Creating Depth Maps from Monocular and Stereoscopic Images

By: Ben Bodner, Jessica Coulston

Advisor: Ricardo Figueroa

Revision Date: May 28, 2012

Submitted in partial fulfillment of the requirements for
the BS degree in the School of Film and Animation
Rochester Institute of Technology, Rochester, NY

0.1 Abstract

The removal of certain portions of an image based on content is known as keying and is an important component of most digital effects work in the Motion Picture Industry. The two algorithms proposed here determine the depth of objects in an image, creating a channel with depth information at every pixel that can be used to key out portions of the image based on their distance from the camera. Keying from depth information reduces the amount of preparation necessary for shooting as compared to traditional chroma keying and has a wider range of uses than luma keying. The use of an accurate depth channel would also allow for insertion of computer generated (CG) content into live action scenes without having to manually select what portions of the scene should occlude the CG content. One algorithm produces depth channels from stereoscopic pairs of images, captured by a dual-camera rig designed for shooting 3D movies, and the other uses single monocular images. The monocular algorithm was ultimately unsuccessful because of problems with the machine learning implementation but the method still retains merit for its simplicity. The stereoscopic implementation found accurate depth in many cases and had several clear advantages over chroma keying, but still had a lot of errors in depth estimation. The implementation has significant room for improvement.

0.2 Introduction

The most common forms of keying are chroma keying, which removes pixels based on color, and luma keying, which removes pixels based on luminance. Chroma keying typically requires the use of a green screen background, along with a significant amount of lighting expertise and planning. The use of chroma keying also limits available colors for foreground objects, in order to avoid unwanted removal of object surfaces. Luma keying is limited to usage on very bright or very dark portions of an image, and is typically used for sky replacement.

The goal of the project was to create depth maps from both two-dimensional (2D) and three-dimensional (3D) captured images for the purpose of depth keying. Depth keying, unlike chroma keying, allows the user to capture an image anywhere, not just in front of a green screen, which makes lighting and set composition much less limited. It also has the possibility of eliminating common problems like fringing and spill.

This paper includes a detailed analysis and discussion of implementation of one approach to depth map creation for both 2D and 3D. Many available methods were assessed and the most promising for each capture type were chosen for implementation. The depth keys have been assessed based on the accuracy of depth information and quality of key as compared to light detection and ranging (LIDAR).

0.3 Background

A significant amount of research and documentation exists on creating depth maps from stereo image pairs. The process typically involves identifying matching pixels between the two images and using their disparity in combination with the cameras' orientations to find depth. It is also necessary to determine which regions of one image do not have a matching region in the other image, due to the differing perspectives of the cameras, and determine how to interpolate depth information in these areas.

Most methods of creating depth maps from single 2D images rely on auxiliary devices to either determine depth independent from the camera or to add information to the scene to be used to help determine depth after image processing. For instance, one method analyzes the frequency of a uniform static stripe pattern projected onto a scene to create a depth map.

The monocular method discussed here is ideal because it does not require an auxiliary device, and relies on traditional monocular image capture alone. This method analyzes variations in texture across the image and how those textures are affected by scaling along with comparison of other features in the image.

0.4 Theory and Method

0.4.1 Depth Maps from Stereoscopic Capture

To create a depth map from a stereoscopic image pair, each pixel in one image must be matched to the corresponding pixel in the other image. Each point in the scene should have a corresponding pixel from both cameras, except in cases where a pixel has imaged scene content that is occluded in the other image. Because the two cameras used to record the scene were arranged horizontally, corresponding pixels will only differ in location horizontally and not vertically, limiting the search for a matching pixel to a single line of pixels in the image, called a scan line.

To search for the corresponding pixels from both cameras, an area of pixels, called an image block, is centered on the pixel under scrutiny in the first image and then compared to image blocks in the second image. The comparison is done using the luminance version of the images, and cross-correlating image blocks along the same scan line. The horizontal distance between two image blocks with high correlation is the disparity between the two blocks. When looking at the correlations between one block in the first image and the blocks on the same scan line in the second image, there will likely be multiple cases of high correlation. To narrow down which correlation actually represents matching image blocks, it must be considered that objects have low variations in disparity across their surface, but large changes in disparity at their edges. Neighboring blocks in the first image should, in most cases, represent different portions of the same object in the scene, and thus neighbors should have approximately the same disparity, determined by their horizontal distance from highly correlated blocks in the second image. So while one block in the first image may have a high correlation with multiple blocks in the second image and thus more than one probable disparity value, all neighboring blocks representing the same object should have one probable disparity value in common.

Figure 1 shows how the disparity of continuous objects can be identified, with the horizontal axis representing horizontal image block location in the first image and the vertical axis representing the disparity (the horizontal distance to the image block in the second image), with bright areas having high correlation between the two image blocks. Although every single block from the first image has a high correlation to multiple blocks in the second image, bright horizontal lines form on the graph where there is a common probable disparity between neighbors, representing a continuous object in the scene [1].

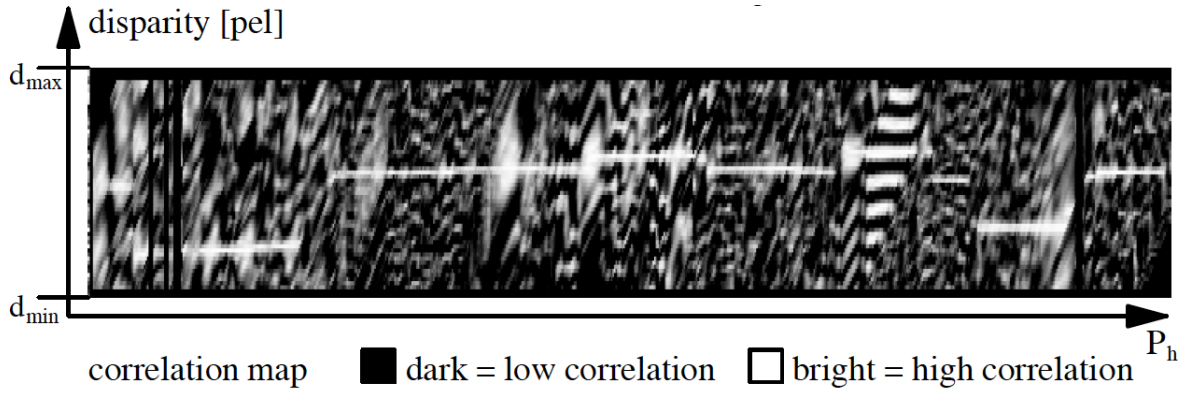


Figure 1: Correlation map showing horizontal image block location versus disparity [1].

To create dense depth maps, more than just an approximate disparity for each object is necessary. Specifically for depth keying, accurate edge locations are necessary. To further determine a more accurate depth at each pixel, candidate pairs of pixels from the same object are determined from the correlation data found in the previous step. A cost function is employed to determine which candidate is actually a matching pixel, most importantly at object edges, and which pixels have no matching candidate because they represented an occluded portion of the image [2].

A cost is assigned to each possible disparity at every pixel location on the scan line, so every point in the correlation map shown in Figure 1 has a cost. Each point in the leftmost column of the correlation map is assigned a cost, C_{match}, which is dependent on the normalized cross-correlation at the point. In each of the remaining columns, cost is determined for each point dependenting on the costs in the preceding columns. For each point, a series of possible costs is generated, with one possible cost for every point in the preceding column, and the minimum cost is assigned to the current point. In other words, all possible disparity values at the previous pixel location are considered and the one with the lowest cost is chosen.

If a point in the preceding column has a lower disparity than the point that is currently being assigned a cost, that point in the preceding column is not considered. Instead, a point several columns to the left is considered, with exact column location determined by the change in disparity. This is because a disparity increase, a change from an object in the background to an object in the foreground, will correspond with pixels in the right image that are occluded in the left image. The exact number of pixels occluded is equivalent to the change in disparity. Figure 2 shows how the series of possible costs is determined.

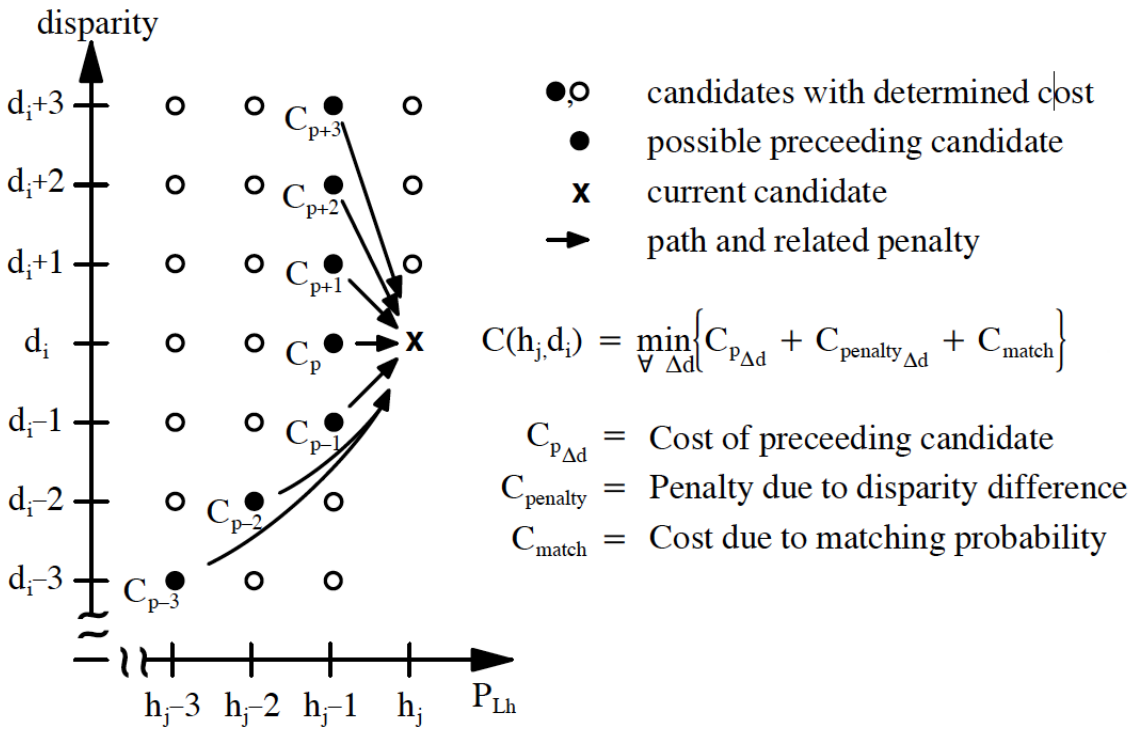


Figure 2: Computation of the cost map [1].

As shown in Figure 2, a possible cost for a current point is the sum of the cost at the preceding point, C_{match} at the preceding point, and a penalty dependent on the change in disparity between the preceding point and the current point. Equations 1 through 5, below, show different ways the penalty term is determined, dependent on the change in disparity.

$$\Delta d = 0 \quad \Rightarrow \quad C_{penalty} = 0 \tag{1}$$

$$\Delta d = \pm 1 \quad \Rightarrow \quad C_{penalty} = C_{inclination} = ln\left(\frac{P_{boundary}}{1 - P_{boundary}} \cdot \frac{1}{\sqrt{2\pi\sigma_n^2}}\right) \tag{2}$$

$$-\infty < \Delta d < -1 \quad \Rightarrow \quad C_{penalty} = C_{discontinuity} = a \cdot C_{inclination} \tag{3}$$

$$+1 < \Delta d < +\infty \quad \Rightarrow \quad C_{penalty} = C_{discontinuity} + \Delta d \cdot C_{occlusion} \tag{4}$$

$$C_{occlusion} \quad = \quad b \cdot C_{inclination} \tag{5}$$

The variable $P_{boundary}$ in equation 2 is a value between .95 and .98 that depends on the number of object boundaries in the image. Variables a and b in equations 3 and 5 have an optimal value depending on the content of the image. Ideally, a user would be able to view different depth map outputs with varying values of $P_{boundary}$, a, and b and then chose the best option. However, for this algorithm, a set of average optimal values were determined to be $P_{boundary} = .98$, $a = 6$, and $b = 8$.

When a minimum cost is chosen from the series of possible costs, the coordinates of the preceding

point that was used in determining the cost are stored at the current point, along with the minimum cost that was chosen. When costs have been determined for all points in the correlation map, a candidate from each column is determined to be the correct disparity, so that every pixel location is assigned a disparity value. First, the point with the lowest cost in the last column of the correlation map is assigned as the disparity value for the rightmost pixel, and the location that was stored at that cost is used to find the correct disparity in the second to last column. From there, the location stored at each disparity chosen is used to find the preceding correct disparity. This way, a disparity value is determined for every pixel location in the line, except where there was an increase in disparity and a corresponding occlusion.

The process of finding a correlation map for a row of pixels and then determining the correct disparities from that map using the cost function is repeated for every row in the image, so that a disparity value is found for every pixel except where there is an occlusion. Along with the gaps in disparity due to occlusion, the resulting disparity map has other inaccuracies. Foreground objects are expanded in all directions an amount dependent on the block size used in finding normalized cross-correlation. Errors in determining correct disparity with the cost function can lead to false changes in disparity. To correct for these issues, the disparity map is compared with the luminance map of the original image. Edge detection is used to find an edge map of both the disparity and luminance images. A line is drawn from each point in the disparity edge map to its nearest neighbor in the luminance edge map. If that line is shorter than a maximum distance dependent on the block size used to find the correlation map, all pixels in the disparity map along that line are changed to a value equal to the disparity, on the side of the disparity edge opposite the nearest neighbor.

This disparity correction has the effect of shrinking foreground objects to their proper size and replacing disparity gaps at occlusions, because disparity edges will be located outside luminance edges, and all values between the edges will be changed to the disparity of the background object. Figure 3 shows the correction process, where ‘a’ is a point on the disparity edge map and ‘b’ is its nearest neighbor in the luminance edge map. All pixels on the line between them in the disparity image are changed to the disparity value at the pixel adjacent to ‘a’ in the direction opposite of the line.

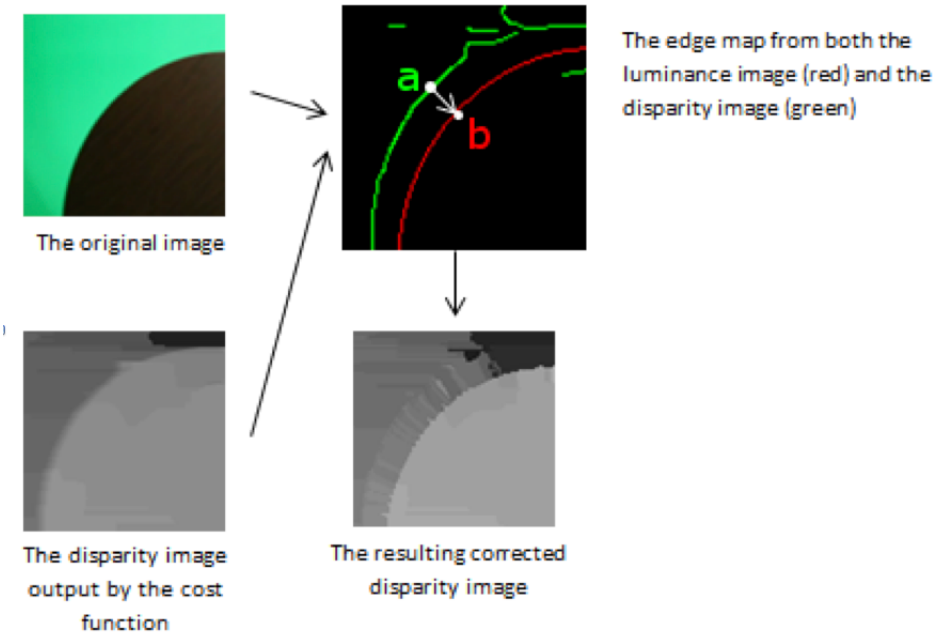


Figure 3: Process of disparity correction.

Once corresponding pixel locations have been identified between the stereoscopic image pair, the depth of the point in the scene represented by each pair of pixels can be determined geometrically using the disparity between corresponding pixels and the relative orientation and position of the cameras. Figure 4 shows the components necessary to determine depth. C is the optical center of the camera. A is a unit vector in the direction of the optical axis of the camera. H and V are unit vectors in the direction of the horizontal and vertical components of the image plane, respectively. f is the focal length of the cameras, so $-f * A$ is a vector from C to the center of the image plane [3].

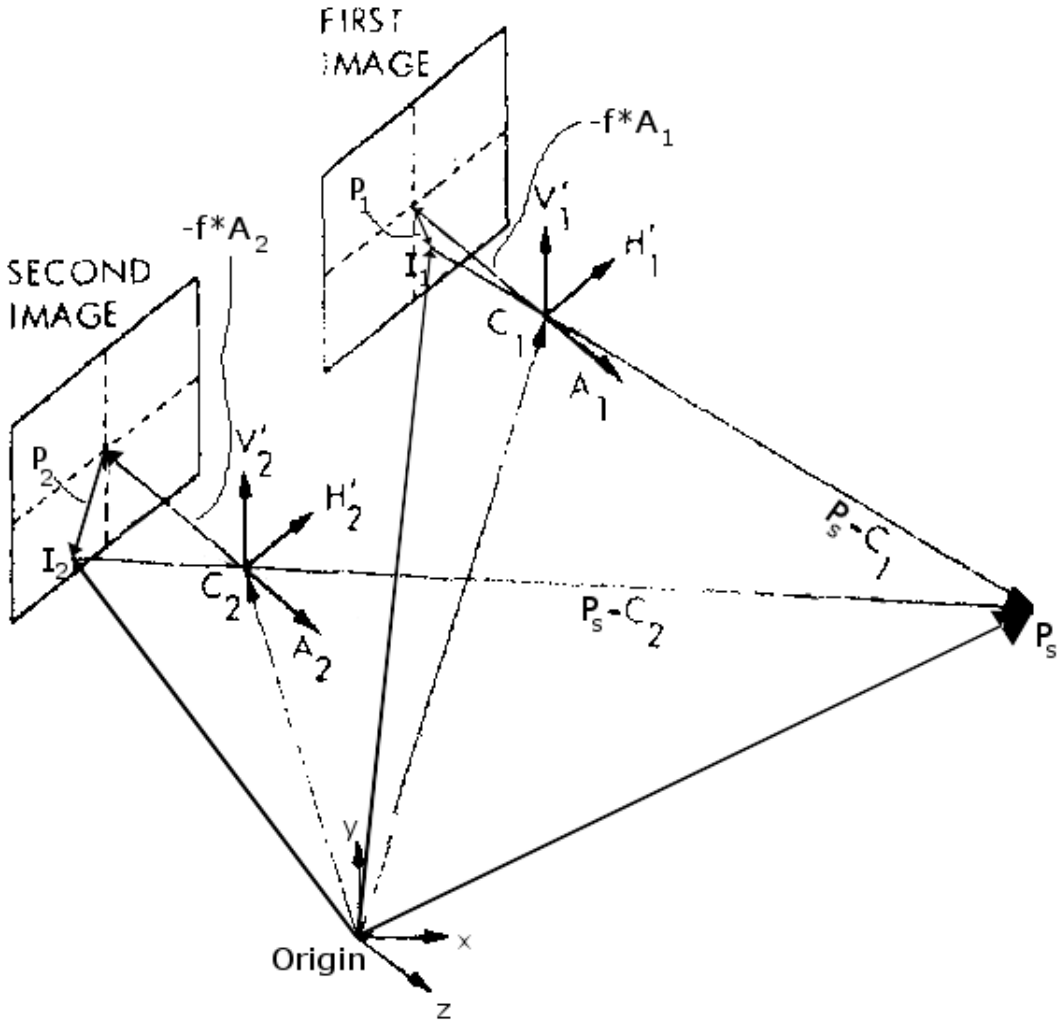


Figure 4: Geometry involved in determining depth from a pair of corresponding pixel locations [3].

P_S is a point in space and P is the projection of P_S onto the image plane, relative to the center of the image plane. Hence, P_1 and P_2 are the corresponding pixel locations previously determined by the algorithm (although adjusted so they are relative to the center of the image plane instead of the bottom-left corner). I_1 and I_2 are the pixel locations relative to the origin, and can be determined with equation 6:

$$I = C + (-f * A) + P_h * H' + P_v * V' \tag{6}$$

To simplify the problem, C_1 can be made the origin of the coordinate system, so A_1 is a unit vector along the z-axis, H'_1 is a unit vector along the x-axis, and V_1 is a unit vector along the negative y-axis. C_2, A_2, H_2 can be easily determined if the user supplies the focal length and the distance, d, between the two cameras and angle, Φ, between A_2 and A_1:

$$
\begin{aligned}
C_2 &= [d, 0, 0] && (7)\\
A_2 &= [\cos(\pi/2 + \Phi), 0, \sin(\pi/2 + \Phi)] && (8)\\
H_2 &= [\sin(\pi/2 + \Phi), 0, -\cos(\pi/2 + \Phi)] && (9)\\
V_2 &= V_1 = [0, -1, 0] && (10)
\end{aligned}
$$

To find the point in space projected onto P_1 and P_2, the location of minimal distance between the line L_1, which passes through I_1 and C_1, and the line L_2, which passes through I_2 and C_2, must be determined. In a perfect situation where I_1 and I_2 where found along a continuous image plane, P_S would be the location where these two lines cross. However, I_1 and I_2 must be in the center of a pixel, and so L_1 and L_2 likely will not cross. The location of minimal distance exists where a vector W from a point on L_1 to a point L_2 is perpendicular to both lines[4]. The distance, sc, from C_1 to the point where W and L_1 intersect can be found using equation 12 where:

$$
\begin{aligned}
u &= P1 - C1\\
v &= P2 - C2\\
w_0 &= C1 - C2
\end{aligned}
$$

therefore:

$$
\begin{aligned}
a &= u \cdot u\\
b &= u \cdot v\\
c &= v \cdot v\\
d &= u \cdot w_0\\
e &= v \cdot w_0
\end{aligned}
$$

and:

$$sc = (b * e - c * d)/(a * c - b^2) \quad (11)$$

With all the substitutions made, equation 11 is equivalent to equation 12 below.

$$s = \frac{\Big((P_1 - C_1) \cdot (P_2 - C_2)\Big) * \Big((P_2 - C_2) \cdot (C_1 - C_2)\Big) - \Big((P_2 - C_2) \cdot (P_2 - C_2)\Big) * \Big((P_1 - C_1) \cdot (C_1 - C_2)\Big)}{\Big((P_1 - C_1) \cdot (P_1 - C_1)\Big) * \Big((P_2 - C_2) \cdot (P_2 - C_2)\Big) - \Big((P_1 - C_1) \cdot (P_2 - C_2)\Big)^2} \quad (12)$$

The point where W and L_1 intersect, which is the approximate location of P_S, can then be determined:

$$P_S = C_1 + s * (C_1 - P_1) \quad (13)$$

The depth at pixel locations can now be found wherever a matching pixel was found. Depth where there was no disparity information must be interpolated based on disparity values at neighboring pixels.

0.4.2 Depth Maps from Monocular Capture

The method of depth analysis implemented here was created by Ashutosh Saxena, Sung H. Chung, and Andrew Y. Ng [5]. Using a single monocular image, 15 filters, a probabilistic model, and machine learning, the team was able to extract fairly accurate relative depth information.

Features for Absolute Depth

The method implemented here for gathering depth information from a single monocular image makes use of nine Laws's masks and six edge detection filters. The Laws's masks were presented by Kenneth Ivan Laws in 1979 and 1980. They are constructed from three basic 1x3 masks:

$$\text{L3} = \begin{bmatrix} 1 & 2 & 1 \end{bmatrix} \tag{14}$$

$$\text{E3} = \begin{bmatrix} -1 & 0 & 1 \end{bmatrix} \tag{15}$$

$$\text{S3} = \begin{bmatrix} -1 & 2 & -1 \end{bmatrix} \tag{16}$$

The letters at the beginning of these three masks stand for *L*ocal averaging, *E*dge detection, and *S*pot detection. By convolving the three masks together in every combination, the nine 3x3 Laws's masks are generated:

$$\mathbf{L3^T L3} = \begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix} \quad \mathbf{L3^T E3} = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \quad \mathbf{L3^T S3} = \begin{bmatrix} -1 & 2 & -1 \\ -2 & 4 & -2 \\ -1 & 2 & -1 \end{bmatrix}$$

$$\mathbf{E3^T L3} = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix} \quad \mathbf{E3^T E3} = \begin{bmatrix} 1 & 0 & -1 \\ 0 & 0 & 0 \\ -1 & 0 & 1 \end{bmatrix} \quad \mathbf{E3^T S3} = \begin{bmatrix} 1 & -2 & 1 \\ 0 & 0 & 0 \\ -1 & 2 & -1 \end{bmatrix}$$

$$\mathbf{S3^T L3} = \begin{bmatrix} -1 & -2 & -1 \\ 2 & 4 & 2 \\ -1 & -2 & -1 \end{bmatrix} \quad \mathbf{S3^T E3} = \begin{bmatrix} 1 & 0 & -1 \\ -2 & 0 & 2 \\ 1 & 0 & -1 \end{bmatrix} \quad \mathbf{S3^T S3} = \begin{bmatrix} 1 & -2 & 1 \\ -2 & 4 & -2 \\ 1 & -2 & 1 \end{bmatrix}$$

Figure 5: Laws's Masks shown numerically [6].

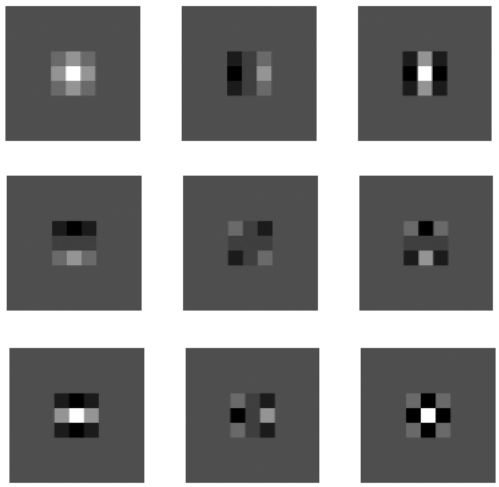


Figure 6: Laws's Masks shown pictorially [5].

All input images are represented in the YC_bC_r space. Formatted this way, the intensity channel, Y, holds most of the images' texture information, while the color channels, C_b and C_r, hold the low frequency information like haze. Haze refers to atmospheric light scattering, which is a monocular cue that humans rely on. The first Laws's mask, $L3^TL3$, acts as a local averaging filter and is applied to the color channels. The entire set of Laws's masks are used on the intensity channel.

The last six filters are simply oriented edge detectors at increments of 30° as shown in Figure 7. The intensity channel is convolved with each of the six edge detection filters. Usually edge

detection is done with a set of four or eight filters operating at 45° as in Sobel and Prewitt. Having six filters at 30°, however, reduces the redundancy of eight filters, while still improving on just four.

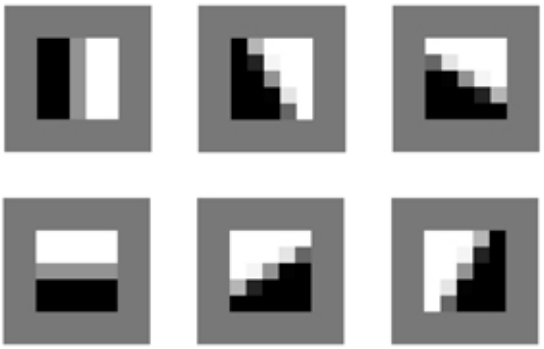


Figure 7: Six edge detecting kernels oriented at 30° [5].

The product of the intensity channel convolved with the nine Laws's masks, the two chroma channels convolved with the first Laws's mask (local averaging filter), and the intensity channel convolved with the six edge detecting kernels creates 17 output dimensions (F_n). Using equation 17, where $k = \{1, 2\}$, the sum absolute energy and sum squared energy respectively can be found by convolving the 17 output dimensions (F_n) with the original image, (I) . This creates the first 34 dimensions of the feature vector for each pixel.

$$E_i(n) = \Sigma_{(x,y)\in patch(i)} |I(x,y) * F_n(x,y)|^k \tag{17}$$

To capture larger, more global features, three scales, or image resolutions are created. The first scale exists at the pixel level and is represented in Figure 8 by the blue squares. The second scale, represented in Figure 8 in red, is produced by averaging the center blue pixel and its four adjacent blue pixels. Similarly, the third scale, represented in purple, is produced by averaging the center red patch from scale two and its four red adjacent patches. Just as equation 17 was used on each patch in the original image(I) to create the first 34 dimensions of each patch's feature vector, equation 17 is also used on each neighboring patch, at each scale. The surrounding patches' features are made a part of the center patch's feature vector, creating a total of 510 dimensions of information at each pixel.

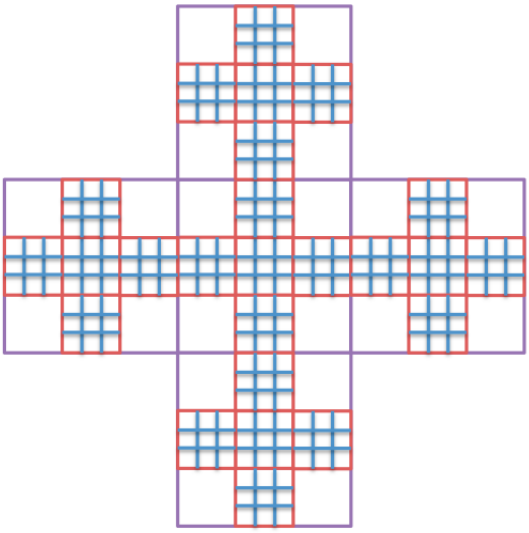


Figure 8: Scale 1 is blue, scale 2 is red, and scale 3 is purple.

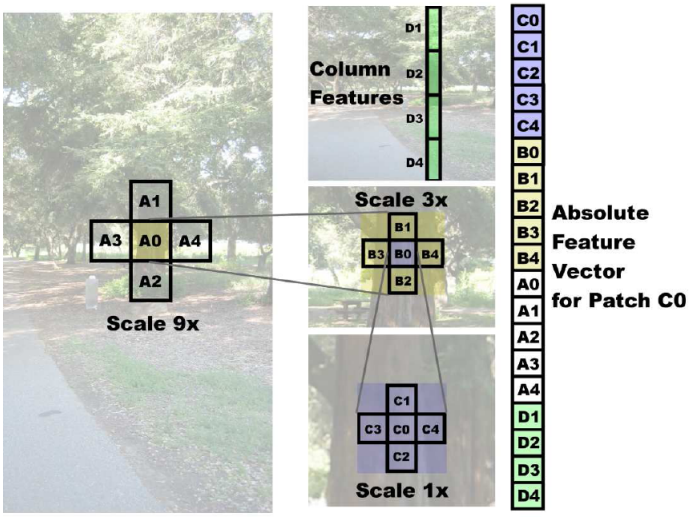


Figure 9: Illustration of scale creation and column features [5].

A vertical feature vector, computed from the pixels in the column of the center pixel, is also factored in to account for the very vertical structure of objects in the world (trees, buildings, furniture, people). The column that the center pixel resides in is divided into four sections, as seen in Figure 9, and the pixels in each quarter are averaged. Then, using equation 17, the 34 features for each quarter of the column are also included in the center pixel's feature vector. This adds an additional 136 dimensions to the 510 current dimensions, resulting in 646 dimensions of absolute depth information.

Features for Realitive Depth

The features for relative depth are employed to help learn the dependancies between depths at neighboring patches. At the largest scale, shown in Figure 8 in purple, each patch is nine pixels high by nine pixels across. For each of the 17 output images, (F_n), the patches at scale three are analyzed. Each patch is composed of 25 scale one pixels. A histogram of the code values present in the patch and a histogram of the code values present in the adjacent patch are created. The histograms for the 17 output images are quantized to 10 bins and then adjacent patch's histograms are subtracted, creating 170 features for relative depth. Figure 10 shows pictorially the process for creating features for relative depth.

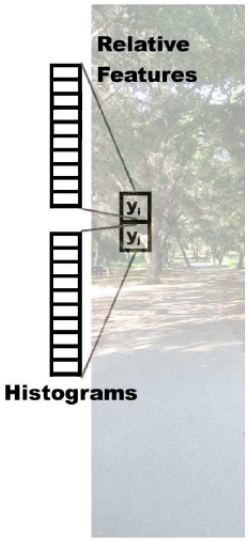
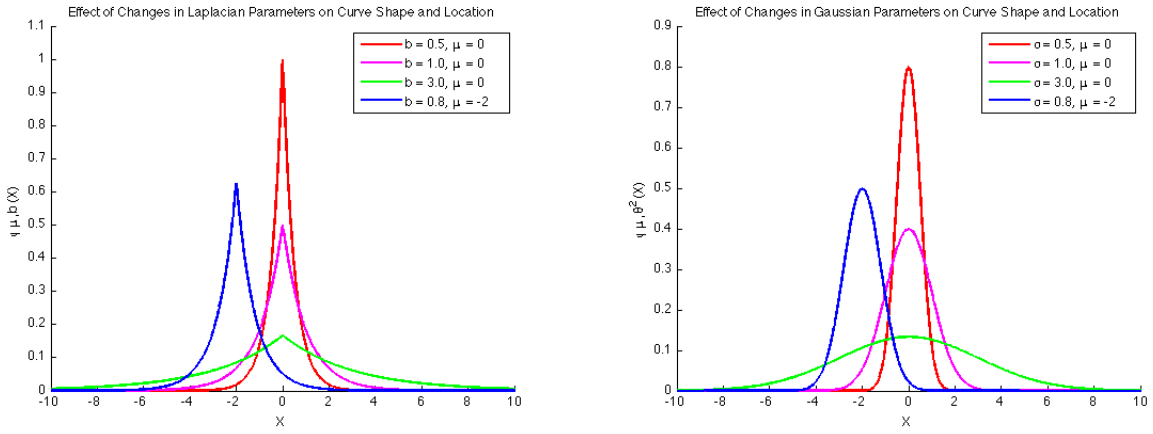


Figure 10: Relative feature acquisition, done at scale three on 10 bin quantized histograms [5].

Probabilistic Model

Two types of probabilistic models, Laplacian and Gaussian, are applicable for this implementation. Figure 11 shows the types of curves generated by each model. Both types can be used to successfully create a depth map but the Gaussian gives a less exact result. The difference in profile of the two curves illustrates the success rate. The Gaussian has a rounded profile, while the Laplacian comes to a sharp point. When calculating the probability for a particular pixel's depth, many positions along the top of a Gaussian curve have a fairly high probability, but only a few points at the Laplacian's peak will have a high probability.



(a) Laplacian Probability Density Function

(b) Gaussian Probability Density Function

Figure 11: Comparison of probability density curves after various manipulations.

The equation for a Gaussian curve is shown in equation 18.

$$f(x|\mu,\sigma^2) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right) \tag{18}$$

The equation for a Laplacian curve is shown in equation 19

$$f(x|\mu,b) = \frac{1}{2b} \exp\left(-\frac{|x-\mu|}{b}\right) \tag{19}$$

The Laplacian equation for modeling depth is shown in equation 20. It has the same basic form as equation 19 but computes one large summation from the product of many Laplacian curves. Equation 20 that was chosen for the depth estimation purposes here.

$$P(d|X;\theta,\lambda) = \frac{1}{Z} \exp\left(-\sum_{i=1}^{M} \frac{|d_i(1) - x_i^T\theta_r|}{\lambda_{1r}} - \sum_{s=1}^{3}\sum_{i=1}^{M}\sum_{j\in N_s(i)} \frac{|d_i(s) - d_j(s)|}{\lambda_{2rs}}\right) \tag{20}$$

Figure 12 shows the published results of the monocular depth map algorithm. The difference between Gaussian and Laplacian estimations are clearly visible in column three and four. The crisp point of the Laplacian curve results in sharper edges as well as more accurate, uniform objects. The Gaussian gives a fair approximation of the scene but is outshone by the Laplacian results.

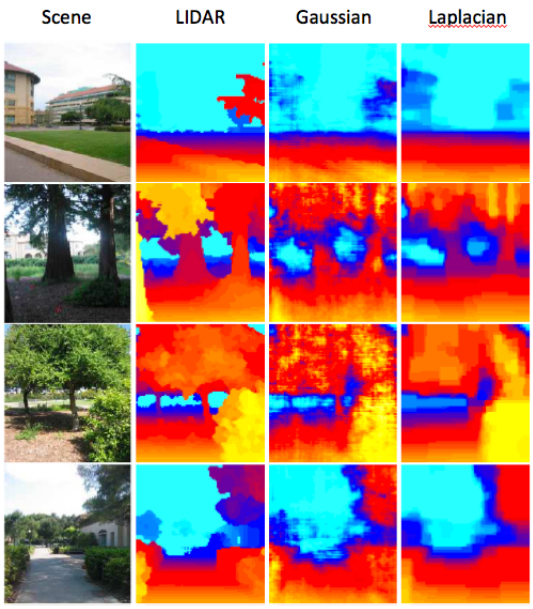
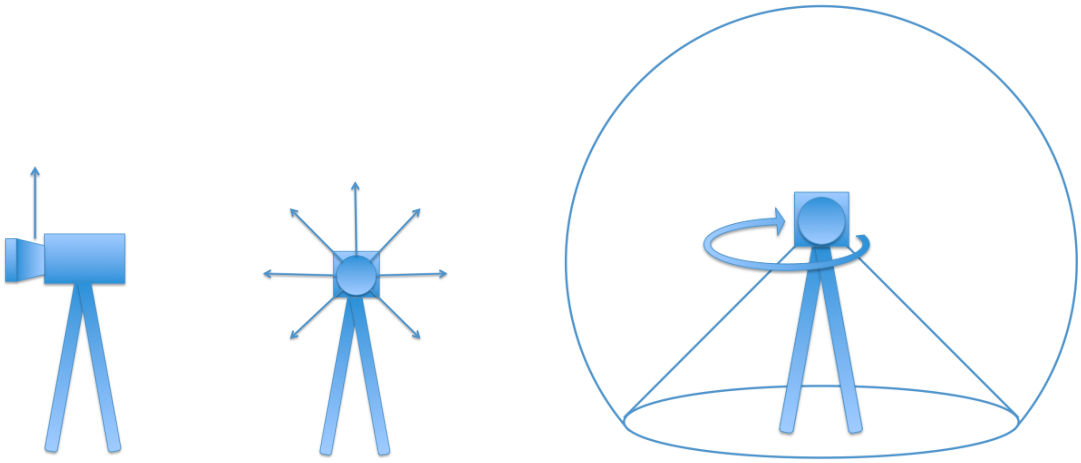


Figure 12: Depth map results from monocular images [5].

0.4.3 LIDAR

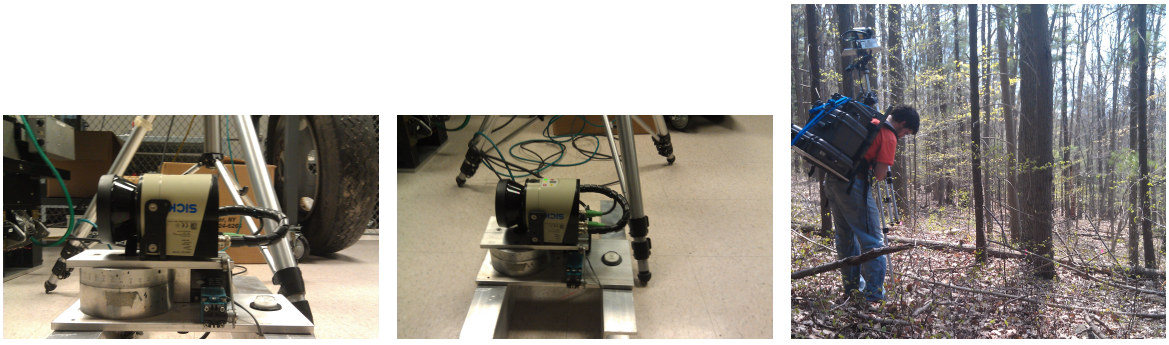
Light detection and ranging (LIDAR), is a laser scanning sensor designed to collect thousands of x, y, z positions during a single scan. LIDAR works by emitting a light pulse and times how long it takes for the pulse to return. Once the laser pulse hits an object it is reflected back towards the scanner. The speed of light along with the time interval can be used to determine the position of the object that the laser reflected from. The Digital Imaging and Remote Sensing(DIRS) laboratory at Rochester Institute of Technology owns a SICK-LMS 151 ground-based LIDAR system. The system has a maximum field of view of 270°, an angular step width resolution of 0.25°, and a scanning range of 50 meters. It has the capability of measuring two pulse returns, but only distance to first return was pertinent for this application. Figure 13 depicts the way the laser beam is emitted from the SICK LMS-151 system. The scanner emits a single laser beam, shown in Figure 0.4.3, that is reflected internally by a mirror to trace a 270° arc, shown in Figure 0.4.3. The base of the scanner can rotate 180°, allowing the laser beam to send out pulses in almost a full sphere, as seen in Figure 0.4.3. The system is restricted to a 270° field of view by its own base and physical presence.



(a) Side View (b) Front View (c) Full field of view

Figure 13: Diagram of the SICK LMS-151 system's field of view.

The DIRS group outfitted the scanner for mobile field work. They included a level on the rotating base for easy use outdoors in variable terrain. The rotating base is controlled by an iPod touch application written by the group. The app allows the user to select the desired field of view and begin the scanner's laser emission. The SICK scanner is connected via ethernet to a self-contained computer accessible by its own wifi hotspot. The computer, rotating base, and scanner are powered by a portable rechargeable power source, which in the case of the DIRS group is a car battery. Figure 14 shows pictures taken by the DIRS group of the scanner set up in the lab, and in the field.



(a) System head (b) System head with cabling (c) System in the field

Figure 14: Operational pictures of the SICK LMS-151 system.

0.5 Results

The stereoscopic algorithm, and its testing, was limited by processing speed. The implementation of the algorithm took about eight minutes to process a single stereoscopic pair of images with a resolution of 350 x 600 pixels and produce a depth map. Ideally, an optimized version of the

algorithm could operate fast enough to process 24 fps video in a reasonable amount of time. An ideal implementation would also allow the user to manipulate the variables a, b, and $P_{boundary}$ used in the cost terms, and view the different depth map results. Because of the slow speed of this implementation, a, b, and $P_{boundary}$ were each given a fixed value, determined from an average of what appeared to produce the best overall results. The slow processing speed of the algorithm also limited the resolution of the images used, because high resolution images took an unreasonable amount of time to process. The accuracy of the algorithm decreases with image resolution. At higher resolution the normalized-cross correlation step preforms better because high frequency information is preserved. Higher resolution also allows for a greater range of disparity and thus more accuracy in finding depth.

The algorithm fell short of expectations in some areas and exceeded them in others. Generally, it was expected that high frequency luminance content would perform well and low frequency content would not. However, very high frequency content that did not accompany changes in depth caused issues in the disparity correction step. When disparity edges were compared with nearby luminance edges that were not an object edge (in other words, a large change in luminance on the surface of an object), the disparity edges often were corrected to the wrong location. This is very evident in both the construction image, Figure 16 column 2, and the image of the person posing who has writing on his clothing, Figure 15 column 1. Conversely, low frequency content often performed surprisingly well, as shown in the green screen images. Although the algorithm sometimes produced small depth changes over the course of what was actually a flat surface, the depth measured by the algorithm was quite close to the actual depth and the inaccuracies were not large enough to cause keying problems.

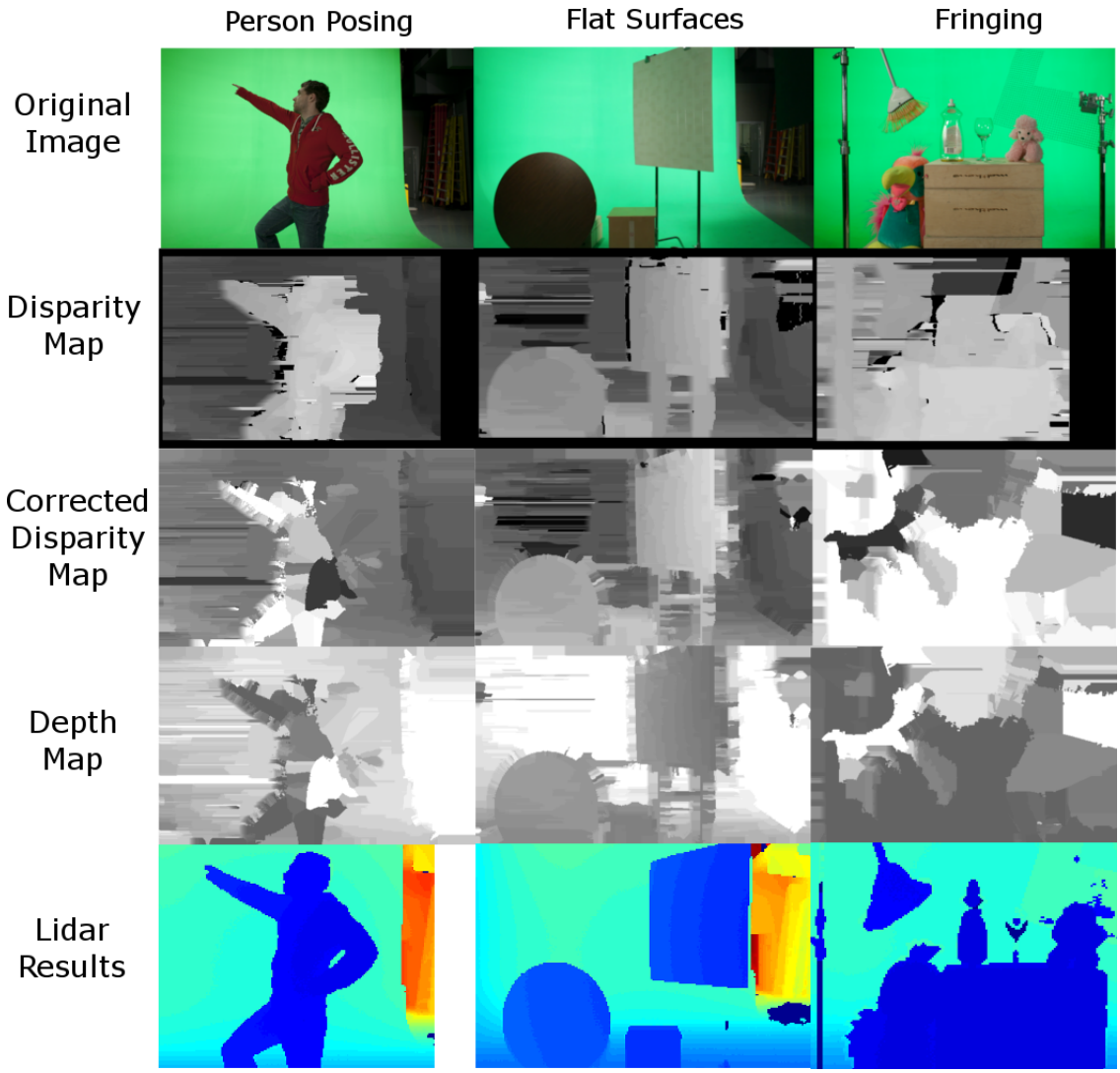


Figure 15: Green screen scene results from stereoscopic algorithm as compared to LIDAR. For all three depth maps, absolute white represents a distance of 6 meters.

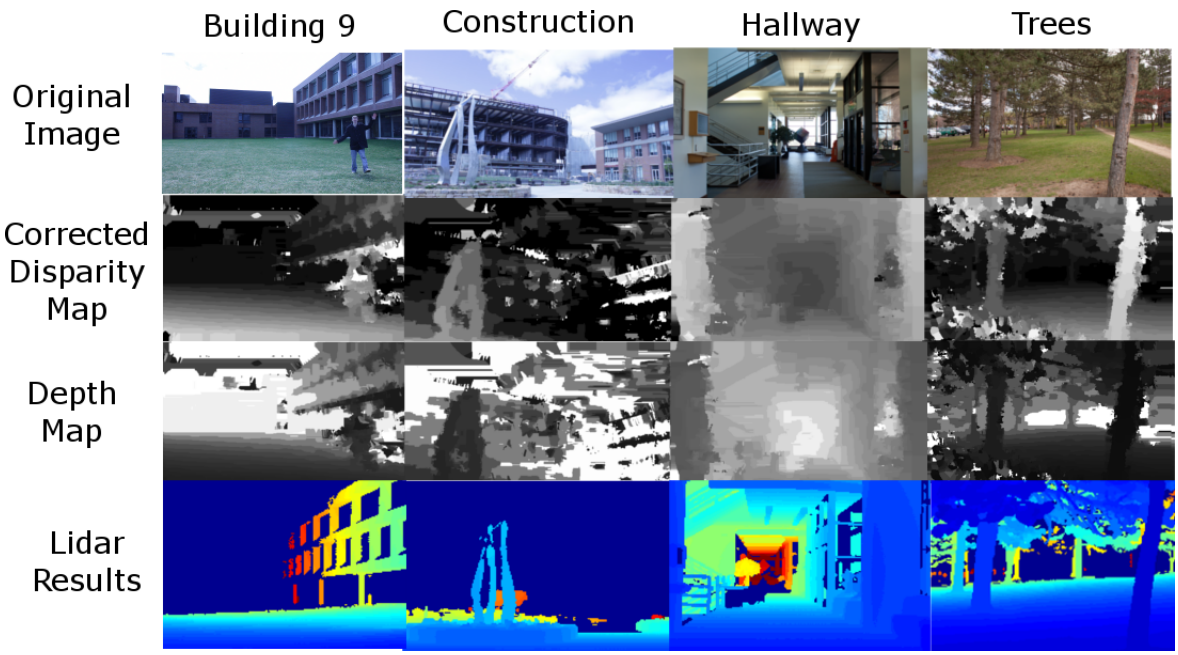


Figure 16: Natural scene results from stereoscopic algorithm as compared to LIDAR. In the depth maps, building 9 has a maximum depth of 45 meters, the construction has a maximum depth of 85 meters, the hallway has a maximum depth of 20 meters and the trees have a maximum depth of 85 meters. An object located beyond 85 meters has a 0 pixel disparity and is represented by a depth of 85 meters.

When the depth images were tested by a compositor, he found the correction process often left some inaccurate bands around object edges. However, the bands often had a large enough difference from the object that they could be keyed out along with the background. The edges were very sharp and matched the objects well in many cases, more so than would be possible with a chroma key. Also, the green light spill onto the objects in the green screen scenes meant that, to chroma key properly, the color of the image had to be manipulated, which produced a noticeable amount of grain and reduced the saturation of the image. The depth keyed versions did not have this problem. The depth images, however, had a lot of noise, which in some cases was easy to matte out but in others caused significant issues.

Figure 17: Depth keyed outdoor image.

Figure 18: Depth keyed green screen image.

Figure 19: Chroma keyed green screen image.

In the keyed images above, the chroma key is obviously more complete. However, the depth keyed images best edges, around the top left side of the circle and left side of the white triangle in Figure 18, are sharper and more accurate than the chroma keyed image, Figure 19. Also, the chroma keyed image has a noticeable color shift, while the depth keyed image maintains the images original color. In this comparison, the chroma key was completed using the professional application Keylight. The depth keying has a good deal of room for improvement, as noted earlier.

Below are the results of the LIDAR scans compared to the results of the stereoscopic depth algorithm. The LIDAR images were rendered in false color to better illustrate the depth changes. In most cases the distance estimated by the stereoscopic method are very close to the ground truth. In a few cases such as Figure 21, the stereoscopic method was more accurate than the LIDAR. The LIDAR acted as ground truth for these comparisons but in a few notable paces, like Figure 21, it is very obviously incorrect. When the scanner sends out a laser pulse into the sky of an outdoor scene the pulse is never reflected back to the scanner and causes the system to register the point as zero depth instead of infinity. This phenomenon can be seen in Figure 23, 24, and 25. The LIDAR's processing system includes an internal correction for dust which can cause other anomalies. In Figure 21, the chicken wire included in the scene was recorded as being *behind* the green screen wall, presumably because the small grid resembled dust. The laser used in the SICK system is 905nm and interacts with objects differently than visible spectrum light. In Figure 21, parts of the wine glass are not recorded at all by the scanner because the glass was so thin that the 905nm light did not interact with it, and merely passed through to the green screen.

Table 1: Green Screen: Table, nightstand, wall.

Point	LIDAR Dist[m]	3D Dist[m]	Discription
1	2.786	2.802	Nearest Wall
2	3.102	3.113	Middle Wall
3	3.541	3.502	Far Wall
4	7.688	6.466	Green Screen

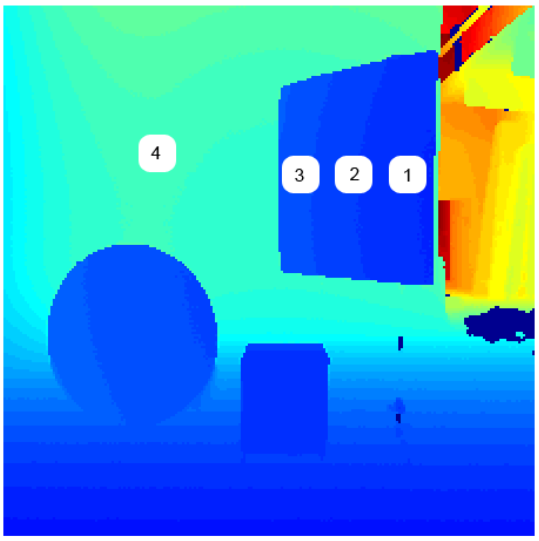


Figure 20: LIDAR results of table scene.

Table 2: Green Screen: Fringing objects.

Point	LIDAR Dist[m]	3D Dist[m]	Discription
1	6.27	3.113	Chicken Wire
2	5.853	5.254	Green Screen
3	0	2.335	Wine Glass
4	1.597	2.335	Broom Bristles

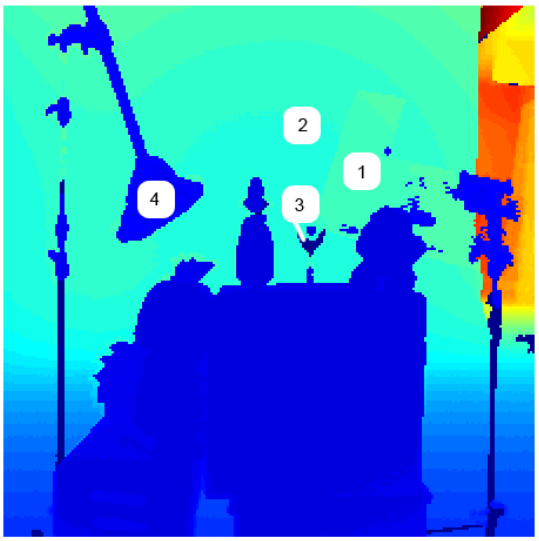


Figure 21: LIDAR results of fringing scene.

Table 3: Green Screen: Human posing.

Point	LIDAR Dist[m]	3D Dist[m]	Discription
1	1.628	2.335	Left Leg
2	1.839	2.402	Right Leg
3	2.129	2.335	Right Hand

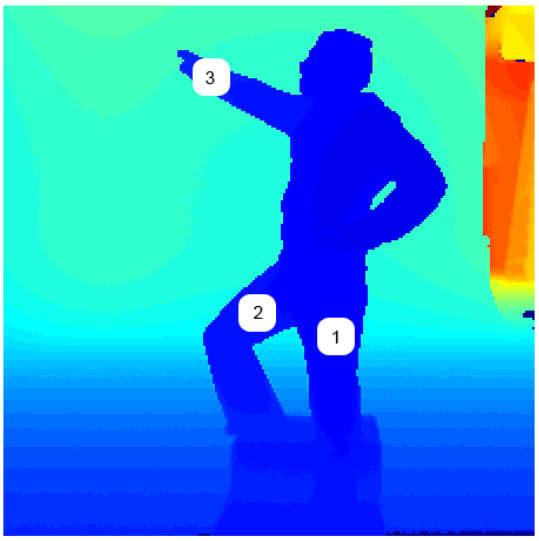


Figure 22: LIDAR results of human figure.

Table 4: Outdoor: Pine Grove.

Point	LIDAR Dist[m]	3D Dist[m]	Discription
1	3.432	4.67	1st Tree
2	7.918	9.34	2nd Tree
3	15.025	16.81	3rd Tree
4	22.959	28.02	4th Tree

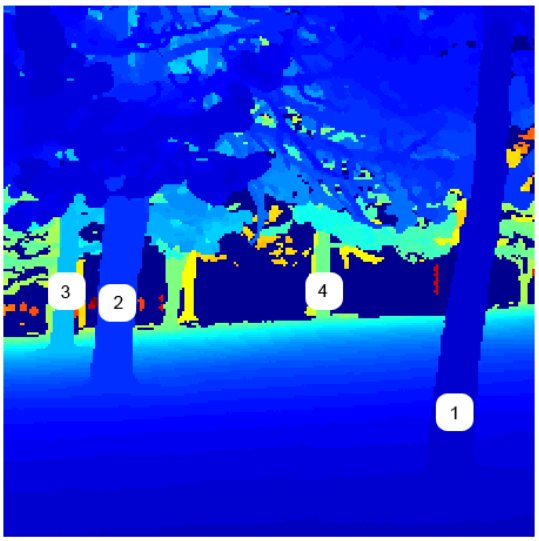


Figure 23: LIDAR results of pine grove scene.

Table 5: Outdoor: Building 9.

Point	LIDAR Dist[m]	3D Dist[m]
1	18.708	21.02
2	24.64	28.02
3	28.088	28.02
4	38.758	42.03

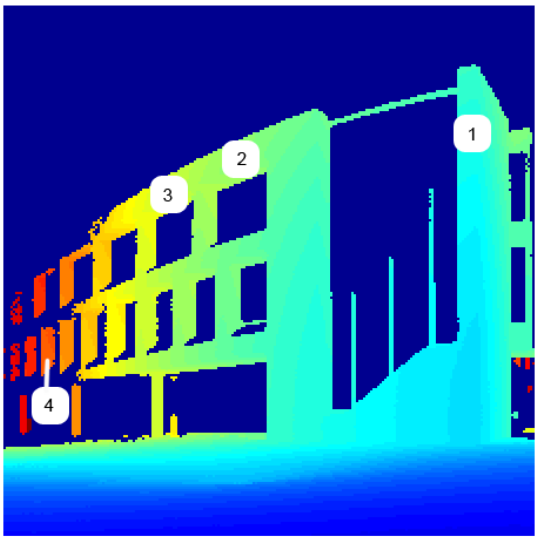


Figure 24: LIDAR results of building exterior.

Table 6: Outdoor: Sculpture and construction behind building 76.

Point	LIDAR Dist[m]	3D Dist[m]	Discription
1	14.059	14.01	Sculpture
2	10.042	10.51	Base
3	18.475	28.02	Further Base
4	0		

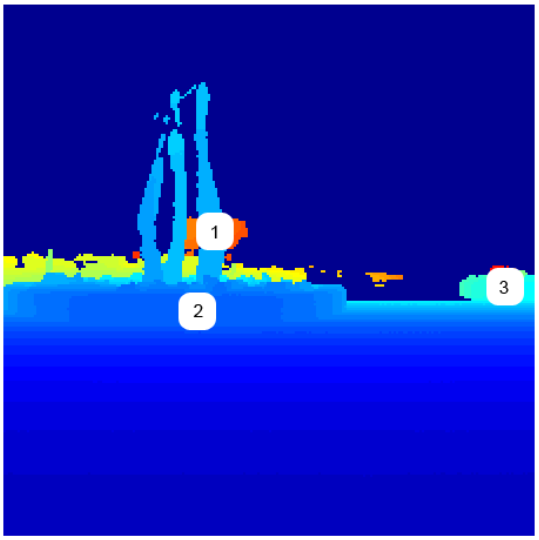


Figure 25: LIDAR results of construction and sculpture scene.

Table 7: Indoor: Front hallway of building 76.

Point	LIDAR Dist[m]	3D Dist[m]	Discription
1	5.425	7.33	Wall
2	12.013	12.932	Stairs
3	15.447	14.01	Stairwell
4	27.881	18.68	Color Cube

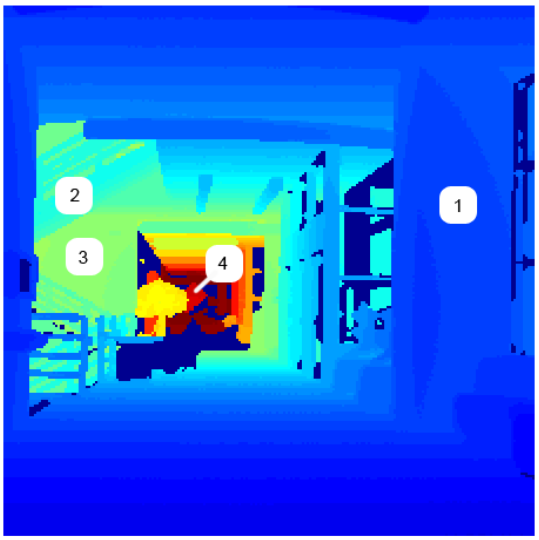


Figure 26: LIDAR results of interior hallway scene.

0.6 Conclusions

While this implementation for monocular capture was unsuccessful, other implementations that were successful have results that would likely not be satisfactory for keying. Edges are typically either very blurred or have inaccurate location, depending on the method used, and would not be very useful for compositing. The results of the LIDAR showed that an auxiliary device could be very useful for gaining depth information on a static scene, but the time required to scan the environment would not be able produce a real-time depth channel for video. The stereoscopic algorithm implemented here showed promise, producing crisp, accurate edges for some image content. While the necessity for a stereoscopic camera rig could be a deterrent, the trend towards recording films in 3D is gaining momentum and many films may be shot with stereoscopic rigs in the near future. Additionally, the setup of a second camera may be a simpler and more attractive option than the setup of a green screen. A more robust and optimized implementation of the algorithm could likely give satisfactory results in a timely fashion and rival or surpass chroma keying, making it a realistic option for depth keying video.

0.7 Acknowledgments

We would like to thank Ricardo Figueroa, David Long, Dr.Jan van Aradt, Dr.Harvey Rhody, Kerry Cawse-Nicholson, Paul Romanczyk, David Kelbe, and Ryan Bliss. Their help this year was greatly appreciated.

Bibliography

[1] L. Falkenhagen, "Depth estimation from stereoscopic image pairs assuming piecewise continuous surfaces," Tech. Rep., Institut für Theoretische Nachrichtentechnik und Informationsverarbeitung Universität Hannover, Hannover, Germany, n.d.

[2] I. J. Cox et al., "Stereo without disparity gradient smoothing: a bayesian sensor fusion solution," in *British Machine Vision Conference*, 1992, pp. 337–346.

[3] Y. Yakimovsky, United States. National Aeronautics, Space Administration, and Jet Propulsion Laboratory (U.S.), *A System for Extracting 3-dimensional Measurements from a Stereo Pair of TV Cameras*, NASA technical memorandum. Jet Propulsion Laboratory, California Institute of Technology, 1976.

[4] D. Sunday. (2006), *Distance between Lines and Segments with their Closest Point of Approach*, [Online]. Available: http://www.softsurfer.com.

[5] A. Saxena et al., "Learning depth from single monocular images," Tech. Rep., Computer Science Department, Stanford University, Stanford, CA.

[6] E.R. Davies, *Machine Vision: Theory, Algorithms, Practicalities*, 3rd ed., Oxford, UK: Elsevier, 2005.

[7] P. Fua, "A parallel stereo algorithm that produces dense depth maps and preserves image features," *Machine Vision and Applications*, vol. 6, no. 1, pp. 35–49, 1993.