1109/ISCAS.2017.8050588.

[14] Lens MTF test results. (n.d.). Retrieved March 15, 2021, from https://www.the-digital-picture.com/Help/MTF.aspx

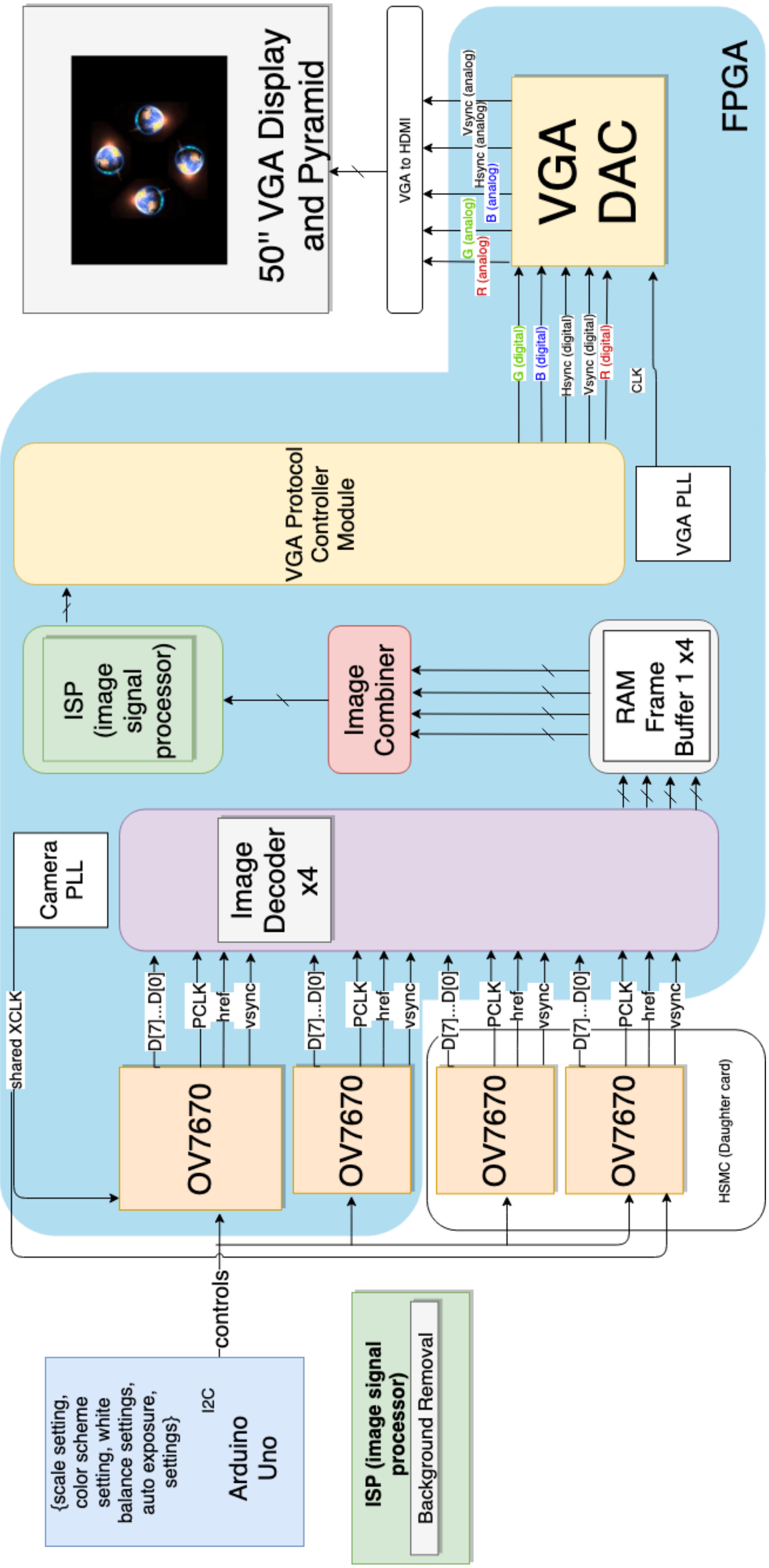
[15] Applied Image, "T-20-P-OP", 2021, https://www.appliedimage.com/product/t-20-p-op/

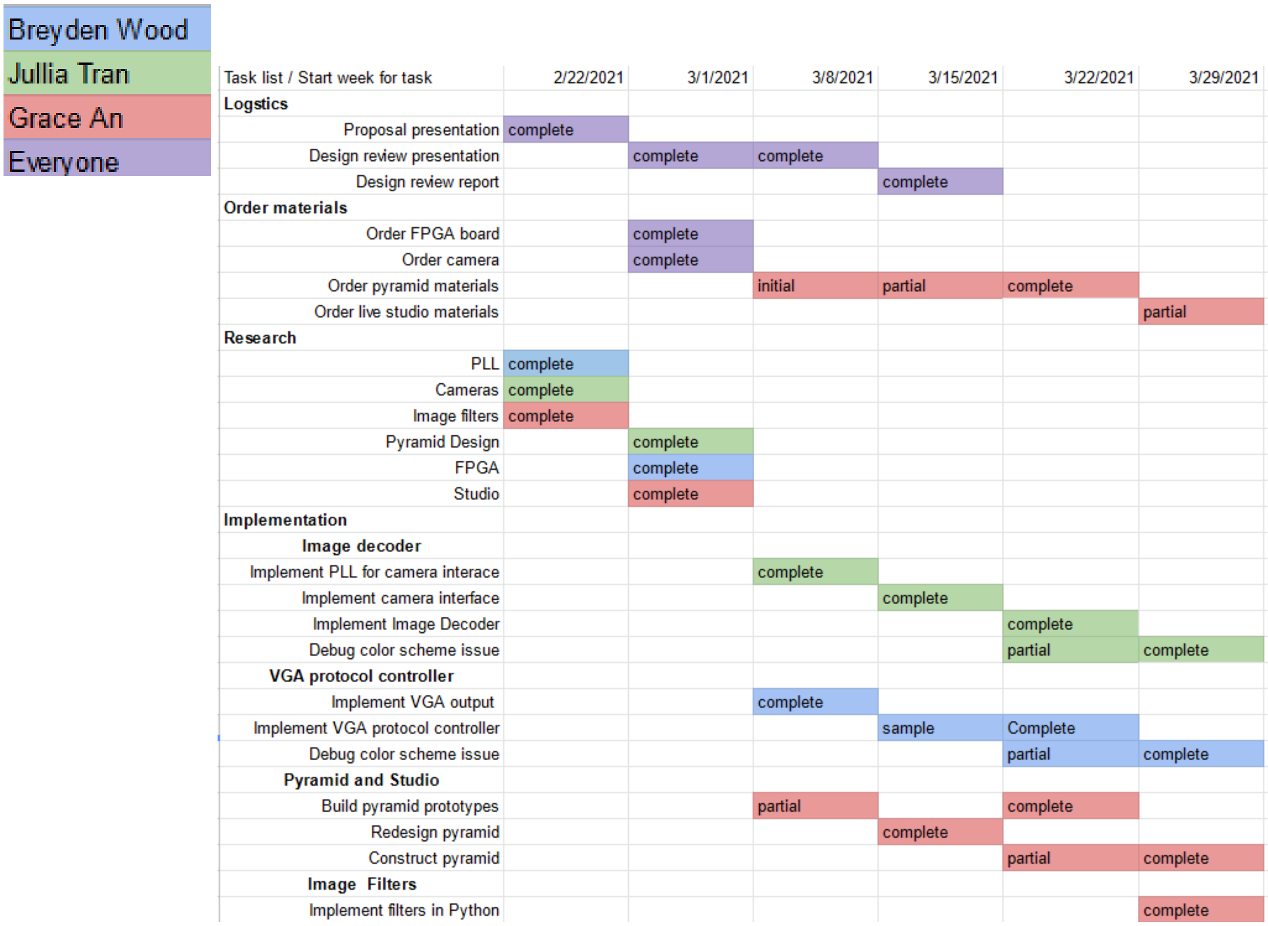
[16] "3D Hologram Pyramid." *3 Side 42 Inch 3D Hologram Pyramid*, Global Sources , www.globalsources.com/3D-holographic/3D-Hologram-Pyramid-1137623471p.htm#1137623471.

Table of Cost Breakdown



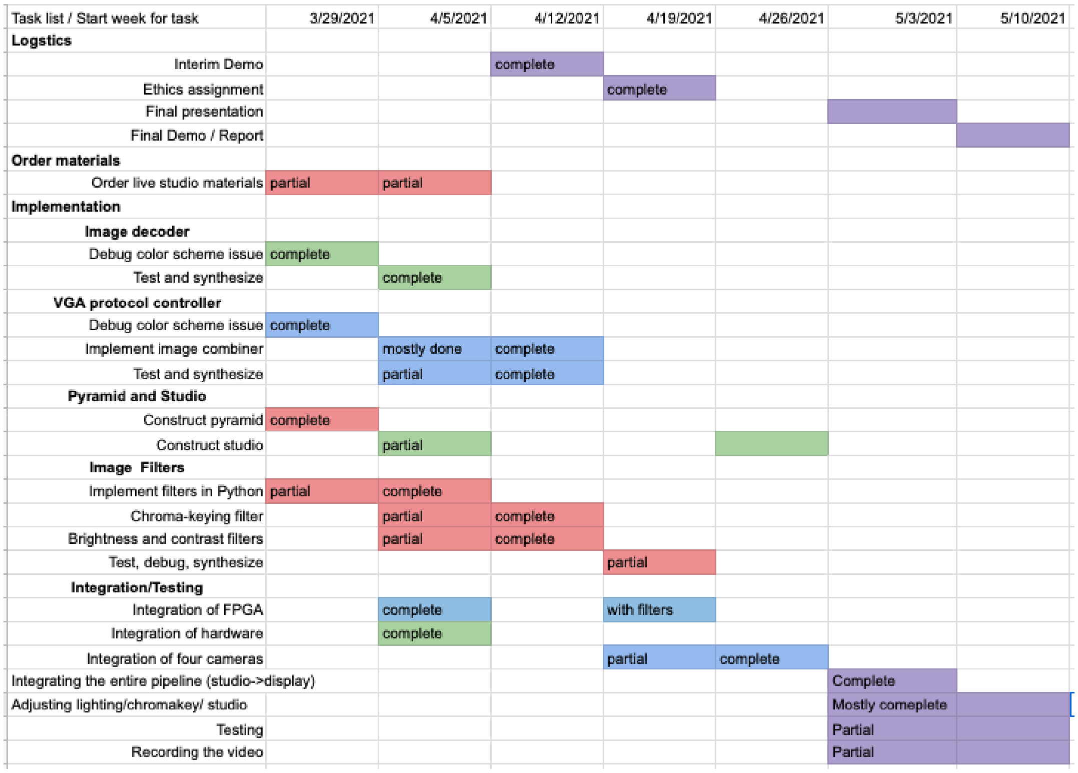
Part Name	Price	Quantity	Total Price
Altera DE2-115 (owned by course staff)	$0.00	1	$0.00
HDTV (owned beforehand)	$0.00	1	$0.00
OV7670	4.49	4	$17.98
HSMC expansion board (for DE2-115)	$81.71	1	$81.71
Pyramid - Plexiglass	$35.00	1	$35.00
Pyramid - Optical clear tape	$5.33	1	$5.33
Live Studio - Green felt	$13.99	1	$13.99
Live Studio - Tape	$6.99	1	$6.99
Wires	$17.98	1	$17.98
Total (without prototype cost)	$178.98		$178.98
Total (with prototype cost - different color felts, extra cameras, different type of plexiglass, plastic cutter, extra wires, construction papers)	$422.93		$422.93


50" VGA Display and Pyramid
VGA to HDMI
FPGA
VGA DAC
Vsync (analog)
Hsync (analog)
B (analog)
G (analog)
R (analog)
G (digital)
B (digital)
Hsync (digital)
Vsync (digital)
R (digital)
CLK
VGA PLL
VGA Protocol Controller Module
ISP (image signal processor)
Image Combiner
RAM Frame Buffer 1 x4
Camera PLL
Image Decoder x4
shared XCLK
D[7]...D[0]
PCLK
href
vsync
OV7670
OV7670
OV7670
OV7670
HSMC (Daughter card)
controls
I2C
{scale setting, color scheme setting, white balance settings, auto exposure, settings}
Arduino Uno
ISP (image signal processor)
Background Removal



🟦	Breyden Wood
🟩	Jullia Tran
🟥	Grace An
🟪	Everyone

Task list / Start week for task	2/22/2021	3/1/2021	3/8/2021	3/15/2021	3/22/2021	3/29/2021
Logstics						
Proposal presentation	🟪 complete					
Design review presentation		🟪 complete	🟪 complete			
Design review report				🟪 complete		
Order materials						
Order FPGA board		🟪 complete				
Order camera		🟪 complete				
Order pyramid materials			🟥 initial	🟥 partial	🟥 complete	
Order live studio materials						🟥 partial
Research						
PLL	🟦 complete					
Cameras	🟩 complete					
Image filters	🟥 complete					
Pyramid Design		🟩 complete				
FPGA		🟦 complete				
Studio		🟥 complete				
Implementation						
Image decoder						
Implement PLL for camera interace			🟩 complete			
Implement camera interface				🟩 complete		
Implement Image Decoder					🟩 complete	
Debug color scheme issue					🟩 partial	🟩 complete
VGA protocol controller						
Implement VGA output			🟦 complete			
Implement VGA protocol controller				🟦 sample	🟦 Complete	
Debug color scheme issue					🟦 partial	🟦 complete
Pyramid and Studio						
Build pyramid prototypes			🟥 partial		🟥 complete	
Redesign pyramid				🟥 complete		
Construct pyramid					🟥 partial	🟥 complete
Image Filters						
Implement filters in Python						🟥 complete



Task list / Start week for task	3/29/2021	4/5/2021	4/12/2021	4/19/2021	4/26/2021	5/3/2021	5/10/2021
Logstics							
Interim Demo			🟪 complete				
Ethics assignment				🟪 complete			
Final presentation						🟪	
Final Demo / Report							🟪
Order materials							
Order live studio materials	🟥 partial	🟥 partial					
Implementation							
Image decoder							
Debug color scheme issue	🟩 complete						
Test and synthesize		🟩 complete					
VGA protocol controller							
Debug color scheme issue	🟦 complete						
Implement image combiner		🟦 mostly done	🟦 complete				
Test and synthesize		🟦 partial	🟦 complete				
Pyramid and Studio							
Construct pyramid	🟥 complete						
Construct studio		🟩 partial			🟩		
Image Filters							
Implement filters in Python	🟥 partial	🟥 complete					
Chroma-keying filter		🟥 partial	🟥 complete				
Brightness and contrast filters		🟥 partial	🟥 complete				
Test, debug, synthesize				🟥 partial			
Integration/Testing							
Integration of FPGA		🟦 complete		🟦 with filters			
Integration of hardware		🟩 complete					
Integration of four cameras				🟦 partial	🟦 complete		
Integrating the entire pipeline (studio->display)						🟪 Complete	
Adjusting lighting/chromakey/ studio						🟪 Mostly comeplete	🟪
Testing						🟪 Partial	🟪
Recording the video						🟪 Partial	🟪