University of Rochester

Design Description Document

Digital Cloaking Instrument

Team
Johana Escudero, Xi Zhou, Shenghan Gao

Client
Curtis Broadbent, Physics and Astronomy

Faculty Advisor
John Howell

Document Number 1

Revisions Level	Date
D	Feb 27 2017

This is a computer generated document. The electronic master is the official revision. Paper copies are for reference only. Paper copies may be authenticated for specifically stated purposes in the authentication block.

Authentication Block

Rev	Description	Date
A	Initial Document	January 23rd 2017
B	First update finalizing a demo goal	February 6th 2017
C	Reorganizing timeline, providing interlacing code updates, adding previous designs into an appendix	February 20th 2017
D	Midterm DDD; reorganizing timeline, adding interlaced image updates, adding risk assessment and transition plan, updating table of content.	February 27th 2017
E	Matlab details, CAD specifications, and a timeline update	April 2nd 2017
F	Revised spec, Description, Results	April 24th 2017
G	Final Revision	May 4th 2017

Table of Contents

1. Design Description

A cloaking system has many applications from conserving natural sceneries in urban environments to disguise objects. Our customer wants to show the potential of a digital cloaking system using commercially available lenslet arrays and three Xperia Z5 cell-phones as displays. An ideal cloaking display should work in real-time from far viewing distance over a wide range of viewing angles, which would require a large high resolution display. Since there is no commercially available display meeting these criteria, our goal is to use 4K resolution cell-phone screens as our displays. The challenge with using small screens is maintaining a high angular resolution and wide range of viewing angles, these tradeoffs are addressed throughout this design description. The final cloaking system is aimed to work from a viewing distance of 5< meters (determined by the resolution of the human eye) with a desired composite image containing enough views to cover ideally ±15 degrees horizontally (determined by the lenslet NA). In addition to the optical challenges, our customer would also like us to write a custom composite image algorithm for our system that is faster than their previous recorded ~15 minutes per composite image.

We are responsible for the following:

1. Cloaking System Specifications: Choosing the appropriate lenslet array, display, viewing distance, and number of views.
2. Image capturing: Creating a method for capturing angular information quickly using a single camera.
3. Display-to-Lenslet Distance: Optimizing the distance between the display and the lenslets and adhering the chosen lenslet array to a display.
4. Altering the Matlab code for interlacing an iPad mini to work with a Z5 display, and optimizing the image processing routine.
5. Demo: Assembling the parts together in a way that communicates what this technology is capable of on a larger scale by creating a demo that will be shown during Hajm School of Engineering and Applied Sciences Design Day.

2. Background

A digital integral cloak consists of three main subsystems:

1) Image capturing: An array of CCD cameras to capture the scene behind the cloaked region from many perspectives.

2) Image processing: The images are processed to create a composite image.

3) Display: A custom display is made from an extremely high-resolution optical display covered by a lenticular array.

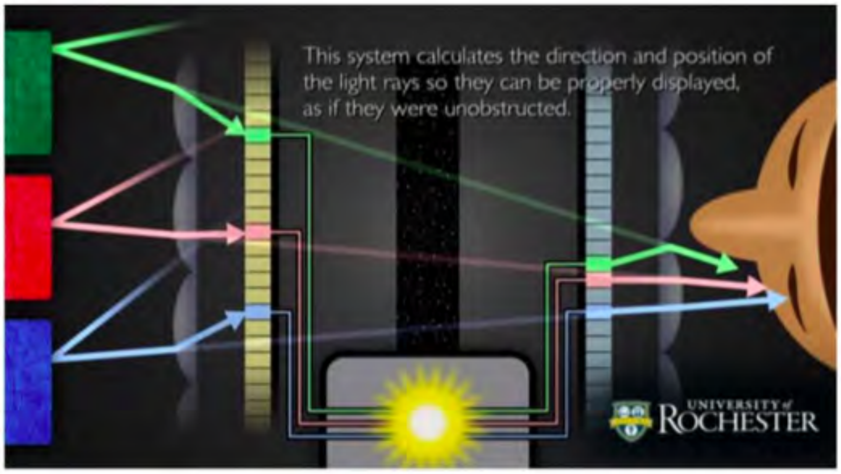
This system calculates the direction and position of the light rays so they can be properly displayed, as if they were unobstructed.
UNIVERSITY of ROCHESTER

Figure 1. Digital cloaking architecture. Left to right: image capturing system, image processing, and a display covered by a lenticular array. The lenslet array makes it appear like the rays from the object have passed the obtrusion (screen) undeviated. Image credit: Frame taken from University promotional video produced by Matt Mann.

A believable scene should change with a viewer's perspective. To do this we can use a lenslets array placed in front of the display (displaying a composite image) that directs angular information to different viewing angles. The display is placed at the focal plane of the lenslet array, making the light from each pixel approximately collimated such that the pixels looks to be the size of a lenslet. Each lenslet then becomes a new "pixel" called a superpixel, shown in Figure 2.

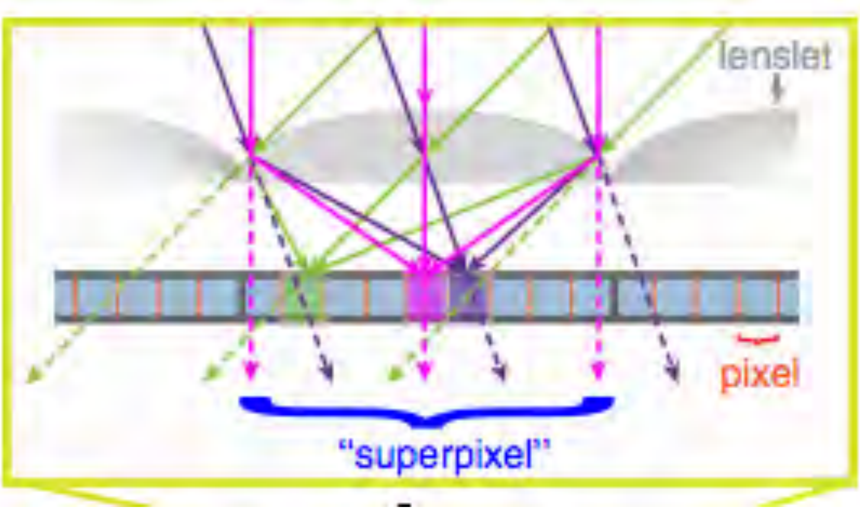

Figure 2. Detailed view of the detector array using a lenslet-based detector. Reference: Joseph S. Choi and John C. Howell, "Digital integral cloaking," Optica 3, 536-540 (2016)

3. Demo Specifications and Approach

This demo's purpose is to show the principles behind digital cloaking, and to demonstrates the potential of a larger cloaking display. The demo consists of a board with rectangular cell-phone-sized holes. Most of the holes will be empty, showing a scene in the background, but three of the hole will have a Z5 phones with their respective lenslet array. The goal of the demo is to trick people into not seeing the phones.

3.1 Resolution vs. Number of Views

Basing on normal human eye angular resolution of 1 arc minute, the number of views required for near perfect cloaking within a 30° range (approximate, lenticular NA was unspecified) is:

$$\frac{30\ degrees}{1/60\ \ degrees} = 1800\ views$$

To meet this resolution, we would need a large display that would accommodate large superpixels; A larger superpixel contains more pixels, which contains the angular information (# of views). Using a smaller screen introduces a tradeoff between views and resolution, where the views are defined by the number of pixels under a lenslet and the resolution is defined by the number of superpixels on the display.

Using the size and pixel per inch (ppi) of the Z5, the number of views can be calculated the following way:

$$\frac{(806\, ppi)}{(\, lpi)} \approx \; views$$

where 806 ppi is a Z5 specification and the lpi depends on the lenslet.

This equation shows 20 lpi lenslet will give 40 views and 40 lpi gives 20 views. In the appendix, there is a method for obtaining a 6x increase in the horizontal resolution which will gives 6 times more views without sacrificing resolution.

3.2 Cell Phone Frame

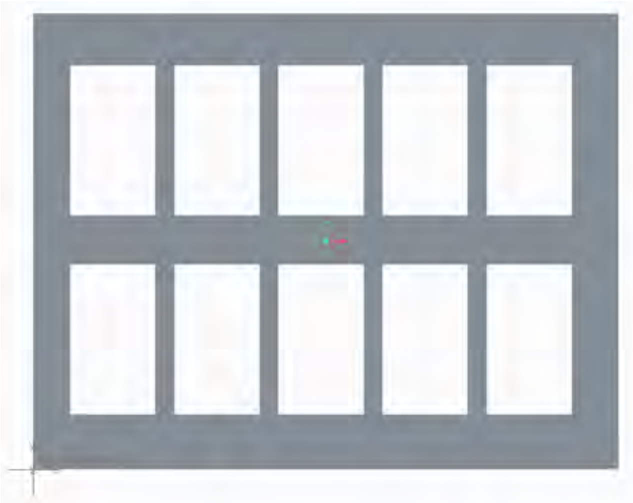

Figure 3. CAD Model of a frame containing cell-phone sized holes.

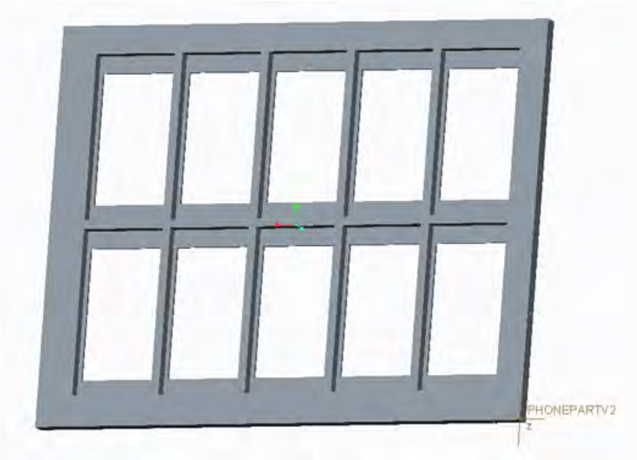

Figure 4. Back view of the frame. Each hole is cut to fit and hold a cell-phone, but only one of three holes will contain a cloaked phone display.

Specifications

- 10 holes
- Hole dimensions: 154.4 mm, 76 mm wide, and 12 mm thick
- Phone thickness = 7.6 mm
- Lenslet thickness = 4 mm

3.3 Lpi Dependence Analysis

Each lens in a lenslet array acts as a superpixel, under which many smaller pixels collimate light out in different directions. Each superpixel contains the angular information of a scene, and the number of pixels under the superpixel determines the number of views available. So for more views (which is desired, since we need it to work over the angular range defined by the lenslet NA) we need superpixels (lenslets) large enough to contain a high number of pixels.

Both 20 lpi and 40 lpi lenslets are available from our customer. The specifications for cloaking with 20 lpi, 40 lpi, and a comparison between the two densities follows.

20 lpi lenslet array

- 20 lpi = 1.27 mm/lens
- Viewing distance: 20 lpi is good for 4.37 m away
- Angular range: ± 15 degrees
- Number of views: the number of view we will get from the lenslet array will be 40 views:

$$\frac{806\text{ ppi}}{20\text{ lpi}} \approx 40\ views$$

- Super-pixels: with the 20 LPI lenslet array, the number of super-pixels is equal to 787 super pixels/m.

40 lpi lenslet array

- 40 lpi = 0.635 mm/lens
- Viewing distance: 20 LPI is good for 2.19 m away
- Angular range: ± 15 degrees
- Number of views: the number of view we will get from the lenslet array will be 20 views:

$$\frac{806\text{ ppi}}{40\text{ lpi}} \approx 20\ views$$

- Super-pixels: with the 40 LPI lenslet array, the number of super-pixels is equal to 1574 super pixels/m.

Comparison: 20 lpi and 40 lpi

	20 lpi	40 lpi
Viewing distance	4.37 m away	2.19 m away
Angular range	± 15 degrees	± 15 degrees
Number of views	40 views	20 views
Super-pixels	787 super-pixel	1574 super-pixel

* Angular range is defined by the NA of the lenslet, but is approximated to be 30 degrees.

For our senior design demo we chose to work with the lpi that provided the best cloaking illusion. To choose we tested both lenslets, and ultimately decided to show the behavior of both during our demo by having two phones with a 40 lpi array and the other with a 20 LPI.

3.4 Vertical Image Processing

Our customer provided Mathematica code intended for an iPad Mini under a 20 LPI lenslets for us to use as a starting point. For our new algorithm, our goals were to:

- Reconstruct a composite image from multiple frames
- Extract only the required number of frames from a video of any length
- Regroup RGB if a higher horizontal resolution is desired

To increase the image processing efficiency our team moved the algorithm from Mathematica to Matlab, which greatly improved the performance from ~15 minutes to 300 seconds per composite image. Since improving the speed of the image processing routine was also desired, after meeting with our customer and faculty advisor, we decided to make our Z5 image processing code work with vertical lenses rather than slant lenses, trading resolution for speed. This means we do not get the 6x resolution along the horizontal direction (as discussed in Appendix D), but the speed improved to ~10 seconds per composite image getting us closer to a real-time composite image.

The algorithm approach is:

1. Input video of a scene being captured left to right by moving a camera linearly across the scene at a constant speed. The length of the video does not matter as long as the camera captures the same range (which should be approximately 30 degrees) each time.
2. The algorithm then selects n-number of frames in equal steps, where n is the total number of lenslets that can fit on the display.
3. Each frame is linearly compressed under a lenslet. How much a frame is compressed depends on how many pixels fits under a lenslet, which is why a larger superpixel contains more angular information.

4. To compress each picture correctly, it is important to know how many pixels are in the frame and how many pixels the frame will be compressed into. For example, if a superpixel contains 10-pixels across, the algorithm selects 10 columns from the frame. Knowing the total number of columns in the original frame allows each selected column to come from equally spaced portions of the original image.
5. In the case when a picture needs to be resized vertically as well, a similar process to step 4 is followed.
6. Frame 1 is compressed to fit under the first lenslet, frame 2 under the second, and this repeats until each lenslet contains a compressed frame.

The algorithm is provided in Appendix C.

4. Results

4.1 Cell Phone Frame Designed on CREO

Total cost for board: $275

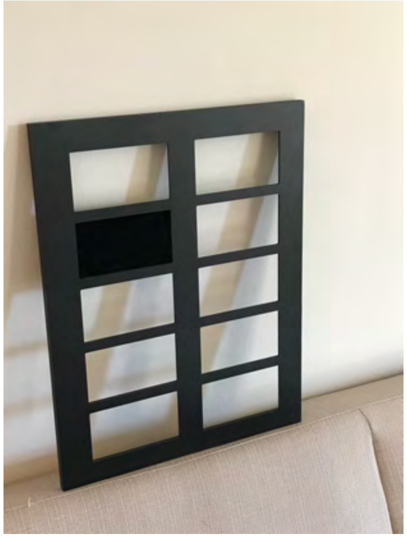

Figure 5. Frame designed on CREO, with an Xperia Z5 phone inside in an upper left hole.

The phones fit securely in each hole with a lenslet underneath. We removed portions of the board to leave room for the phones to be connected to their charger while in the screen, and we also removed some portions of the sides to relieve pressure from the sounds/on/off buttons.

4.2 Composite Images for 40 and 20 lpi Lenslet Arrays

These are the composite results from following the algorithm approach outline in section 3.4 using an artificial 3D video.

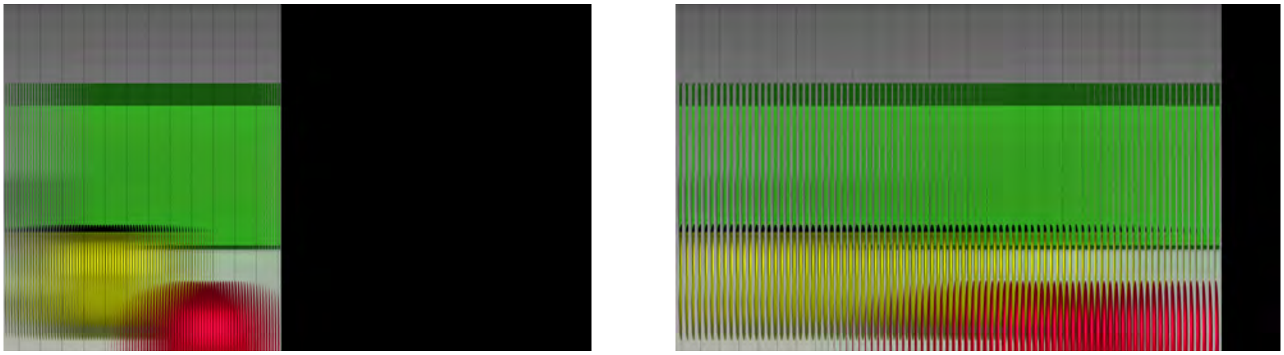

Figure 6. Left: Composite image for 40 LPI lenslet array. Right: Composite image for 20 LPI lenslet array. In both cases, the black is due to the input video does not have enough frames, resulting in an empty portion of the composite image.

We also tested the algorithm with a video taken with an Xperia Z5 phone.

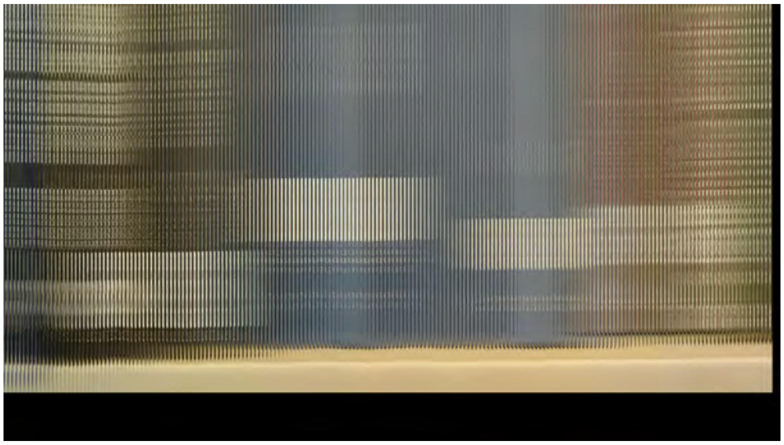

Figure 7. Composite image for 40 LPI lenslet using a Xperia Z5 filmed video. The video length was not quite long enough to fill all of the frames, and the video's y-resolution is not high enough to fill the length of the display resulting in the lower empty portion of the composite image.

We were able to increase the image processing speed from ~15 minutes per composite image to ~10 seconds per composite image.

4.3 Cloaking Performance

We found that in order to get a good image, the lenslets needs to be in direct contact with the display (acrylic top, lenslets middle, display bottom) with a few page's in between the lenslets and the display.

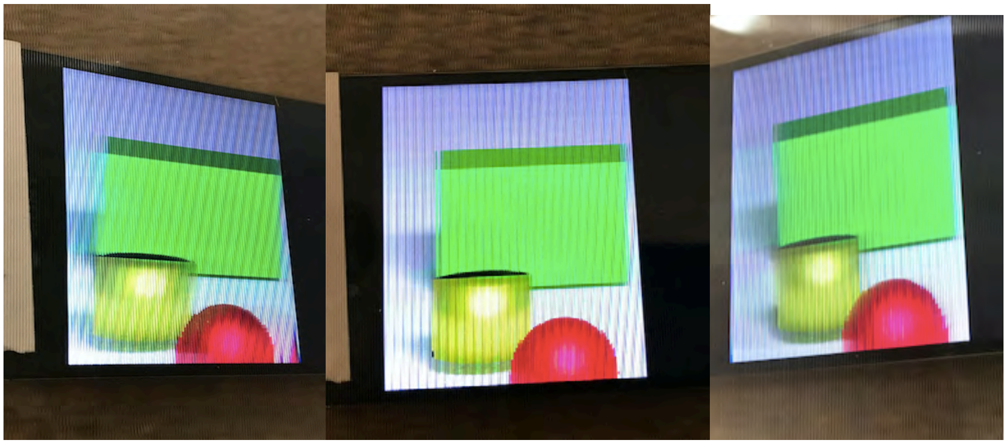

Figure 8. Composite images from Figure 6 using a 40 LPI lenslet array on a Z5 phone screen. Left image taken from a left viewing direction, middle image taken from front perspective, right image taken from a right viewing direction.

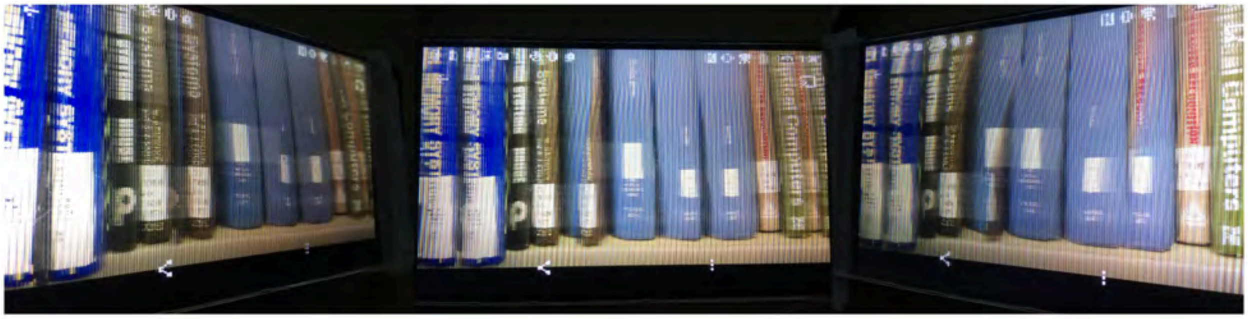

Figure 9. Composite image from Figure 7 displayed on a Z5 phone under a 40 LPI lenslet array. Images taken from a left, front, and right perspective.

With both composite images, we see a change in perspective with a change in viewing angle.

5. Encountered Difficulties

1. The group budget was reduced from $50,000+ to around $6,000, meaning we needed to use materials that had already been purchased or commercial items. We were able to stay below the new budget by only purchasing Z5 phones and the frame.

2. Commercial lenslets arrays are not significantly better than the ones we have in lab, but custom lenslets are expensive. The best way to accommodate is to reduce the number of views we originally desired, but the risk is not having enough views (or angular range) to make the demo believable. Sacrificing views for resolution provided good results.

3. Understanding how to run Prof. Howell's Mathematica code to regroup RGB pixels and reconstruct a composite image. Originally we were not responsible for creating composite

images, but using previous code we were able to create our own algorithm on Matlab. The algorithm will become more complex when working with slant columns (discussed in Appendix D).

4. When assembling the design setup, it is crucial to have the lenslets array placed at the correct focal length such that a) light is collimated out and b) each pixel appears to be the size of a superpixel (lenticular lens). Since the exact distance of the displays pixels from the surface is unknown, we estimated about a few sheets of paper and optimize from there.

5. Aligning the lenslets array to match the correct pixels is also challenging. It is important to check each view is correct, and to secure the lenslets array onto the display once the correct orientation is found. We cut the lenslets to be slightly smaller than the phone holes so we could make any necessary alignment adjustments.

6. Since the final presentation will consist of having the displays in holes of a board, the height of each phone needs to be considered when using capturing video for its composite image. This means we will have to adjust the scanning height of the camera for each phone.

6. Future Improvements & Transition Plan

Our customer is from the University of Rochester. Once the semester is over we will hand our finished design directly to our customer. We will also upload our Matlab code for LPI calculation and interlaced image onto our shared Dropbox such that a different senior design team can continue improving this project. A few improvements we have in mind are:

6.1 Interlacing RGB Pixels To Improve Horizontal Resolution

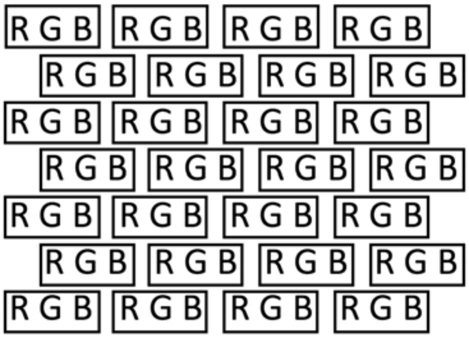

Figure 10. Conventional RGB layout of Z5 pixels. The actual RGB pixels are squares not rectangular.

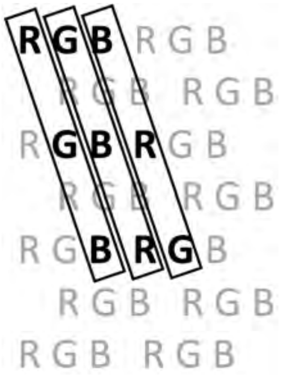

Figure 11. Regrouped pixels alignment.

For a high number a views (which defines the angular resolution), the most important factor is the horizontal resolution. Rearranging RGB pixels as shown in Figure 11 would allow us to increase the horizontal resolution by a factor of six while trading off vertical resolution.

The following spreadsheet shows how different parameters change according to different viewing distances, for maximum human resolution, based on using Xperia Z5 phone, with regrouped RGB pixels.

Human Resolution (mm)	Distance			Views
	ft	m	lpi	
0.29	3.3	1	87	56
0.58	6.6	2	44	112
0.87	9.8	3	29	168
1.16	13.1	4	22	225
1.45	16.4	5	17	281
1.75	19.7	6	15	337
2.04	23.0	7	12	393
2.33	26.2	8	11	449
2.62	29.5	9	10	505
2.91	32.8	10	9	561
3.20	36.1	11	8	618
3.49	39.4	12	7	674
3.78	42.6	13	7	730

Figure 12. Excel chart is only based on theory, the quality of the performance would also depend on the specification of the lenticular lens provide from the lens company.

6.2 Composite Image Algorithm Speed

If a composite image could be reconstructed fast enough, the cloaking device would also be able work in scenes with moving objects. We were able to decrease the image processing time by a factor of 6 per composite image. To get closer to real-time, an array of CCD cameras (rather than a single scanning camera) capturing angular information would further increase the processing speed.

6.3 Custom Display

The number of views required for near perfect cloaking over 30° (according to the human eye resolution) is:

$$\frac{30\,degrees}{1/60\ \ degrees} = 1800\,views$$

This is not possible with current commercially available displays. Having a custom display made to have a resolution similar to the Xperia Z5 over a larger area would help get closer to the desired cloaking resolution.

6.4 Environmental Considerations

If the cloaking display is used outdoors, the following must be considered:

- Brightness: We need a display screen bright enough in the day, but also adaptable to nighttime use
- Weatherproofing: Making sure CCD cameras and cloaking display is safe outdoors during different seasons (over a range of temperature degrees) and regular weather conditions (water proof)

Appendix A. Xperia Z5 Specifications

Since we need multiple views, where the number of views corresponds to how many pixels we can fit under a lenslet, we need a display with a high pixel density. A higher density gives us more pixels to create views with in a given area (lenslet size). Even though the Xperia screen has a display that is beyond the resolution of the human eye, only a few of its pixels are used per view making all of the pixels in the display important. In the end each view will be at/close to the spot size resolution of the human eye.

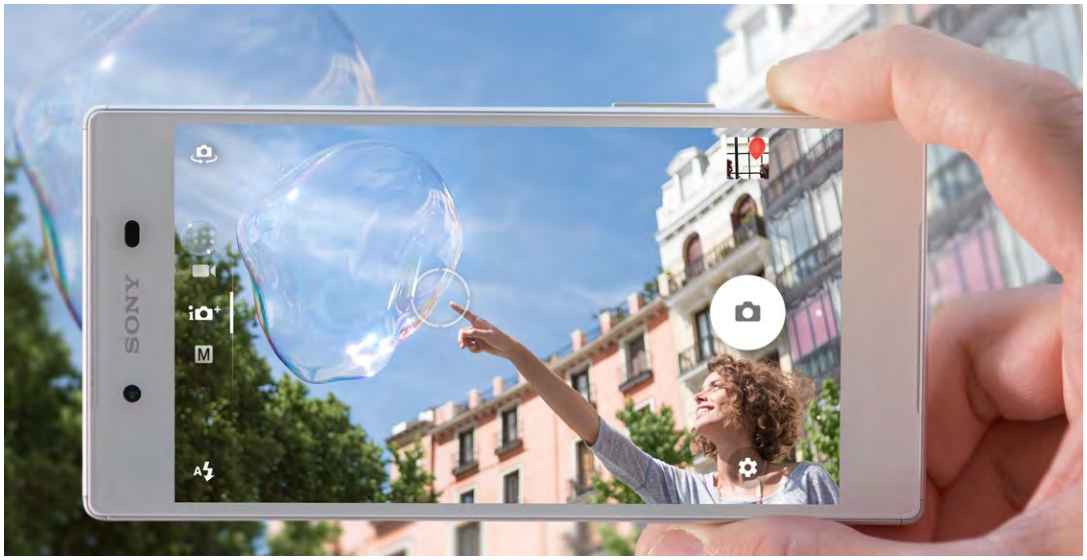

Figure 13. Picture of an Xperia Z5 phone screen.

- Display model: Xperia Z5 cell phone screen.
- Dimension: 5.5 inches (diagonal)
- Pixel Pitch: 806 pixels per inch (ppi).
- Resolution: 2160 x 3840
- Resolution with regrouping:

  Super pixel (resolution of one image of the composite image): 53 x 631

  Sub pixel: 13138 x 631

Where the horizontal Super pixel resolution is calculated by:

$$\frac{(2160)\left(\sqrt{37}\right)}{246\ views} = 53\ Super\ pixels$$

This is a rough estimation, since the pixels are actually aligned at a 9.5-degree angle counterclockwise to the vertical direction. The $\sqrt{37}$ comes from the phone screen rations, which was previously discussed in section 3.5.

Appendix B. LPI Calculations

Size: 20 LPI

Focal length estimation:

1. Start with the image of a single RGB pixel line (with a black background) on the display screen. The line should be in the lenslet orientation.

2. Place the lenslet above the line, where the line should be approximately in the middle of one lenslet.

3. The focal length should be about 8 layers of paper, so start to test the spacing with 5 pieces of paper, and add one layer at a time.

4. When the lenslet is a focal length away, only one lenslet will show the pixel line while all the other lenslet stay dark. Even when changing the viewing angle, only the particular lenslet will be illuminated. If the distance between the display and lenslet is shorter than the focal length, there will be more than one illuminated lenslet. If the distance between the display and the lenslet is longer than the focal length, there will be one lenslet partially illuminated.

Appendix C. Composite Image Matlab Code

```
%% Clear Variables and Command Line

clc;
clear variables;
close all;
imtool close all;
tic

%% Open movie file to make a composite image

Vid=VideoReader('MOV_0009.mov');
frames_num=Vid.NumberOfFrames;
freq=Vid.FrameRate;
vid_width=Vid.width;
vid_height=Vid.height;
imtool(read(Vid,1));
%% Now we want to stitch together a composite image using a subpixel interlace algorithm we have
developed

ppi=806;
lpi=40;
views=ppi/lpi
xdist=20;    % this sets the distance between image slant columns in the frame put into the composite
image
x_center=round(views/2);  %As columns are added to the composite image, this ensures the center view
for each lens corresponds to the center of the acquired frame  % This is the composite image with a
little extra dimensionality.

frame_first=1; % We will only use frames in the middle of the video.  This sets the first frame


yshift=200;

PhonexDimension=3840;
```

```
PhoneyDimension=2160;

lengthInx=PhonexDimension/ppi
lengthIny=PhoneyDimension/ppi

frames_needed=floor(lpi*lengthInx)

frame_last=148%frame_first+frames_needed-10

frame_distance=round(frames_num/frames_needed);

VCI=uint8(zeros(PhoneyDimension,PhonexDimension,3)); %Name of my composite image
VCIShift=uint8(zeros(PhoneyDimension,PhonexDimension,3));


%%  Make the composite image from a series of frames in a movie

xshift=1;
yshift=1;
lens=0;
reset1=0;

xStep = floor(frames_num/frames_needed);

for frame = floor(frames_num/2-xStep*(frames_needed-
1)/2):xStep:floor(frames_num/2+xStep*(frames_needed-1)/2) % go from the first defined frame to the last
defined frame
   v=read(Vid,frame);  % We read in the video frame by frame, I start at the last frame simply because
I scanned the image from right to left.
   reset1=reset1+floor(mod(frame,7)/6);
   for FrameColumn=1:views+floor(mod(frame,7)/6)
            for color = 1:3
                for PD = 1:(1920)
       VCI(PD,FrameColumn+lens*round(views),color)=v(PD,(FrameColumn-
x_center)*xdist+round(vid_width/2),color);
                end
            end
   end;

   lens=lens+1;

end;


%%

figure;
imshow(VCI);

imwrite(VCI,'XperiaCompositeImage40lpiv3.png');
toc
```

Appendix D. RGB Slant Re-grouping

The display’s RGB pixels are horizontally aligned, but to increase the horizontal resolution we will regroup the RGB pixels diagonally. Regrouping the RGB pixels allows us to increase the horizontal ppi by a factor of six by trading off six times the vertical ppi. Regrouping RGB pixels involves some coding to reprogram the image being displayed. This is done by rearranging the composite image’s array, which can be done on Matlab, Mathematica, or Python.

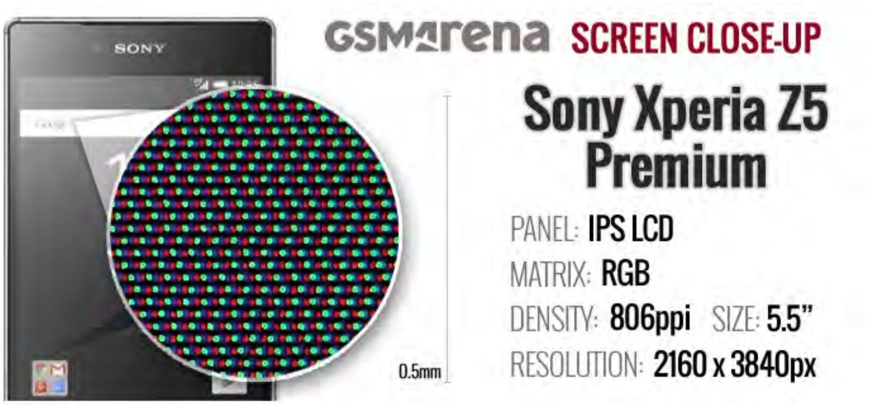

Figure 14. Pixel arrangement of Xperia Z5 Screen. This layout is not an accurate representation of the RGB pixel distribution.

*Note: We need the screen to work for ± 15 degrees horizontally, but only ± 3 degrees vertically. This means it is acceptable to improve the horizontal resolution while trading off vertical resolution.

After checking a Z5 screen with a microscope, we found their pixel alignment actually looks like the drawing below.

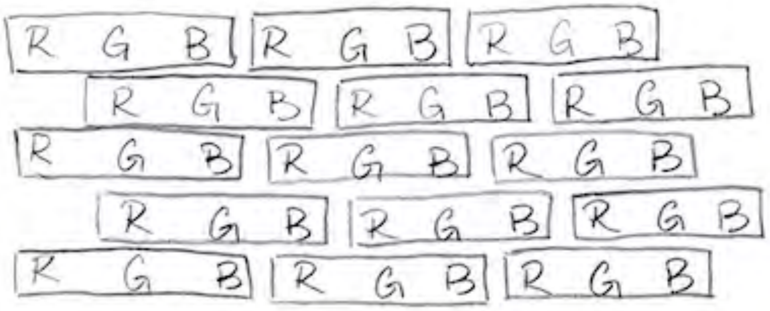

Figure 15. RGB pixels alignment of the Z5 screen. Drawing is not to scale. The actual RGB pixels are squares not rectangles.

With regrouping, the RGB grouping is the following.

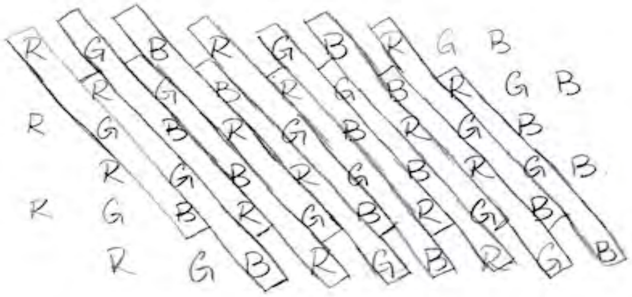

Figure 16. Regrouped pixels alignment. Drawing is not to scale.

After regrouping, the horizontal pixel density of the screen will increase about 6 times (by a factor of $\sqrt{37}$). This factor of six comes from a combination of a 3X increase from the sub RGB pixels combined in rows 1, 3, 5, and another 3X increase from the sub RGB pixels combined in rows 2, 4, 6. This also means the vertical pixel density of the screen will decrease by about 6 times. The re-arranged RGB pixels now formed a new pattern of columns called slant columns as shown in the Figure 5. When plugging lenslet array on the display screen, the lenslet array should also be placed with the same pattern just like the slant columns with the degree of $\tan^{-1}(\frac{1}{6})$ to the vertical. This angle of the regrouped pixel is calculated by

$$arctan\left(\frac{1\ pixel}{6\ lines}\right) = 9.46°$$

Note: The horizontal and vertical pixel density is a rough estimation, since the pixels are actually aligned at a 9.46 degree angle counterclockwise to the vertical direction.

The steps for regrouping the iPad mini's pixels are:

1. Figuring out how to write a function which takes a video and integer I as input and spits out the i-th frame of the video.

2. Writing a function with determines how many frames are in a video.

3. Writing a function which takes an integer N and calculates M integers which are all separated by the same integer m and roughly centered around N/2. For example, N = 1024, M = 6, m=3 gives solution 500, 504, 508, 512, 516, 520 as a satisfactory solution. This will be used to determine which frames to pull from the N frames of the video given M views and frame spacing of m frames.

Appendix E. Previous Design Ideas

Previously we had two other designs. One was our original design plan from last semester, terminated by our significant budget change from $100,000 to $6,000. The other design plan was a proposal for the new budget, but it was eliminated mainly due to our time constraint from completing our project and difficulties with rearranging the RGB pixels of a new display. Details of the two designs are provided below.

Original Plan with Custom 6 lpi Lenslet

A cloaking screen that works outdoors (day and night) at viewing distances of 50 feet or further, with +/-15 degree horizontal viewing angle, with an angular resolution close to the maximum resolution of the human eye. This is so the cloaking effect is believable.

Specification:

- Display: Using multiple Xperial Z5 4K phones connected together to make a 1m x 1m high-resolution display.
- Custom Lenslet array: 6 lpi.
- Viewing distance: 50 feet away. From 50 feet away this corresponds to a spot size of 4.43 mm, meaning each lenslet (super pixel) should be ~ 4.43 mm (equivalent to 6 lpi).
- Angular range: Undecided
- Views: 817 views

The number of views required for near perfect cloaking (according to human eye resolution) is:

$$\frac{90\ degrees}{1/60\ \ degrees} = 5400\ views$$

Which is not possible to obtain with current display technology. The number of views will instead depend on our display, which is the highest density 4K display available on the market. The maximum number of views we can get from the Z5 screen is:

$$\frac{(806\ ppi\ )(\sqrt{37})(3\ pixel/rows)(2\ rows)}{(6\ rows)\ (6\ lpi)} \approx 817\ views$$

- Superpixels: using 6 lpi lenslet array, the number of superpixels is equal to 236 super pixel/m
- Interlacing: Improved code on Mathematica
- Screen control: Customer would like to be able to control multiple Z5 screens simultaneously, to meet the desired 1m by 1m screen.

Since our budget was cut down from 100,000 to 6,000, we were no longer able to neither get a customized lenslet array nor buy multiple phones.

Proposal plan for the new budget requirement:

- Replace phones (that would sum to a larger display) with a large 4K UHD Monitor (under $1,500)
- Aim for 4x Human resolution
  -Lenslets 2x bigger
  -Viewing distance 0.5x
- Use commercially available lenslet array instead of using a customized one.

Effects on our design:

- Less pixels = Less views
- Lower resolution
- Limited to market availability of materials
- RGB regrouping program would need to be modified for regrouping new monitor's RGB pixels

Second Design with 4K Monitor

The second design idea came after our budget was reduced. It consisted of buying a commercially available 4K monitor to use as our display screen. Our specs were to cover +/-15 For this plan, we would have needed to recalculate number of views, lpi, and viewing distance. Additionally, we would have had to find a new way to regroup the RGB pixels for this monitor (different from the Z5) and a new lenslets array. Creating new code would have delayed our progress, and purchasing a lenselt array would have used up too much of our budget. After these considerations we decided to do keep work with the Z5 displays, but only three (the ones that were already purchased).

Third Design with Regrouped RGB pixels and 20 lpi

This design is also a result of our budget cut.

20 lpi with Rearranged RGB pixels:

- 20 lpi = 1.27 mm/lens
- Viewing distance: 20 lpi is good for 4.37 m away
- Angular range: ± 15 degrees
- Number of views: Since we are going to use the 20 lpi lenslet array from Prof. John Howell to proof concept, the number of view we will get from the lenslet array will be 245 views:

$$\frac{(806\,ppi)\left(\sqrt{37}\right)(3\,pixle/rows)\,(2\,rows)}{(6\,rows)(\,20\,lpi)} \approx 245\,views$$

- Superpixels: with the 20 lpi lenslet array, the number of superpixels is equal to 787 super pixels/m.
- Interlacing: Interlacing on Mathematica

Appendix F. Development & Timeline

Work Accomplished Fall 2016

- Framing: We developed and sent the CAD design of our demo’s board design to someone at UR Ventures. The board will be made of wood and painted matte black.

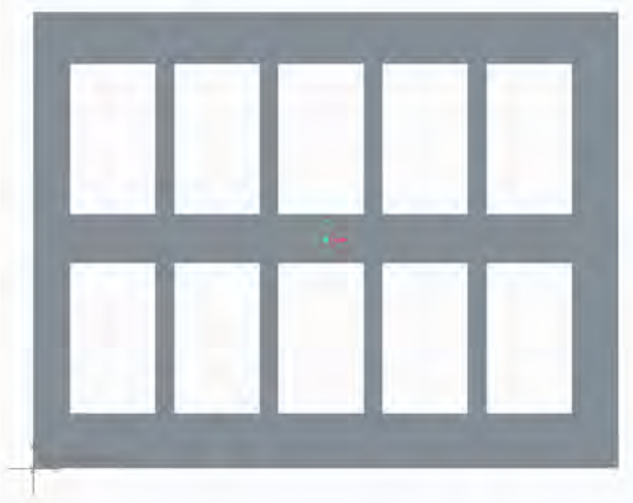

Figure 17. Snapshot of the CAD file showing the front view of the frame.

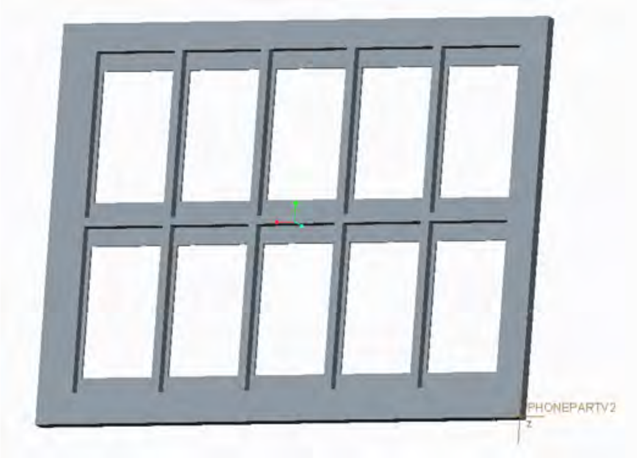

Figure 18. Snapshot of the CAD file showing the back view of the frame. Each hole is cut to fit a cell-phone, but only one of the holes will contain a cloaked phone display.

- Interlacing processing: We learned the basics of interlacing, which means taking a composite image and modifying its array to how you would like the phone to display the image. By rearranging the array, you can change which pixels are showing specific parts of the image, meaning the pixels get “regrouped” to function vertically rather than horizontally.

- PC phone interface: After the lenslet is aligned to the display, it would be tedious to take the lenslets off just to be able to swipe on the phone screen (to change an image, change phone settings, etc.). That would mean realigning the lenslet to the screen every time we need to use the phone screen. It would be much more convenient (and time efficient) to remote control the display phone with another device. Currently we are using an application called TeamViewer. Installing this application on two separate phones gives us the ability to control one phone (with lenslets attached) with another. It will be like the two phones are sharing a screen; the phone without a lenslet array will be used to change settings on the phone being used as a display.

- Lenslet array: Our group learned how to work with lenslet arrays, which are cut to the size of the phone screen and then adhered onto an acrylic board that is also the size of the screen. The acrylic's purpose is to protect the phone display. To correctly place the lenslet array (adhered on the acrylic) one focal length from the display pixels, we would need to know the exact location of the pixels and the focal length of the lenslets. Since we do not know these parameters, we can only estimate by putting the lenslet at various distances from the display (using small pieces of stacked paper as spacers) and finding the best resolved image. Previously with the iPad mini, the distance was about 8 pieces of paper. Due to this display being different the thickness of the screen (from pixels to the cover) will be different. So we will work from the 8 pieces of paper to find the best spacing.

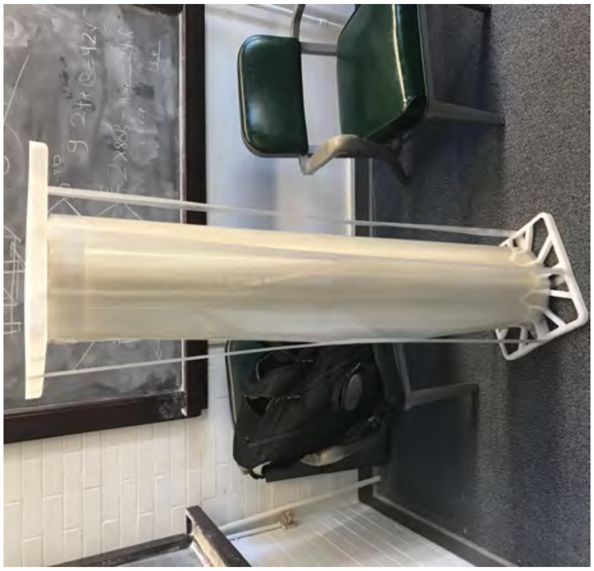

Figure 19. Lenslet array sheet.

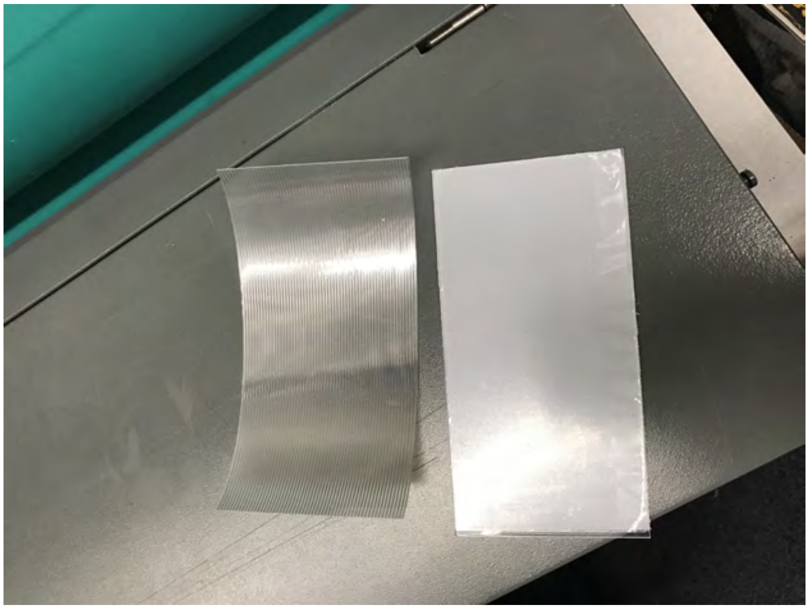

Figure 20. Cut lenslet array (left) and acrylic board (right).

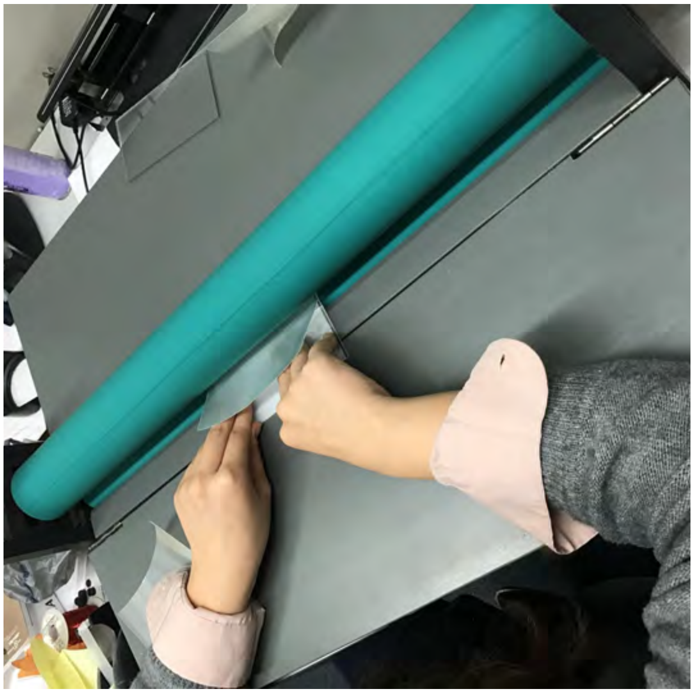

Figure 21. Adhering a lenslet array to an acrylic board using a roller.

Work Accomplished Spring 2017

- RGB Regrouping: We generated a test-interlaced image from a video taken with a Z5 phone and Mathematica code from Prof. Howell.

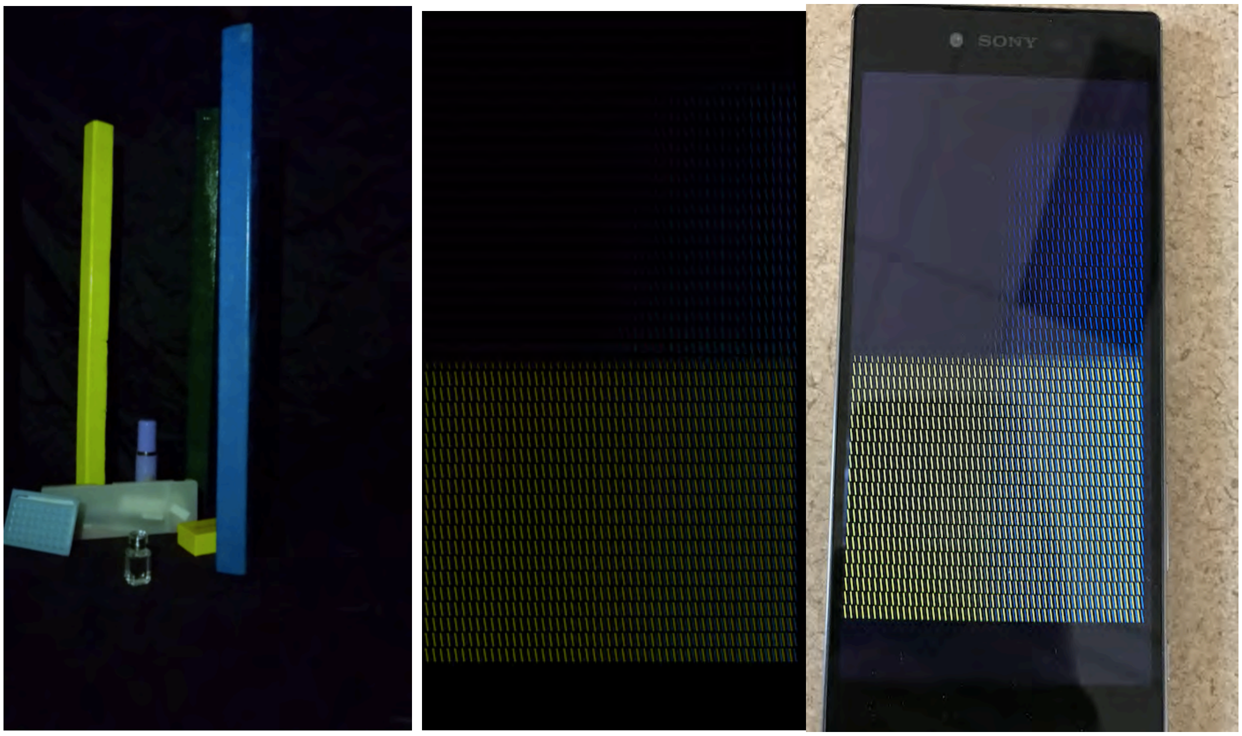

Figure 22: Left: Captured frame from demo video. Middle: interlaced image from a computer screen (regular pixel arrangement) Right: How it looks on the Z5 Screen.

We decided to take the image processing to Matlab to lower the time is takes to generated interlaced images. Prof. Howell translated his Mathematica code for the iPad to give us as a starting point. His code takes a video input, selects specific frames (the number of frames is specific to the iPad and video length) and generates a composite image. After regrouping the displays RGB pixels, we get a final interlaced image. With the lenslet array on the display, a viewer is able to see different portions of the interlaced image, making it appear like there is a real object inside the phone.

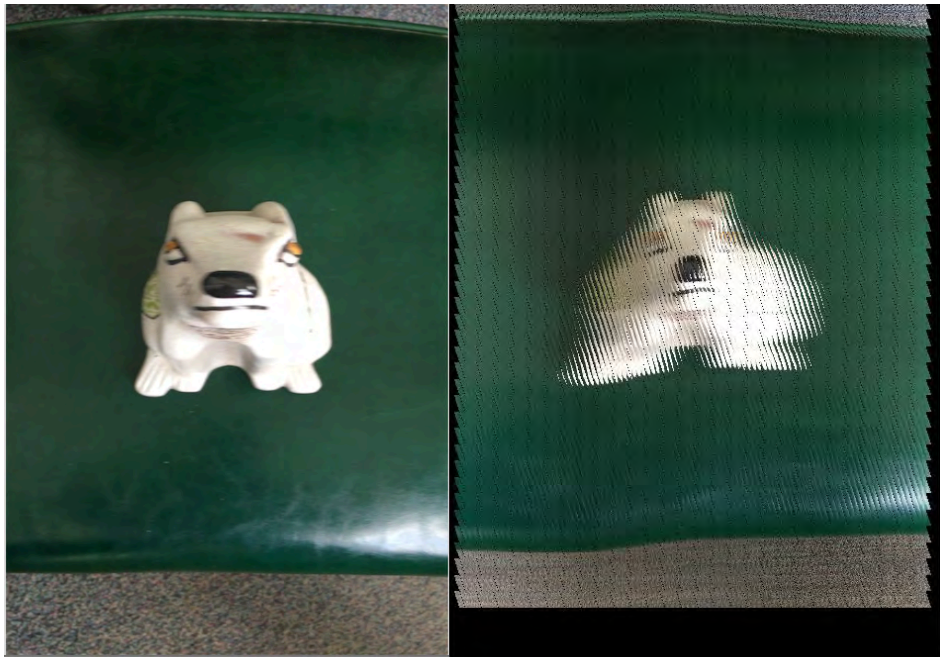

Figure 23: Left: Composite image generated from video provided by Prof. Howell. Right: image with regrouped RGB pixels from a computer screen works on iPad mini. (Our goal is to make our video generate an interlaced image that works for the Z5.)

The Mathmatica code works well for the iPad mini and a 6 second video. Now our goal is to make the code more adjustable for the Z5 (or other displays) and longer videos. The rewrite for the Z5 includes having adjustable displaying dimensions, an automates slant column length calculation, and a way for selecting the appropriate number of frames regardless of the input video length.

Week Feb 6-15

- Settled down on the lenslet that is going to used: find the Lenslet with smallest LPI with price and delivery date, etc.
  -Progress: due to budget constraints and time, we decided to use the 20 lpi that is already available to us from Prof. Howell's lab.
- Discuss what we want to present for the view in the meeting.
  -Progress: No meeting happened, but we decided to do a demo using three Z5 phones as displays mounted on a board we designed on Creo.
- Set up the track for camera to capture images.
  -Progress: We found lab space in Wilmot 5th floor and have equipment from Prof. Howell's lab.
- Evaluate interlacing code.
  -Progress: we have the iPad Mini Mathematica code. We are still having issues troubleshooting it, but we have made progress in understanding it conceptually.

Week Feb 15-Feb 24

- Create a scene that works for our CAD board display.
  -Progress: We created a scene and recorded a video that was sent to Prof. Howell.
- Set up the track for camera to capture images.
  -Progress: We set up both the camera and the scene for test demo.
- Take a video for Prof. Howell to create 2 composite images for each Z5 phone.
  -Progress: We sent a video to Prof. Howell and are currently waiting for Prof. Howell to make generate the composite image. After testing this composite image, we will make a new scene for our final design. We will need to record a video for each phone.
- Modify code for Z5 RGB layout.
  -Progress: We successfully ran the Mathematica code on our Macs, but the output is not as expected:

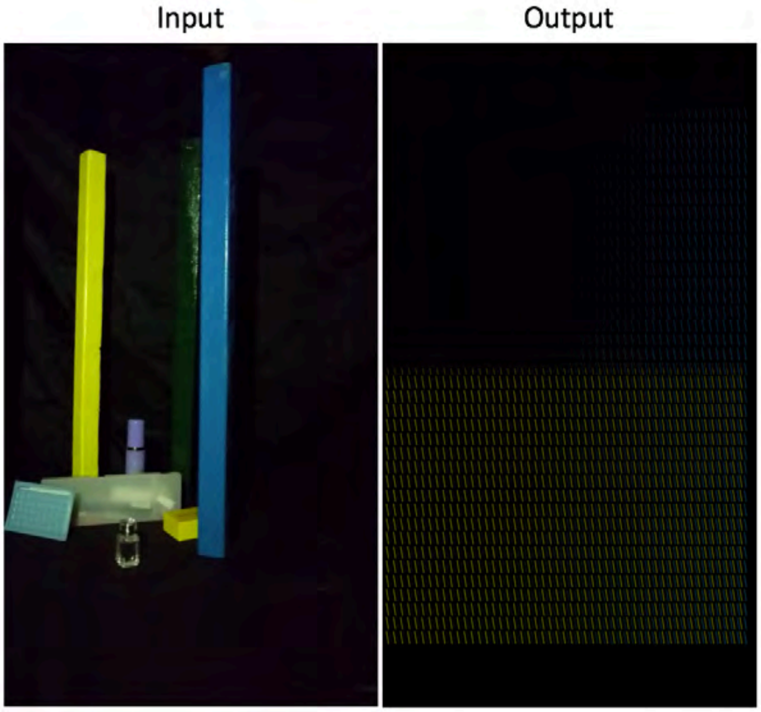

Figure 24. Test of RGB regrouping for an iPad mini. Video was taken with a Z5 phone.

  We are hoping to meet with Prof. Howell and our customer next week to get an understanding why the original image is being cropped to this size.
- Obtain composite image.
  -Progress: Waiting for Prof. Howell.

Week Feb 25-March 4

- Find a way to use a PC to remote control the phones. This would allow us to turn each phone on, off, or to unlock the phone by using the PC rather than by removing the lenslet array to swipe on the phone.
- Place a lenslet array on a cell phone screen at the focal distance (approximately 6 pieces of paper).
  -Progress: Last semester we practiced placing lenslet array on acrylic boards.

- Test the composite image with the phone.

Week March 5-March 11

- Meet with Prof. Howell to discuss the idea of how to obtain an composite image from a video
  --Progress: Write out the idea flow of how the program should work

Week Mar 12-18

- Spring break

March 19-March 25

- Prepare for the presentation on March 27th for IA.

Week Mar 26-Apr 1

- Finalize the CAD design.
  -Progress: Remeasured the dimension and adjusted the design.
- Order the CAD design.
  -Progress: Sent the CAD file to manufacturer and waiting for a quote and lead-time.
- Worked on the Matlab code for composite image and interlacing RGB pixels.
  -Progress: We had successfully gone through and adjusted most of the matlab code, but there was a line that we were not able to figure out. We will meet with Prof. Howell on Apr.4th to solve this problem.

Week Apr2 -Apr 8

- Discussed the logic to write the Matlab code for regrouping pixels for our Xperia Z5 display.

Week Apr 9-Apr 15

- Working on Matlab code for interlacing pixels.
  -Progress: code would run but need to be debugged.

Week Apr 16-Apr 22

- Work out the Matlab code with Prof. Howell.
  -Progress: We had obtained a code that would work with our display, there are minor error would need to be fixed.
- Editing DDD for final review.
  -Progress: We had changed our DDD with new specs.

Week Apr 24-Apr 30

- Final design Review.
- Decide which lenslet we are going to work with.
  -Progress: We decided to use both 20 and 40LPI lenslets so the viewers could compare the differences.
- Take our final videos that are going to be used on the design day.
  -Progress: We asked Prof. Zavislan for permission to getting into the presenting area a day earlier to take videos, however, due to power shortage, we decide to do it on the design day.
- Set up our design.
  -Progress: We took videos in POA library to make demo composite images, and rough cut our lenslets.

Week May 1st-May 5th

- Finalize the design set up.
- Finalize the DDD.
  -Progress: We finalized the DDD and sent it to our customer.
- Prepare for design day.
  -Progress: We had lenslets cut to the proper size, reshaped our frame to fit the sound button and charging hole, and took videos on design day before presenting.