Reverse Engineering the Kinect Stereo Algorithm

Erick Ball
Greg Taschuk

Abstract

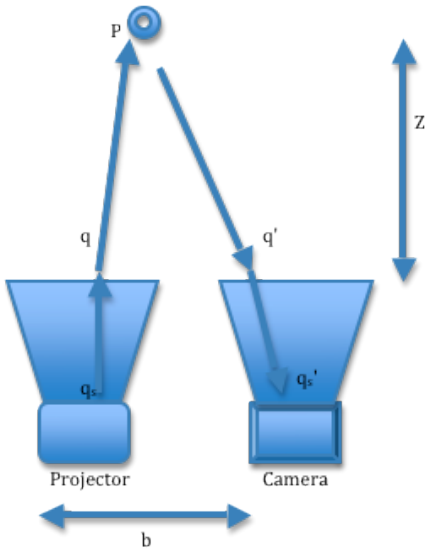
In this lab we reverse engineered the algorithm used to find disparity and depth and set up an approximation of the Kinect using a conventional, visible-light, projector. An algorithm was developed that compared two images – an image of dots projected into a scene and the dots that were projected into the scene. The algorithm was then able to calculate the disparities of objects' representation in the two images and interpolate distance. First, we were able to produce quality renderings of a few sample projector images provided for the purpose of this lab and then we were able to create a stereo vision setup similar to the Kinect with a visible light projector. We were able to generate surprisingly good renders of the scenes captured by our version of the Kinect.

Introduction

It is often very useful to have some information about the relative and absolute depth of objects in a scene. The most important biological mechanism for this is stereo vision and a similar method can be applied to computer vision. The disparity of an object in two image frames can be compared to the distance between two eyes to approximate distance of the object from the viewer. Recently, Microsoft released an implementation of stereo vision for the Xbox game system called the Kinect. The Kinect produces a depth-map of a scene by measuring the disparity between a number of dots projected with an IR-projector and their image, projected onto the scene, as captured with an IR camera. In this lab we developed an approximation of the Kinect by developing a similar algorithm and simulating the IR camera/projector with a visible light camera/projector.

Theory

The Kinect stereo algorithm uses an infrared projector that projects a known, near-random pattern of bright dots onto its field of view. An infrared camera, offset by 2.5 centimeters from the projector on its x-axis (but oriented the same way), images the same area. Because the view angle is slightly different from the projection angle, the pattern of dots in the camera image will be shifted slightly from that in the projector image, and the amount of shifting depends on the distance from the camera to the surface reflecting the light. This is very similar to the technique of stereo vision in general.



If a light ray leaves a point q on the normalized projector image plane and bounces off an object at point P, it strikes a point q' on the normalized camera image plane (Figure 1), where the camera is offset by a distance b. In general, the transformation from the point

$$P = \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

in space to the point

$$q_s' = \begin{bmatrix} x' \\ y' \\ w' \end{bmatrix}$$

in the camera sensor image is $q_s' = M'P$, where

$$M' = K'\begin{bmatrix} R' & t' \end{bmatrix}.$$

Similarly, to get q_s in the projector image, we use $q_s = MP = K\begin{bmatrix} R & t \end{bmatrix}$. R and R' describe the rotations of the projector and camera, and since the projector and camera have the same orientation, they are identity matrices. t is the projector position,

$$\begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

and t' describe the position of the camera—in this case,

$$t' = \begin{bmatrix} b \\ 0 \\ 0 \end{bmatrix}.$$

K' is the camera's intrinsic parameter matrix, which we take to be

$$K' = K = \begin{bmatrix} f & 0 & u_0 \\ 0 & f & v_0 \\ 0 & 0 & 1 \end{bmatrix},$$

This assumes the projector and camera have the same focal length *f* and optical center (u_0, v_0). Multiplying out and dividing by Z, we get

$$\bar{q}_s = \begin{bmatrix} \frac{fX}{Z} + u_0 \\ \frac{fY}{Z} + v_0 \\ 1 \end{bmatrix} \text{ and } \bar{q}'_s = \begin{bmatrix} f\frac{X+b}{Z} + u_0 \\ f\frac{Y}{Z} + v_0 \\ 1 \end{bmatrix}.$$

We then define

$$\Delta = \bar{q}_s' - \bar{q}_s = \begin{bmatrix} fb/Z \\ 0 \\ 0 \end{bmatrix}$$

as the disparity, which measures the distance, in pixels, by which the camera image is displaced along the x-axis relative to the projector image. In a system like the Kinect, where the offset *b* is known and we can estimate the disparity by comparing the camera image to the projector image, we can use that result to create a depth map for the image using the formula $Z = \frac{fb}{\Delta}$. Knowing the Z-coordinate for a point in the sensor image makes it possible to determine X and Y as well, so that if desired we can project the image into three dimensions.

Procedure

Our algorithm translates the camera's image of the projected dots into a depth image by calculating the disparity between the two images at every pixel. Disparity estimates come from a matching algorithm that compares a small window around the pixel in question to nearby, but shifted, locations in the projector image. In each case, we use the OpenCV MatchImage function to determine the correlation between the two image segments. We continue shifting along the x-axis over a search range that depends on the minimum depth (maximum disparity) needed for the image—in the simulated Kinect images, 16 is sufficient. Figure 2 shows an expanded view of the search zone (a portion of the projector image) and the template (a segment of the camera image) with corresponding points lined up.

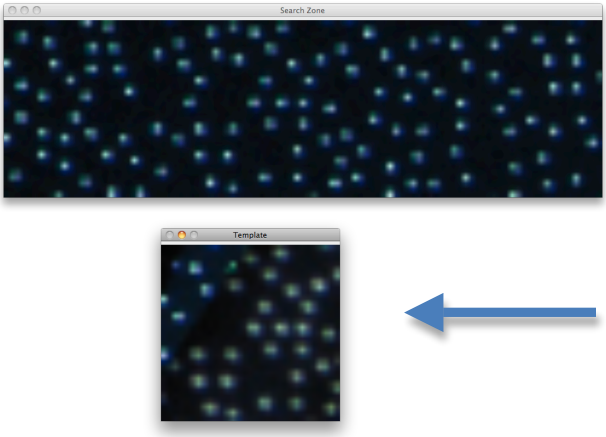


Figure 1: Calculating disparity by matching image segments. The blue arrow shows the disparity of the pixel at the center of the template window (the distance it shifted to line up with its match in the search zone).

The disparity is the number of pixels shifted for the case with the highest correlation, since those two image segments match up well except for a horizontal translation. Increasing the size of the comparison window drastically improves the accuracy of the matching algorithm, but also takes much longer to run.

To reproduce the functionality of the Kinect algorithm, we further refine the precision of the disparity calculation by interpolating in the images to get sub-pixel shift measurements. We blow up the image horizontally by a sub-pixel factor, usually 2, 4, or 8, using a linear interpolation. The creation of the depth map is the same, except that the search range must increase by the same factor to maintain the same range of depth, as must the width of the comparison window.

To improve efficiency in building the depth map, we also implemented a system to trade detail for time, by declining to calculate disparity for every pixel. Instead, we calculate the disparity of just one pixel in an n x n square, and set the estimated disparity of the entire square accordingly. We thus reduce the running time of the main portion of the algorithm by n^2. This can produce acceptable results with n as high as 8.

Finally, in some cases we perform post-processing on the image (open, close, and/or Gaussian blur) to reduce noise.

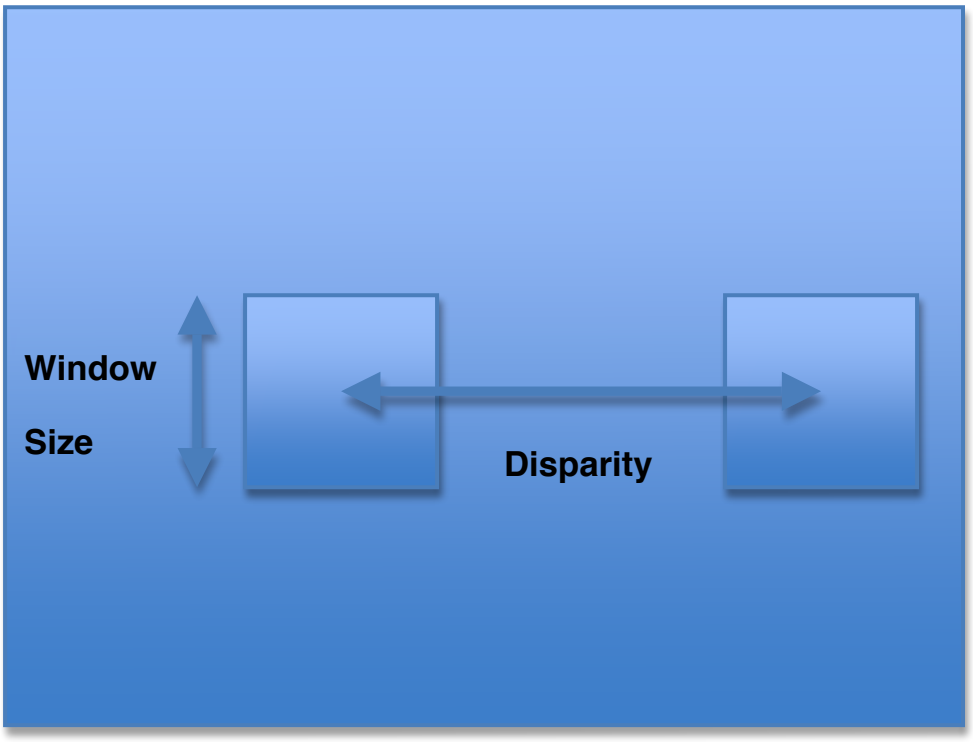
In order to demonstrate the ability of the algorithm to generate depth images from real projected dots, we created a rudimentary approximation of the Kinect using a visual light projector and a regular camera.

First we projected a projector image – a randomly positioned image of dots (white on black background) onto the scene. The projector in the Hick's Hall mural room was used. A simple point-and-shoot camera was set up directly below the projector.

An image of the projected dots on the projector screen was captured.

Objects were placed between the projector and the screen and more images were captured.

The disparity between corresponding windows of dots was used to produce a depth map of the scene.

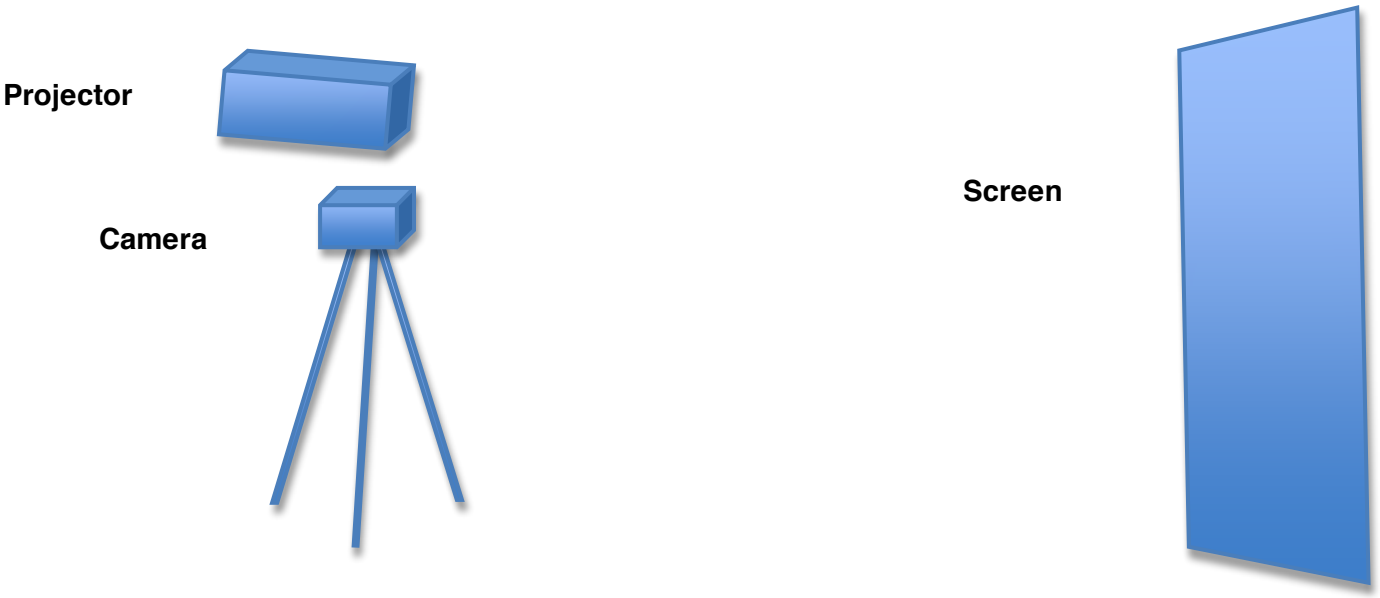


For each image we computed a low-quality (faster processing) and high-quality (slower) render. The parameters used for each are contained in the table below.

	Low Quality	High Quality
Subpixels	1	4
Skip Size	8	1
Window size	9	17
Threshold for min correlation	.8	.8
Openings	4	0
Closings	4	0
Gaussian blur size	0	0

Subpixels	degree to which comparison image was stretched horizontally, to provide greater accuracy of disparity values
Skip size	calculated correlation for every n^{th} pixel in every n^{th} row.
Window Size	Size of window to be compared
Threshold	Minimum correlation needed to consider two regions a match
Opening	Number of times disparity and z-map image were Dilated and then Eroded
Closing	Number of times disparity and z-map image were Eroded and then Dilated

Gaussian Blur Size of Gaussian blur window



Setup of our simulated “Kinect”

Results

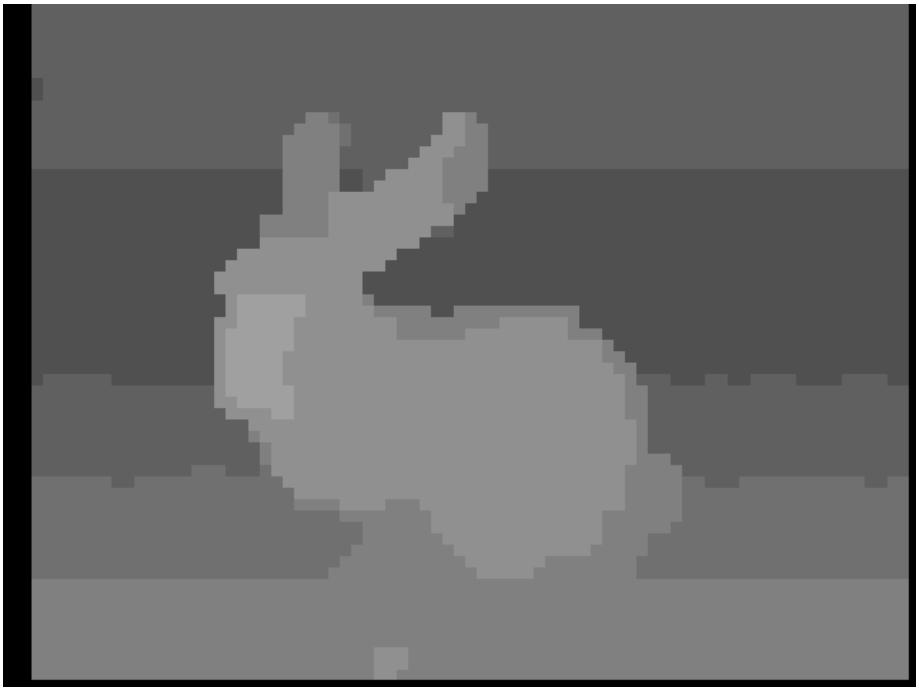


Figure 2a – Rabbit - Low Quality

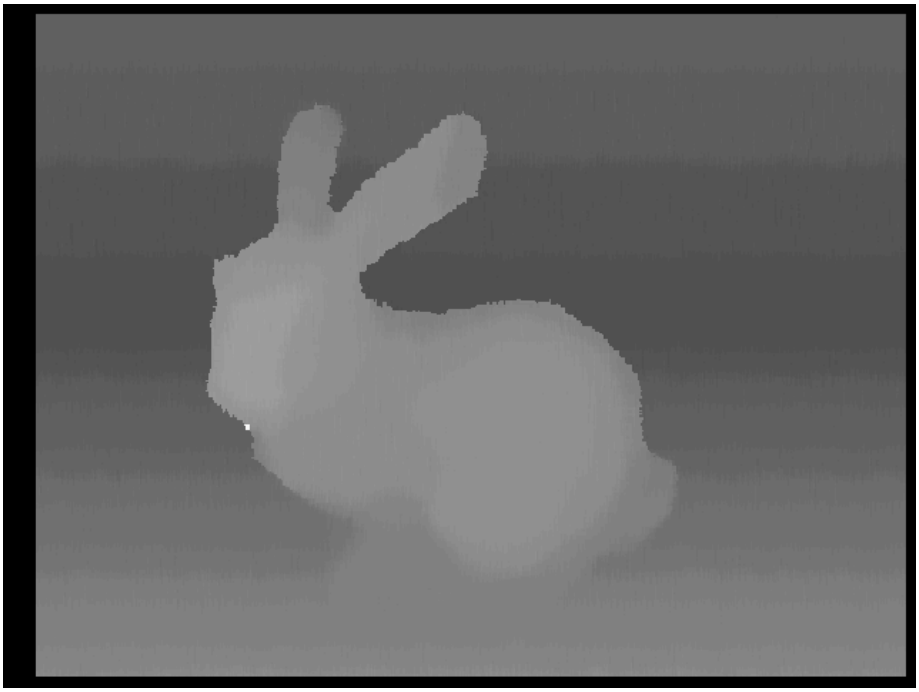


Figure 1b – Rabbit - High Quality

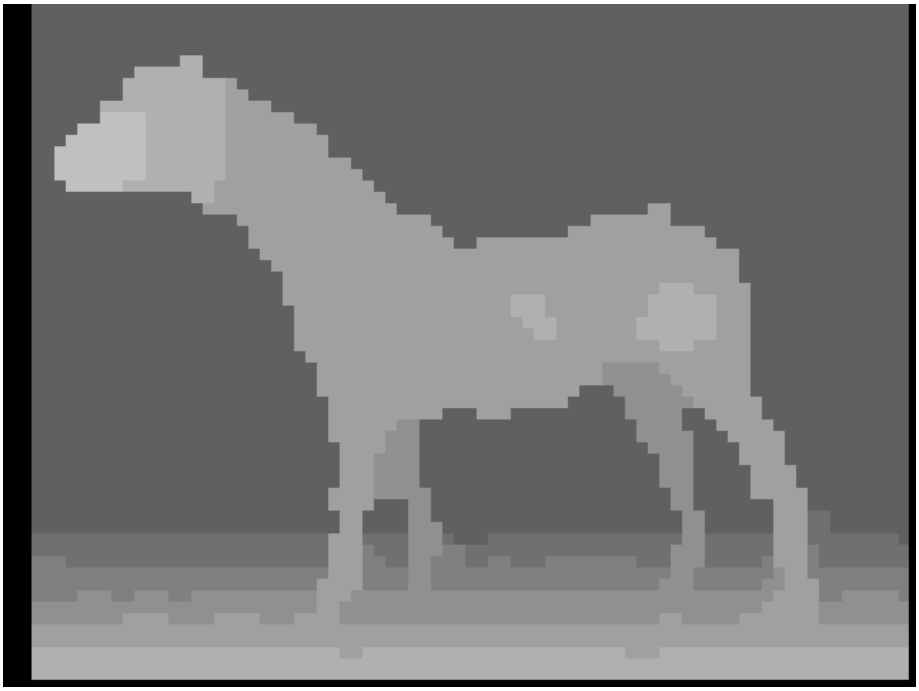


Figure 3a – Horse - Low quality

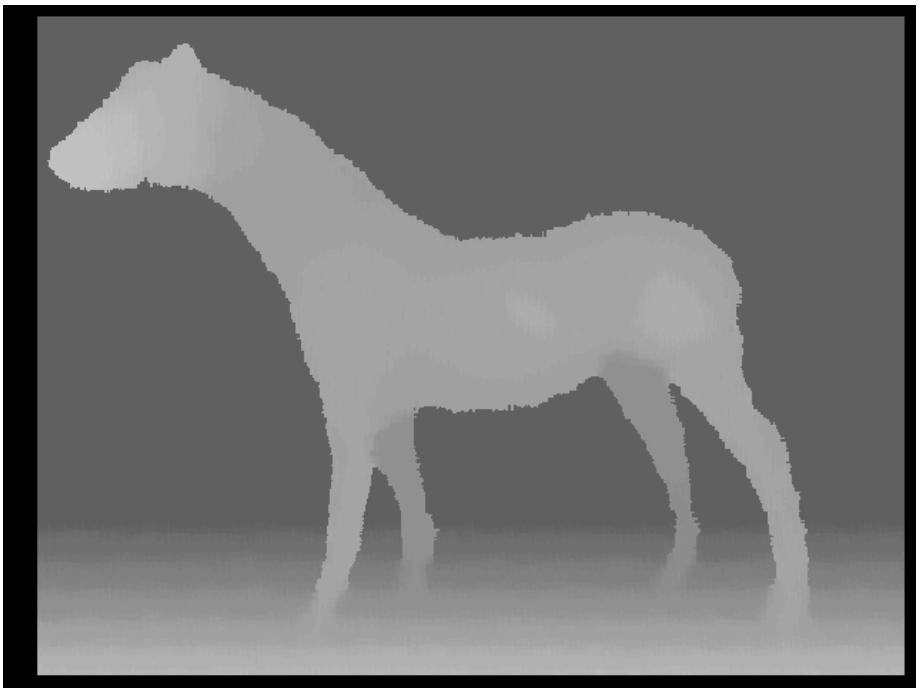


Figure 2b - Horse - High Quality

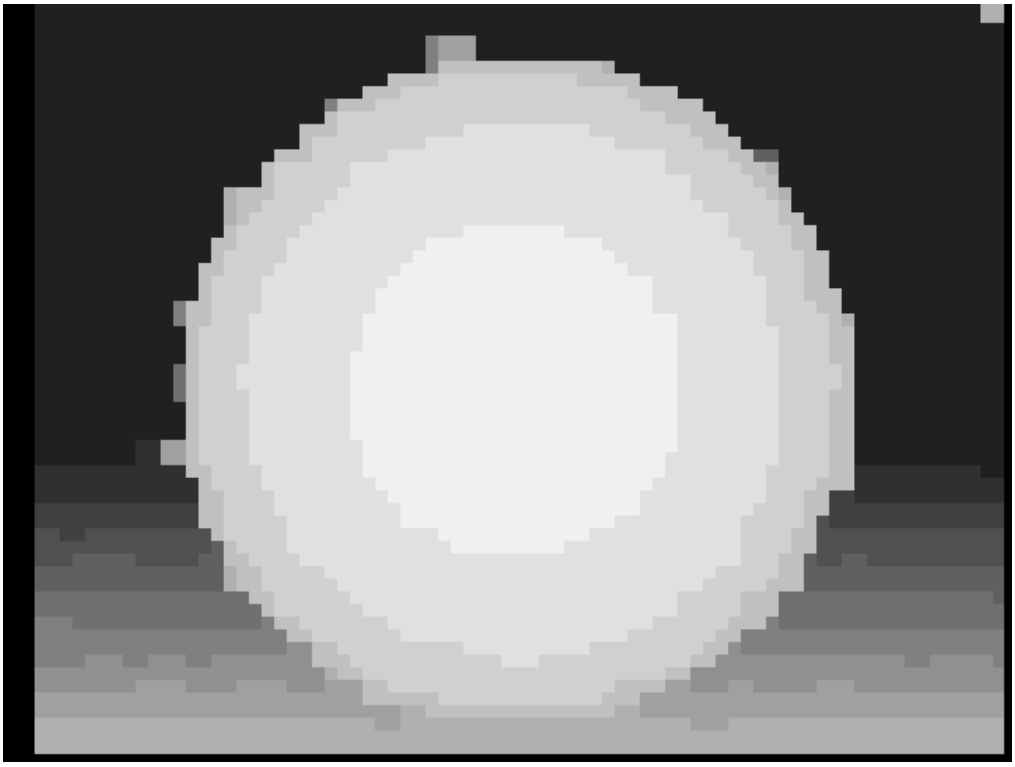


Figure 4a - Sphere - Low Quality

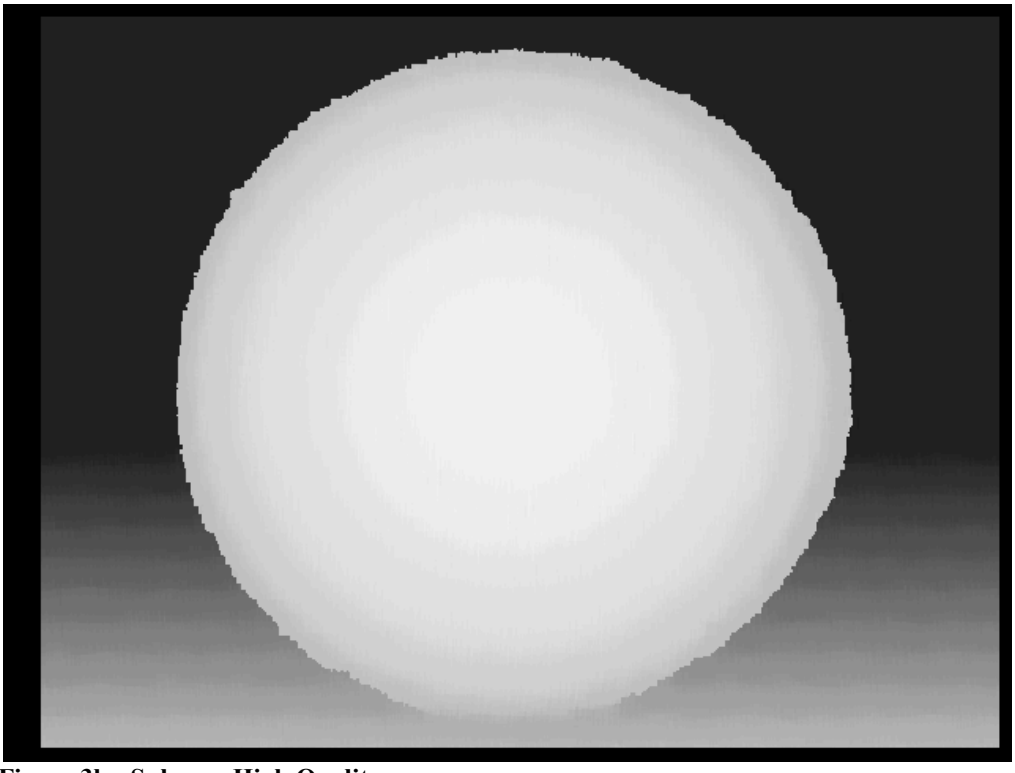


Figure 3b - Sphere - High Quality

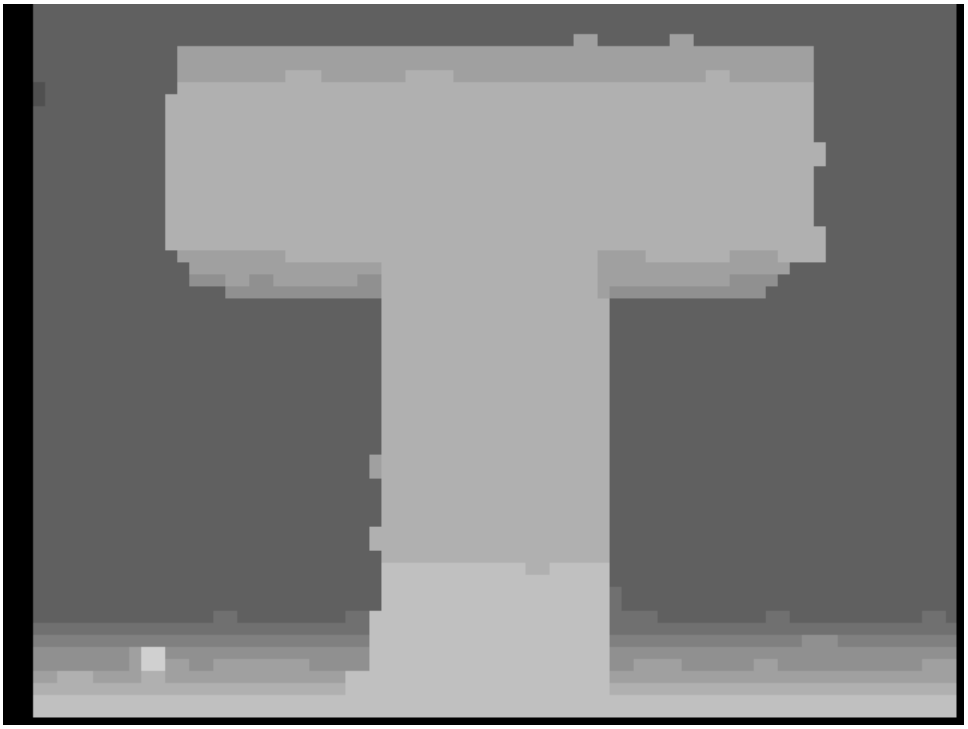


Figure 5a - "T" - Low Quality

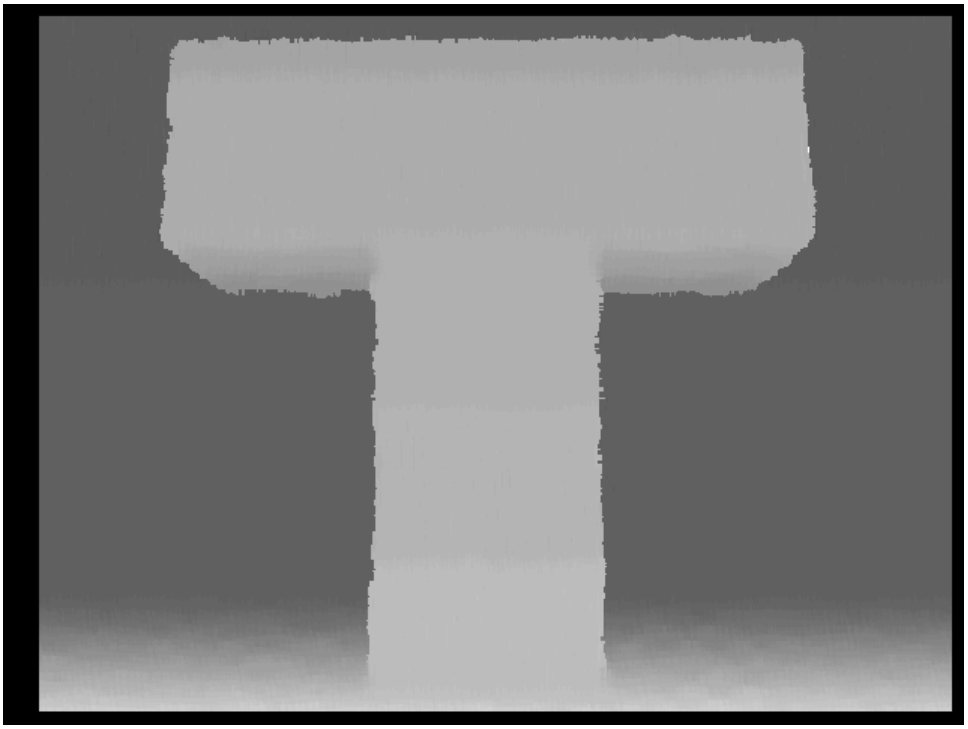


Figure 4b - "T" - High Quality

Figure 5a - Camera Image

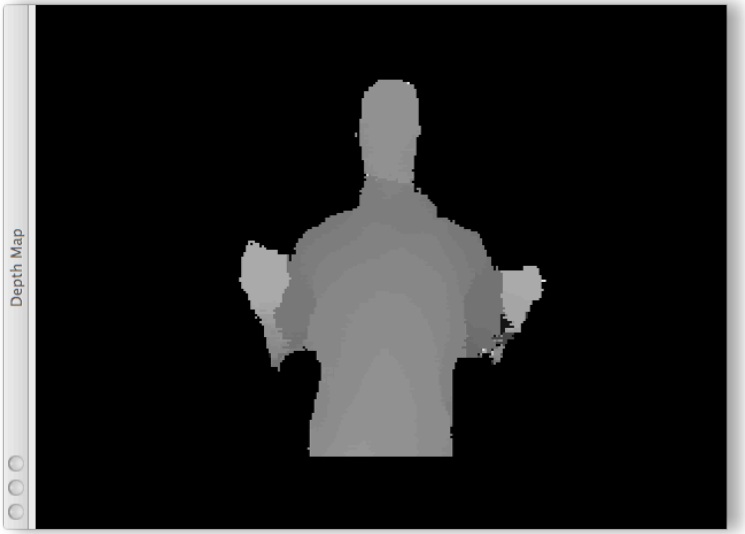


Figure 5b - Disparity map

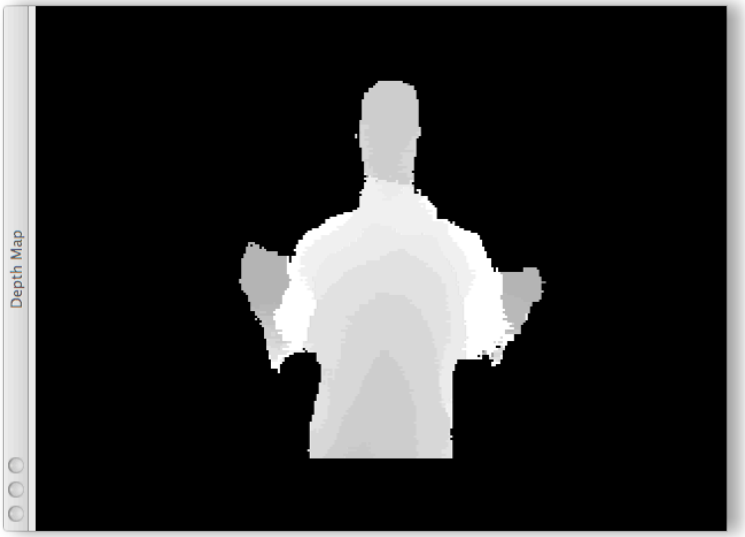


Figure 5c - Depth Map

Figure 6a - Camera Image

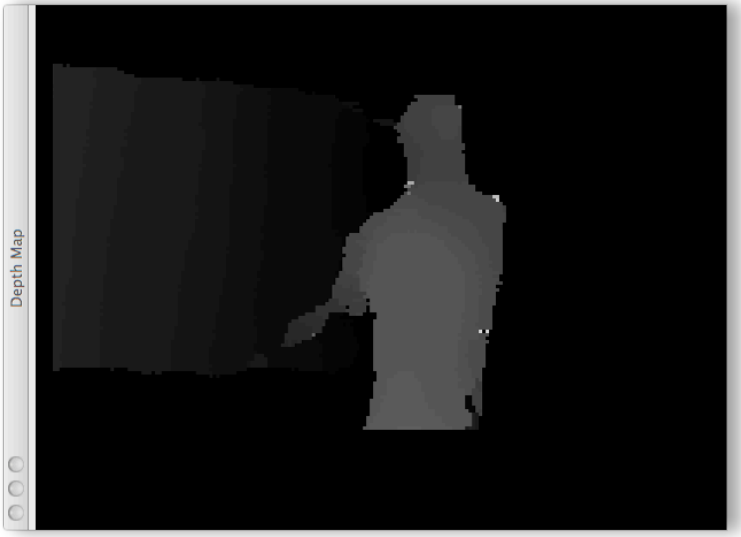


Figure 6b - Disparity mapping

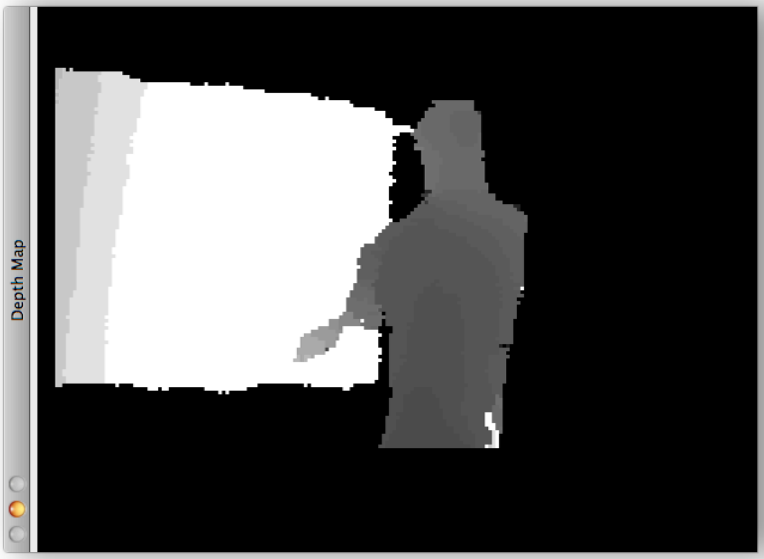
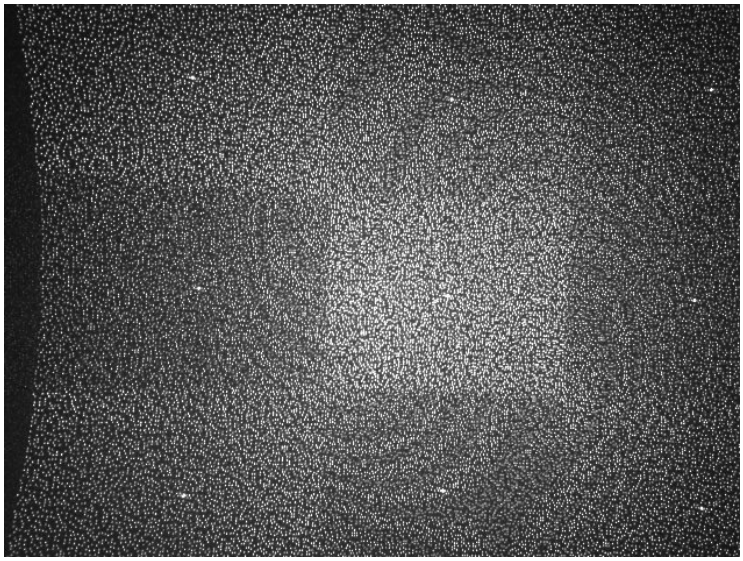


Figure 6c - Depth Mapping

Discussion and Conclusion

The quality of the depth image produced seems to depend most on the sub pixel filtering (although a factor of 8 provided little improvement over a factor of 4). Of course the skip size decreased the accuracy of the algorithm – we included it in the algorithm in order to reduce runtime as it reduces the runtime by a factor of n^2. Additionally a larger window size seemed to increase the quality of the depth image and reduced the noise. For the lower quality images we were able to decrease the amount of noise by "opening" and "closing" the image. However, increasing the window size increased the runtime of the algorithm considerably.

Our approximation of the Kinect worked surprisingly well. There was relatively low noise (with almost no noise in the background) indicating that, despite imperfections in the camera sensor, and slight shifting of the camera between shots, the images aligned quite well and the depth image is a good representation of the actual depths present in the scene. This was quite remarkable because, as mentioned in class, much of what enabled the Kinect's technology was not any revolutionary technological breakthrough but increased accuracy of hardware manufacture. Also, the Kinect uses a more sophisticated algorithm for placing the IR dots. Our spread of dots was purely random. The Kinect uses a pattern generated from a set of diffraction gratings and a more sophisticated comparison algorithm. The pattern used by the Kinect is pictured below.[1]



The one caveat of our setup is that, unlike the Kinect, both scene image and object free image were captured from the same location. This places the depth of the screen at infinity whereas in the Kinect, there exist disparities at all depths. The screen effectively has infinite depth in our model because the image is the same with or without objects, as the camera images of regions not containing objects were virtually identical in an image of the screen only and scenes with objects. In this case the relationship between disparity and depth is summarized as $\Delta_{object} - \Delta_{screen} = fb/Z_{screen} - fb/Z_{object}$. Where the disparity of the dots on the screen in the two images is 0 and Z_{screen} is the known distance from projector to screen.

As expected, there was a relatively short range of depths for which the depth perception worked well (close to the screen.)

Obviously, our "Kinect" would not be a very good replacement for the real thing. The obvious limitation is that it would not perform well in the light. A similar but more flexible technique might be other methods of pattern matching in a pair of stereo images. While object recognition is generally considered difficult, being able to identify an object in each image frame could allow them to be matched so that disparities and depth maps could be developed. Also, movement may be a good cue for object matching in pairs of stereo images.

Acknowledgments

Special thanks to Professor Zucker and the media center (for providing a tripod) and to the fine folks at ROS.org for their informative page on the Kinect.

[1] http://www.ros.org/wiki/kinect_calibration/technical