Visual Object to Audio Mapping Using Stereovision

Michael Yeung (60955044), Neil Pahl (88483045)
Department of Electrical Engineering, University of British Columbia

Abstract—Stereovision allows us to recreate an object or space in three dimensions from two or more two dimensional images. Humans intrinsically have this ability to perceive an object in a spatial area while any man made image process does not because images are received on a two dimensional plane. In this paper we design a method to retrieve the three dimensional information of an object in the field of view from two stereovision sources and map it to a three dimensional audio source. By primarily using the methods of template matching and cross correlation, we develop a novel method of retrieving this three dimensional information and mapping it to an audio source to use as an object locator for a visually impaired person.

I. Introduction

Stereovision is used every day by human beings. It allows us to thread a needle and catch a ball with exceptional proficiency. Without out it, many things like backing into a parking stall or shaking someone's hand would be an extremely difficult task to achieve.

Not only does it apply to nature, but many man-made applications have arisen with the need for artificially constructed stereo vision solutions.

For instance, robots are a predominant subject of stereovision [1] research. Many automated systems need a reliable and passive way to extract depth information of surrounding objects. For example, many automated mobile robots require range information its surroundings to navigate autonomously around obstacles.

While there are several different methods of retrieving the 3D information of an object, from structured light techniques [2] to auditory methods [3].Stereo vision was used in our project to retrieve our necessary 3D information by acquiring two similar but offset images from two planar vision sources, detecting the object in our field of view and processing the information to retrieve its X,Y,Z co-ordinate in a normalized fashion.

For our project, the goal was to create a proof of concept stereovision system that was able to detect a single object within two 2D images and output the object's 3D co-ordinates into a multi-dimensional auditory signal in the form of a pulsed tone. Initially, the pulsed tone will have dimensions of periodicity (depending on depth in the field of view), fade (depending on the lateral position) and pitch (depending on the vertical position). We decided that stereovision is an ideal technique to achieve our goal.

Eventually this concept would be ideally implemented and applied to situations where a visually impaired person required object awareness of their surrounding areas.

This paper is organized as follows. In section II we provide some basic background information about stereovision basics, and cross-correlation. Section III we describe the types of data and equipment used. Section IV we provide our methods and algorithms. Section V we summarize our results and problems and finally offer a conclusion in Section VI.

II. Background Information

A. Stereovision Basics

Vision systems capture images in two dimensional forms where the horizontal and vertical disparities can be recovered from a single frame. However, the object's depth in a field of view requires some additional processing methods.

For example, humans have two eyes located side by side on a horizontal plane that takes to two similar images at slightly different angles. The brain then processes these slightly offset images by looking at the dissimilarity between the images and angle offsets to reproduce the field of view in way we can understand it in the X,Y,Z planes [4].

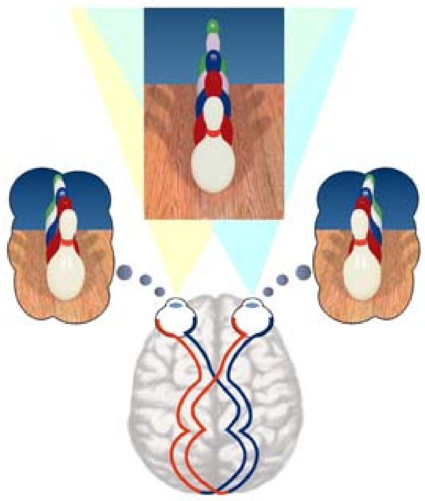


Figure 1 - Human Imaging Example

The function of stereovision is to replicate this process of taking two (or more) offset vision sources and deriving their three-dimensional information given the disparity found from comparing the data in from both images.

B. Cross Correlation

Cross correlation is a signal processing technique used to measure the similarity between two offset signals [5]. It is commonly used to rectify the delay caused by a long duration signals (e.g. satellite communication), but more importantly for us, pattern recognition.

Cross Correlation is similar in nature to convolution. While convolution flips and slides, correlation only slides.

In our case we used the discrete definition of cross correlation (Figure 2) because it enabled us to find the disparity or offset between the images from each camera.

$$(f \star g)[n] \stackrel{\mathrm{def}}{=} \sum_{m=-\infty}^{\infty} f^*[m]\ g[n+m].$$

Figure 2 - Discrete Cross Correlation Definition

MATLAB made this a fairly painless process with the availability of xcorr() and corr() functions in both one and two dimensional forms. However because of computational intensity of these types of iterative functions they should only be used sparingly to save processing time of any system.

III. Materials

The project required two vision sources dynamically inputting data into the MATLAB environment. Doing this was accomplished by purchasing two cost-effective Pixxo AW-I1130 webcams, using our own laptop and the MATLAB environment

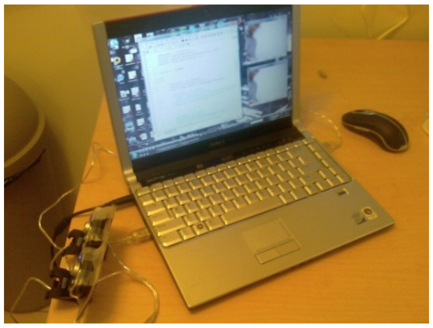


Figure 3 – Webcam/LAPTOP to MATLAB Setup

As shown in Figure 3, the two webcams (bottom left hand corner) are connected to the laptop which directly feed the data obtained from each camera into MATLAB. It is then passed through object and audio processing blocks respectively. Finally the processed data is outputted to an audio source (in our case our laptop speakers) as a WAV[6] file .

It is also important to note that because this was only a proof of concept, we were aiming to detect a single object on a clean white background.

This was justified by the fact that this system was going to be fully automated and multiple object detection would have made the objectives exponentially harder. Moreover, for "noisy" backgrounds this created various other signal processing problems which resulted in additional computational power needed that we did not have the resources to obtain.

Therefore we decided upon using a two webcam system which detects a single object on a clean background because it was still able to provide a good proof of concept of our audio mapping system working.

IV. Methods and Algorithms

Our project can be broken down into two main processing blocks. The first is the object processing block where the raw data from the cameras is processed to the point where an object can be detected in spatial field and its location can be defined in a three-dimensional co-ordinate system.

With the object's location in the form of a single three-dimensional co-ordinate, this data is inputted into the audio processing block which takes this information and converts this into a three dimensional pulsed tone.

A. Object Processing Block

Before any image processing was done we first took the color image and converted it into its HSV components where we then only took only the saturation components of each camera source.

Because a white background has a very low saturation component, we were able to assume that any object in the field of view would have a high enough saturation component that we could distinguish from the background.

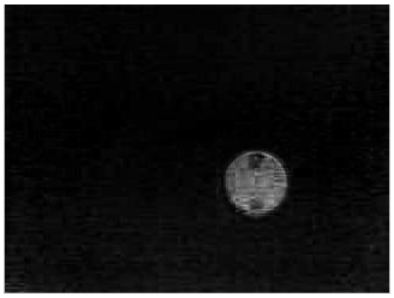


Figure 4 - Left Camera Saturation Only Image

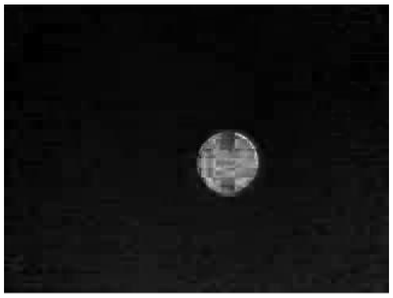


Figure 5 - Right Camera Saturation Only Image

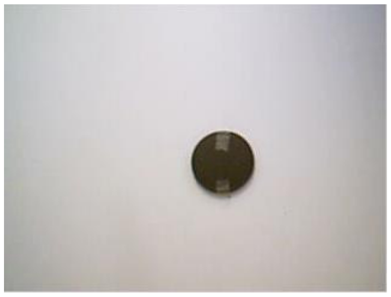


Figure 6 - Original Color Image All Components

Figure 3, 4 and 5 show the results of converting the image into its saturation components only compared to the original data collected by the cameras

To determine where an object is in 3D space, we must locate the object in the 2D space of both the left and right image. To do this, we use the method of template matching [7][8]. This method involves localization of subsections from the target (left) image with respect to the reference (right) image. I our case, we divide the left image into a 6x8 grid of sub images (Figure 7), where each is first checked to see whether it contains an object.

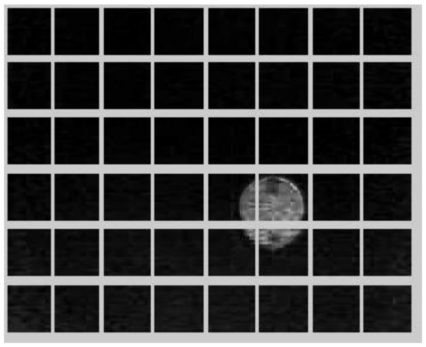


Figure 7 - Left Image Divided Into Sub Images

To make this judgment, we must recognize the following properties of a box containing an object. If a box from the target image contains an object, that object must also be present in the reference image, but offset horizontally. If there is no horizontal offset between an object in the two images, the object can be assumed to be far away and not detected. Objects which are nearby will show large horizontal offsets between the left and right images.

In our implementation, however, some measures are taken to enhance the robustness of our detection system. Each sub image is reduced to a row vector containing the max value of each column in the image (Figure 8).

```
ytarg_start = (ii-1)*boxY+1;
ytarg_end = ytarg_start+(boxY-1);
target_row = max(targ_im(ytarg_start:ytarg_end, 1:320));
target_row = single(im2bw(target_row, .2));
```

Figure 8 - MATLAB Grid and Row Code

For example, Figure 9 is the row vector which corresponds to the 4th row in Figure 7. Doing this, allows us more flexibility with MATLAB functions and some resistance against misaligned cameras.

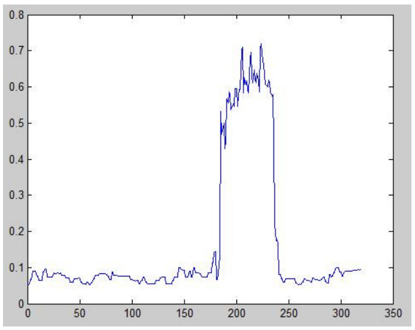


Figure 9 – Max Values of Left Image Row 4

To observe an offset between the right and left image, we cross correlated each target box with the entire row in both the left and right images. The point where the cross correlation has the greatest value is where the target box resides in the image. MATLAB returns the shift of the cross correlation as a lag (Figure 10).

```
[corr_val1,lag_l] = xcorr(target,target_row);
[corr_val2,lag_r] = xcorr(target,refr_row);
corrLag_L = [corr_val1' lag_l'];
corrLag_R = [corr_val2' lag_r'];

max_corr_val1 = max(corr_val1);
max_corr_val2 = max(corr_val2);
```

Figure 10 - MATLAB Cross Correlation Code

After finding the lag of the left and right image, the difference gives us the horizontal offset of the target box.

We then insert the horizontal offsets into the 6x8 *obj_disparity* matrix. This matrix represents our disparity map, providing the depth of objects in our field of view with a horizontal and vertical resolution of 8 columns and 6 rows. When an object is present in the field of view, the *obj_disparity* matrix will have a non-zero value in the corresponding matrix element. *obj_disparity* elements with large disparity values are perceived to be near while smaller values represent farther objects.

B. Audio Processing Block

After the disparity map has been determined, the 3D information is available to be mapped into the audio space. The audio processing block maps the 3D visual information into a pulsed tone signal which can alert the user about the location of object in view. The horizontal location is mapped to fade, the vertical location is mapped to pitch, and the depth of the object is mapped to pulsing frequency of the tone. Because each non-zero element in the *obj_disparity* matrix represents an object in that grid location, a tone signal is created to represent that object. The fade ratio and pitch frequency of the object is mapped via the following MATLAB code in Figure 11.

```
fade = jj/boxNumX;
pitch = (boxNumY-ii)/boxNumY * 400 + 125;
```

Figure 11 - MATLAB Fade and Pitch Code

Where *jj* and *ii* is the column and row of the *obj_disparity* matrix, repectively. *boxNumX* and *boxNumY* is the number of object grid boxes in the x and y. The fade calculation calculates the fade ratio between left and right audio channel. The pitch calculation maps a high frequency to objects which are above in the image. Using these parameters, a signal for these grid locations are created as shown in Figure 12.

```
signal_L = sin(2*pi*pitch*t +.2*(0.5-fade));
signal_R = sin(2*pi*pitch*t +.2*(0.5-(1-fade)));

stereo_out = stereo_out + [signal_L' signal_R'];
```

Figure 12 - MATLAB Object Audio Signal Code

At this point, the fade is only contributing to a phase offset to the left and right audio signal. This aids in the user's ability to detect the direction. However, to provide a more user friendly output, the actual fade and pulse frequency is implemented during the last stage using average values of all the present objects. This way, any discrepancies in fade and disparities will not confuse the listener. To do this, the values of fade and disparity are loaded into an array which is averaged in a later stage.

In the final stage, the depth pulse effect is created by multiplying the output signal by a square wave of the desired pulse frequency. The Square wave also exhibits a phase delay between left and right audio. Figure 13 contains the final stage of the audio output block.

```
pulse_f = mean(depth_ok)/10;
mean_fade = mean(fade_delay);

depth_pulse_L = 0.5*square(2*pi*pulse_f*t
                +.2*(0.5-mean_fade))+0.5;

depth_pulse_R = 0.5*square(2*pi*pulse_f*t
                +.2*(0.5-(1-mean_fade)))+0.5;

stereo_out = stereo_out.*[(depth_pulse_L*
    (1-mean_fade))'  (depth_pulse_R*(mean_fade))'];

wavplay(stereo_out,Fs)
```

Figure 13 - MATLAB Final Stage Audio Output Code

V. Problems and Results

A. Object Detection Problems

Originally we set up our system to do the correlation between the right and left cameras after we had grey scaled and Canny Edged [9] the image. We soon found out the method of taking the object location at the max correlation point was extremely inconsistent and we could never obtain reliable image information.

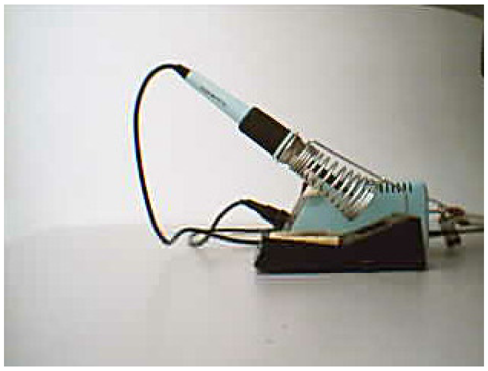


Figure 14- Original Preprocessed Image

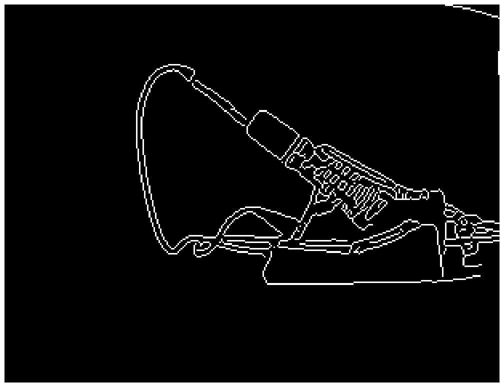


Figure 15- Canny Edge Filtered Object after Grey Scaling

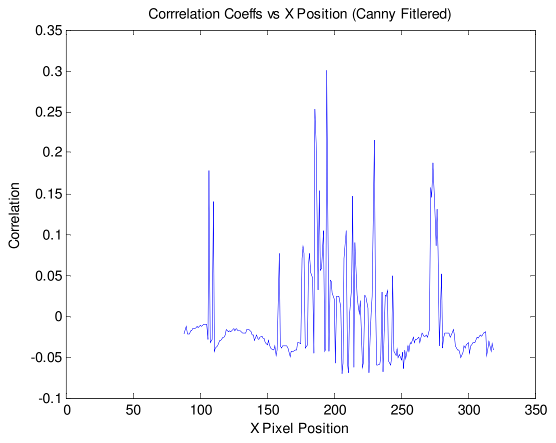


Figure 16 - Correlation of Canny Edge Filtered Object

As you can see in Figure 16 the correlation algorithms would always give us outliers which routinely would be larger than the correlation coefficients where the object was located. Moreover the magnitudes of the correlation would jump all over the place

However once we changed our method and used the saturation component of an image, we were able to determine an object's horizontal location as shown in the results for Figure 9 for an object shown in Figure 6.

B. MATLAB Performance Problems

Because the whole system is entirely software based we relied on many performance intensive instruction sets such as cross-correlation and for loop iterations to achieve our goals.

Upon testing, the system could only 1-2 full computations cycles per second. More relevant though is the fact that the proof of concept works and a hardware based solution for our problem could significantly increase the performance

C. Results

To represent objects at different 3D locations, we adhered paper cut-outs to a white wall and captured the left and right images of these objects using two web cameras which are feed directly into a laptop for MATLAB processing. The camera and laptop combination were kept together on a table and the whole table would be moved to different distances from the wall. Figure 17 shows this setup.

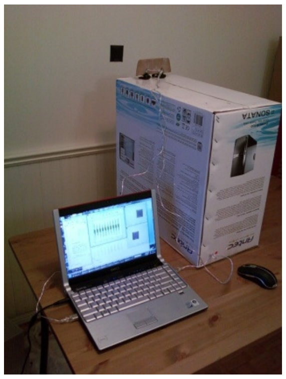
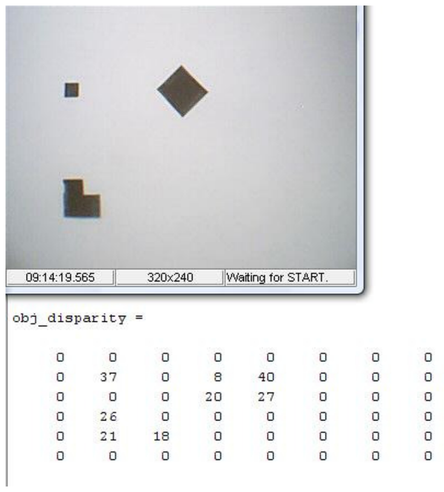


Figure 17 - Experimental Setup

The object processing block would build the *obj_disparity* matrix which contained the 3D data extracted from the two initially captured images. Each element in the *obj_disparity* matrix, if non-zero, represents an object in 3D space. The X coordinate is given by the column in the matrix, The Y coordinate is given by the row in the matrix, and the Z coordinate (depth) corresponds with the value (disparity) of the element.

Using the template matching algorithms, we were able to create an object map with the locations of the multiple objects all contained inside the *obj_disparity* matrix as shown in Figure 18-31

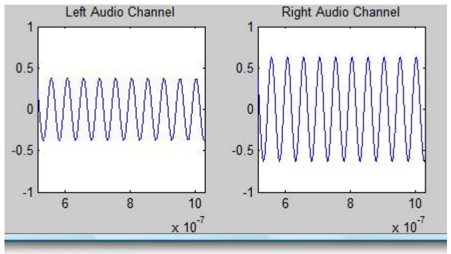


```
obj_disparity =

     0     0     0     0     0     0     0     0
     0    37     0     8    40     0     0     0
     0     0     0    20    27     0     0     0
     0    26     0     0     0     0     0     0
     0    21    18     0     0     0     0     0
     0     0     0     0     0     0     0     0
```

Figure 18 - Multiple Object Detection

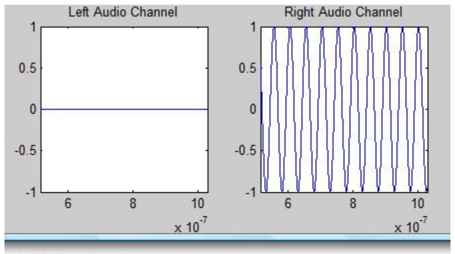
However, as our design was originally specified to locate one object, the audio processing block cannot adequately output sounds that map these object to the audio domain intuitively. Also, because of the thresh holding stages, rows with multiple objects have high correlation with both objects and can result in inaccurate disparity measurements (Figure 18). This can possibly be corrected if the object in the image has more entropy, such that the highest correlation will only occur when it is overlapping itself.

When observing a single object, audio mapping can be completed for a given *obj_disparity* matrix. Figure 19, Figure 20, and Figure 21 illustrate the audio output of varying fade (X coordinate) and frequency (Y coordinate) as created from a single object located in the field of view of the cameras. When an object is higher up in the field of view, the frequency is higher. The amplitudes of the output left and right sound channels are weighted accordingly depending on the column (X coordinate) of the object.



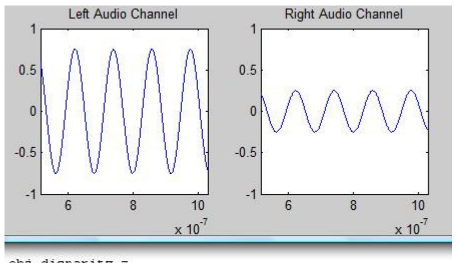
```
obj_disparity =

     0     0     0     0    41     0     0     0
     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0


pitch =

  458.3333
```

Figure 19 - Object at Row 1 Column 5



```
obj_disparity =

     0     0     0     0     0     0     0    38
     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0


pitch =

  458.3333
```

Figure 20 - Object at Row 1 Column 8

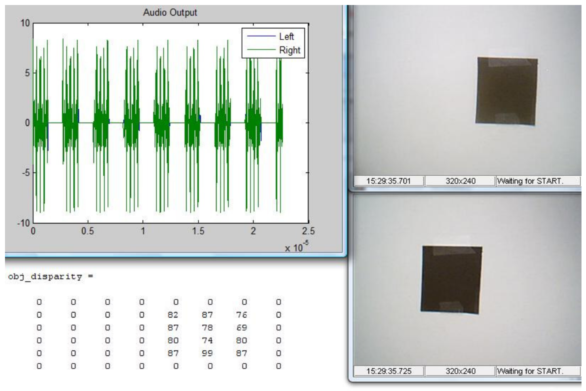


```
obj_disparity =

     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0
     0    27     0     0     0     0     0     0
     0     0     0     0     0     0     0     0


pitch =

  191.6667
```

Figure 21 - Object at Row 5 Column 2

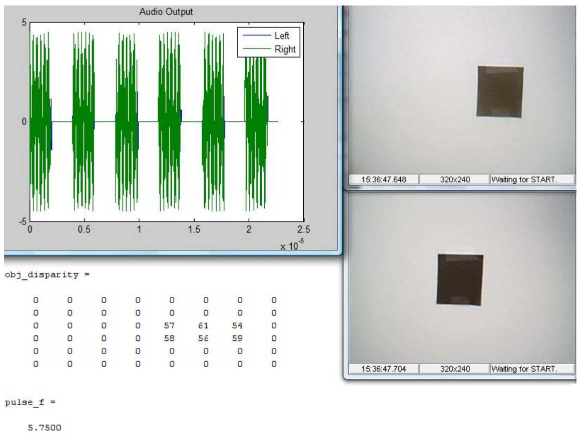
For larger objects that fill more elements in the *obj_disparity* matrix, the signals created from each element are summed together so that a user can realize larger objects via the presence of more tones.

To show the depth, the summed tones are pulsed at a frequency, relating to the average value of the disparity elements present in the object. Figures 22-31 show how the audio signals pulse according to distance. In these figures, the top right and bottom right images are the captured images of the left and right web cameras, respectively. Closer objects have a faster pulsed tone.



```
obj_disparity =

     0     0     0     0     0     0     0     0
     0     0     0     0    82    87    76     0
     0     0     0     0    87    78    69     0
     0     0     0     0    80    74    80     0
     0     0     0     0    87    99    87     0
     0     0     0     0     0     0     0     0


pulse_f =

    8.2167
```

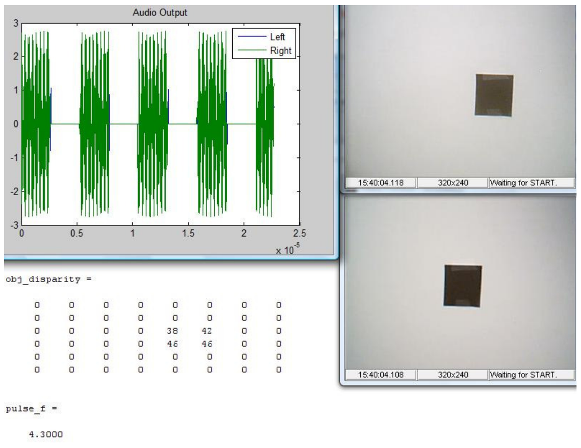
Figure 22 - Object at 0.30m



Audio Output
Left
Right
x 10^-5
15:36:47.648 320x240 Waiting for START.
15:36:47.704 320x240 Waiting for START.

```
obj_disparity =

     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0
     0     0     0     0    57    61    54     0
     0     0     0     0    58    56    59     0
     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0


pulse_f =

    5.7500
```

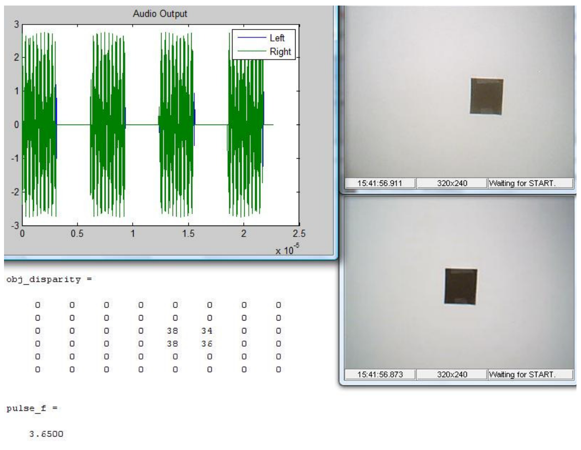
Figure 23 - Object at 0.40m



Audio Output
Left
Right
x 10^-5
15:40:04.118 320x240 Waiting for START.
15:40:04.108 320x240 Waiting for START.

```
obj_disparity =

     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0
     0     0     0     0    38    42     0     0
     0     0     0     0    46    46     0     0
     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0


pulse_f =

    4.3000
```

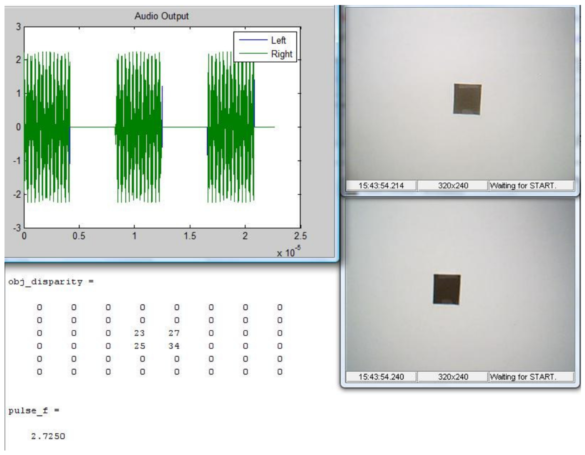
Figure 24 - Object at 0.50m



Audio Output
Left
Right
x 10^-5
15:41:56.911 320x240 Waiting for START.
15:41:56.873 320x240 Waiting for START.

```
obj_disparity =

     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0
     0     0     0     0    38    34     0     0
     0     0     0     0    38    36     0     0
     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0


pulse_f =

    3.6500
```

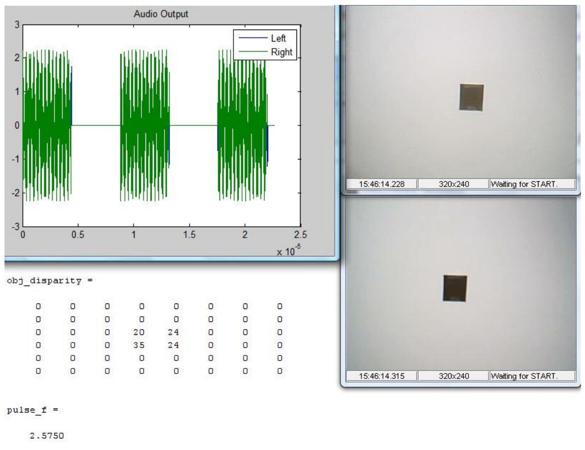
Figure 25 - Object at 0.60m



Audio Output
Left
Right
x 10^-5
15:43:54.214 320x240 Waiting for START.
15:43:54.240 320x240 Waiting for START.

```
obj_disparity =

     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0
     0     0     0    23    27     0     0     0
     0     0     0    25    34     0     0     0
     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0


pulse_f =

    2.7250
```

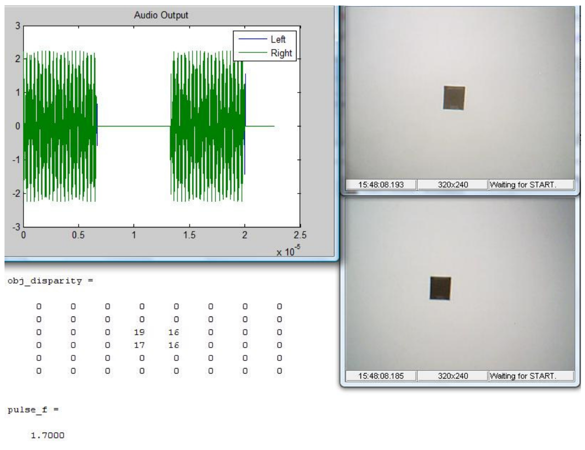
Figure 26 - Object at 0.70m



Audio Output
Left
Right
x 10^-5
15:46:14.228 320x240 Waiting for START.
15:46:14.315 320x240 Waiting for START.

```
obj_disparity =

     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0
     0     0     0    20    24     0     0     0
     0     0     0    35    24     0     0     0
     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0


pulse_f =

    2.5750
```

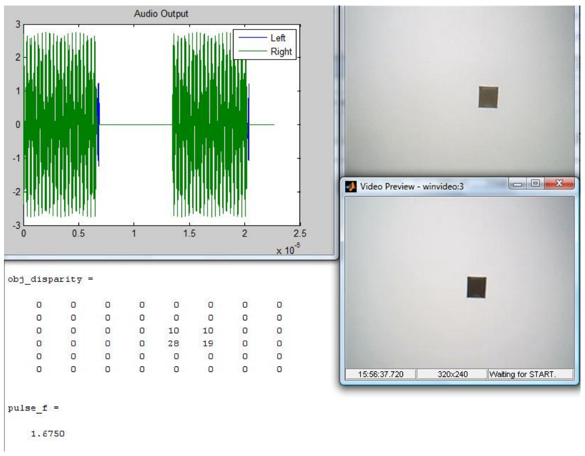
Figure 27 - Object at 0.80m



Audio Output
Left
Right
x 10^-5
15:48:08.193 320x240 Waiting for START.
15:48:08.185 320x240 Waiting for START.

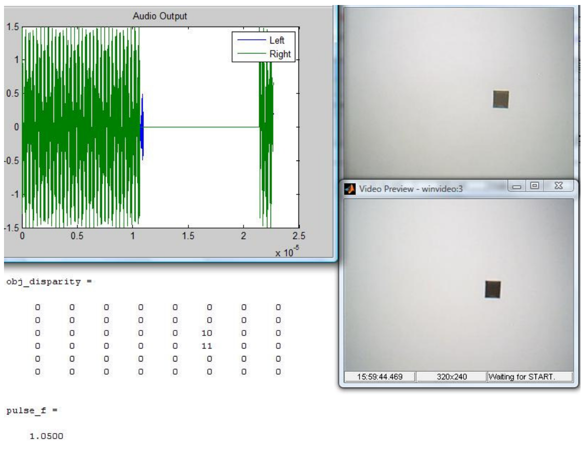
```
obj_disparity =

     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0
     0     0     0    19    16     0     0     0
     0     0     0    17    16     0     0     0
     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0


pulse_f =

    1.7000
```

Figure 28 - Object at 0.90m



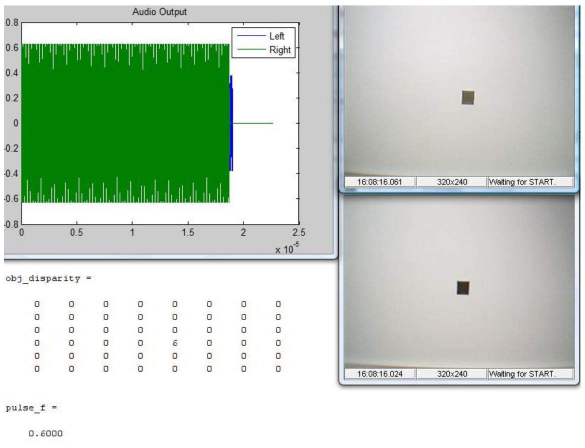
```
obj_disparity =

     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0
     0     0     0     0    10    10     0     0
     0     0     0     0    28    19     0     0
     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0


pulse_f =

    1.6750
```

Figure 29 - Object at 1.00m



```
obj_disparity =

     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0
     0     0     0     0     0    10     0     0
     0     0     0     0     0    11     0     0
     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0


pulse_f =

    1.0500
```

Figure 30 - Object at 1.25m



```
obj_disparity =

     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0
     0     0     0     0     6     0     0     0
     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0


pulse_f =

    0.6000
```

Figure 31 - Object at 1.65m

The system is more sensitive to near objects, because the change in disparity per distance is larger. A distant object has very little disparity change as the distance is changed, and therefore will less accurately map the depth of a far away object.

D. Future Work

Our system was designed with the assumption that only one object was in the field of view. When our cameras were presented with multiple objects, the algorithms do not process the information properly. However, if some adjustments were made to the object disparity algorithms we believe that support for multiple objects is achievable.

Ideally, template matching would allow us to have a noisy background because the background scenery will have little to no disparity, and thus can be filtered out. However, our initial thresh holding stages reduce the entropy of target sub images and allow for non-unique disparities for rows with multiple objects. These thresh holding stages allow for more robust single object detection as a design trade-off.

Alternatively, for the problems encountered by the background we believe a method like the one presented by Richard Graetz and Kris Smeds in their “Visual Object” presentation [10] could provide us a solution to our reliance on a clean white background.

VI. Conclusion

In this paper we successfully developed a novel way of detecting an object in the field of two stereovision sources by using the methods of template matching and cross-correlation and mapping it to a three dimensional audio source.

The object processing block was successfully able to distinguish an object in the cameras fields of view and determines its X,Y,Z co-ordinates using the methods of template matching and cross-correlation across the cameras. This information was then inputted into our audio processing block which converted this information into an audio signal with a variable frequency, fade and amplitude as shown in Figures 18-31.

In conclusion we provided proof of concept that our three dimensional visual to audio mapping system can achieve our objectives and with a little fine tuning we can create a robust system that can track and output multiple objects in a real-life environment.

VII. References

[1] Sokolova, Katrina, and Barak Shilo. "Experiments in Stereo Vision." 12 Dec. 2006. California State University. 12 Nov. 2008.

[2] Silva, Jorge A., Aurelio J. Campilho, and Marque Santos. 3-D Data Acquisition and Scene Segmentation System 3 (1996): 563-67.

[3] Cheng, Binbin, and Hai Zhang. "A computational Model for bats' echolocation based on mammalian auditory system." A computational Model for bats' echolocation based on mammalian auditory system (2008): 1436-440.

[4] Visual Perception. 11 Dec. 2008. Wikipedia. 11 Dec. 2008 <http://en.wikipedia.org/wiki/visual_perception>.

[5] Abugharbieh,. Rafeef. "EECE 466 Digital Signal Processing - Cross Correlation." University of British Columbia. Fall 2008.

[6] Wilson, Scott. "WAVE PCM soundfile format." EE367C Course Website. 20 Jan. 2003. Stanford University. 12 Dec. 2008 <http://ccrma.stanford.edu/courses/422/projects/waveformat/>.

[7] Mattoccia, S., F. Tombari, and L. Di Stefano. " Fast Full-Search Equivalent Template Matching by Enhanced Bounded Correlation." Image Processing IEEE Transactions on 17 (2008): 528-38.

[8] Template Matching 11 Dec. 2008. Wikipedia. 11 Dec. 2008 <http://en.wikipedia.org/wiki/Template_matching>.

[9] Duraiswami, Ramani, and Luo Yuancheng. "Canny edge detection on NVIDIA CUDA." Computer Vision and Pattern Recognition Workshops. College Park, MD. 23 June 2008.

[10] Graetz, Richard, and Kris Smeds. "Visual Object Tracking." EECE 466 Project Presentations. University of British Columbia. 27 Nov. 2008.

VIII. MATLAB Appendices

```
data = getsnapshot(vid);
data2 = getsnapshot(vid2);

datahsv = rgb2hsv(data);
datahsv2 = rgb2hsv(data2);

vidSizeX = 320;
vidSizeY = 240;
boxX = 40;  %must divide equaly into vidSizeX
boxY = 40; %must divide equaly into vidSizeY
boxNumX = vidSizeX / boxX;
boxNumY = vidSizeY / boxY;

targ_im = datahsv(:,:,2);        % left camera -> this image will contain the
targets
refr_im = datahsv2(:,:,2);       % Right Camer -> this image will be used as
reference

Fs = 44100 ; % Sample Rate of Output Sound
t = 1/Fs:1/Fs:1;
stereo_out = zeros(length(t),2);


obj_disparity = zeros(boxNumY,boxNumX);
depth_ok = [];
fade_delay = [];



for ii = 1:(boxNumY)
% ii=3;

    ytarg_start = (ii-1)*boxY+1;
    ytarg_end = ytarg_start+(boxY-1);
    target_row = max(targ_im(ytarg_start:ytarg_end, 1:320));
    target_row = single(im2bw(target_row, .2));

    yrefr_start = (ii-1)*boxY+1;
    yrefr_end = yrefr_start+(boxY-1);
    refr_row = max(refr_im(yrefr_start:yrefr_end, 1:320));
    refr_row = single(im2bw(refr_row, .2));


    for jj = 1 : boxNumX

          xtarg_start = (jj-1)*boxX+1;
          xtarg_end = xtarg_start+(boxX-1);
          target = single(target_row(xtarg_start:xtarg_end));

          [corr_val1,lag_l] = xcorr(target,target_row);
          [corr_val2,lag_r] = xcorr(target,refr_row);
          corrLag_L = [corr_val1' lag_l'];
          corrLag_R = [corr_val2' lag_r'];

          max_corr_val1 = max(corr_val1);
          max_corr_val2 = max(corr_val2);

          Lag_L = 0;
          Lag_R = 0;

          for zz = 1:length(corr_val1)

              if(corrLag_L(zz,1) == max_corr_val1)
                  Lag_L = corrLag_L(zz,2);
              end
              if(corrLag_R(zz,1) == max_corr_val2)
                  Lag_R = corrLag_R(zz,2);
              end
          end
%           end

          obj_disparity(ii,jj) = Lag_R-Lag_L;

         %create sound for this box
         if(obj_disparity(ii,jj) > 0 && obj_disparity(ii,jj) < 100)

            depth_ok =  [depth_ok obj_disparity(ii,jj)];


            fade = jj/boxNumX;
            fade_delay = [fade_delay fade];
            pitch = (boxNumY-ii)/boxNumY * 400 + 125;


            signal_L = sin(2*pi*pitch*t +.2*(0.5-fade));
            signal_R = sin(2*pi*pitch*t +.2*(0.5-(1-fade)));


            stereo_out = stereo_out + [signal_L' signal_R'];
%             wavplay(stereo_out,Fs)
         end
    end

end

pulse_f = mean(depth_ok)/10;
mean_fade = mean(fade_delay);
depth_pulse_L = 0.5*square(2*pi*pulse_f*t +.2*(0.5-mean_fade))+0.5;
depth_pulse_R = 0.5*square(2*pi*pulse_f*t +.2*(0.5-(1-mean_fade)))+0.5;
stereo_out = stereo_out.*[(depth_pulse_L*(1-mean_fade))'
(depth_pulse_R*(mean_fade))'];
 wavplay(stereo_out,Fs)

    figure(3)
plot(stereo_out)
```