High-Accuracy Stereo Depth Maps Using Structured Light

Daniel Scharstein
Middlebury College
schar@middlebury.edu

Richard Szeliski
Microsoft Research
szeliski@microsoft.com

Abstract

Recent progress in stereo algorithm performance is quickly outpacing the ability of existing stereo data sets to discriminate among the best-performing algorithms, motivating the need for more challenging scenes with accurate ground truth information. This paper describes a method for acquiring high-complexity stereo image pairs with pixel-accurate correspondence information using structured light. Unlike traditional range-sensing approaches, our method does not require the calibration of the light sources and yields registered disparity maps between all pairs of cameras and illumination projectors. We present new stereo data sets acquired with our method and demonstrate their suitability for stereo algorithm evaluation. Our results are available at http://www.middlebury.edu/stereo/.

1. Introduction

The last few years have seen a resurgence of interest in the development of highly accurate stereo correspondence algorithms. Part of this interest has been spurred by fundamental breakthroughs in matching strategies and optimization algorithms, and part of the interest is due to the existence of image databases that can be used to test and compare such algorithms. Unfortunately, as algorithms have improved, the difficulty of the existing test images has not kept pace. The best-performing algorithms can now correctly match most of the pixels in data sets for which correct (ground truth) disparity information is available [21].

In this paper, we devise a method to automatically acquire high-complexity stereo image pairs with pixel-accurate correspondence information. Previous approaches have either relied on hand-labeling a small number of images consisting mostly of fronto-parallel planes [17], or setting up scenes with a small number of slanted planes that can be segmented and then matched reliably with parametric correspondence algorithms [21]. Synthetic images have also been suggested for testing stereo algorithm performance [12, 9], but they typically are either too easy to solve if noise, aliasing, etc. are not modeled, or too difficult, e.g., due to complete lack of texture in parts of the scene.

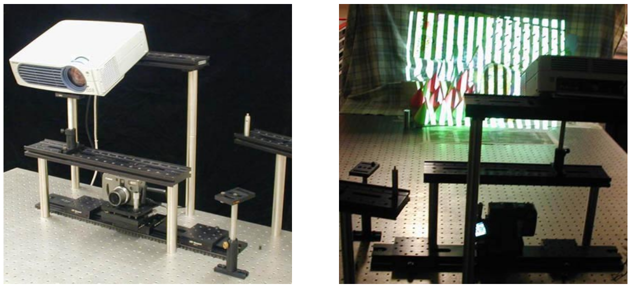


Figure 1. *Experimental setup, showing the digital camera mounted on a translation stage, the video projector, and the complex scene being acquired.*

In this paper, we use structured light to uniquely label each pixel in a set of acquired images, so that correspondence becomes (mostly) trivial, and dense pixel-accurate correspondences can be automatically produced to act as ground-truth data. Structured-light techniques rely on projecting one or more special light patterns onto a scene, usually in order to directly acquire a range map of the scene, typically using a single camera and a single projector [1, 2, 3, 4, 5, 7, 11, 13, 18, 19, 20, 22, 23]. Random light patterns have sometimes been used to provide artificial texture to stereo-based range sensing systems [14]. Another approach is to register range data with stereo image pairs, but the range data is usually of lower resolution than the images, and the fields of view may not correspond exactly, leading to areas of the image for which no range data is available [16].

2. Overview of our approach

The goal of our technique is to produce pairs of real-world images of complex scenes where each pixel is labeled with its *correspondence* in the other image. These image pairs can then be used to test the accuracy of stereo algorithms relative to the known ground-truth correspondences.

Our approach relies on using a pair of cameras and one or more light projectors that cast structured light patterns onto the scene. Each camera uses the structured light sequence to determine a unique *code* (label) for each pixel. Finding inter-image correspondence then trivially consists of finding the pixel in the corresponding image that has the same unique code.

The advantage of our approach, as compared to using a separate range sensor, is that the data sets are automatically registered. Furthermore, as long as each pixel is illuminated by at least one of the projectors, its correspondence in the other image (or lack of correspondence, which indicates occlusion) can be unambiguously determined.

In our current experimental setup (Figure 1), we use a single digital camera (Canon G1) translating on a linear stage, and one or two light projectors illuminating the scene from different directions. We acquire images under both structured lighting and ambient illumination conditions. The ambient illuminated images can be used as inputs to the stereo matching algorithms being evaluated.

Let us now define some terminology. We distinguish between *views* – the images taken by the cameras – and *illuminations* – the structured light patterns projected onto the scene. We model both processes using planar perspective projection and use coordinates (x, y) for views and (u, v) for illuminations.

There are two primary camera views, L (left) and R (right), between which correspondences are to be established. The illumination sources from which light patterns are projected are identified using numbers $\{0, 1, \ldots\}$. More than one illumination direction may be necessary to illuminate all surfaces in complex scenes with occluding objects.

Our processing pipeline consists of the following stages:

1. Acquire all desired views under all illuminations.
2. Rectify the images to obtain the usual horizontal epipolar geometry, using either a small number of corresponding features [15] or dense 2D correspondences (step 4).
3. Decode the light patterns to get (u, v) codes at each pixel in each view.
4. Use the unique codes at each pixel to compute correspondences. (If the images are rectified, 1D search can be performed, else 2D search is required.) The results of this correspondence process are the (usual) *view disparities* (horizontal motion).
5. Determine the projection matrices for the illumination sources from the view disparities and the code labels.
6. Reproject the code labels into the two-view geometry. This results in the *illumination disparities*.
7. Combine the disparities from all different sources to get a reliable and accurate final disparity map.
8. Optionally crop and downsample the disparity maps and the views taken under ambient lighting.

The remainder of this paper is structured as follows. The next section describes the algorithms used to determine unique codes from the structured lighting. Section 4 discusses how view disparities and illumination disparities are established and merged. Section 5 describes our experimental results, and Section 6 describes our conclusions and future work.

3. Structured light

To uniquely label each pixel, we project a series of structured light images onto the scene, and decode the set of projected intensities at each pixel to give it a unique label. The simplest kind of pattern to project is a series of single stripe images (light planes) [3, 7, 19], but these require $O(n)$ images, where n is the width of the image in pixels.

Instead, we have tested two other kinds of structured light: binary Gray-code patterns, and series of sine waves.

3.1. Gray codes

Gray-code patterns only contain black and white (on/off) pixel values, which were the only possibilities available with the earliest LCD projectors. Using such binary images requires $\log_2(n)$ patterns to distinguish among n locations. For our projector (Sony VPL-CX10) with 1024×768 pixels, it is sufficient to illuminate the scene with 10 vertical and 10 horizontal patterns, which together uniquely encode the (u, v) position at each pixel. Gray codes are well suited for such binary position encoding, since only one bit changes at a time, and thus small mislocalizations of 0-1 changes cannot result in large code changes [20].

Decoding the light patterns is conceptually simple, since at each pixel we need only decide whether it is illuminated or not. We could for example take two reference images, all-white and all-black, and compare each code pixel with the average of the two. (With a gray-level projector, we could also project a reference image with 0.5 intensity). Such reference images measure the *albedo* of each scene point. In practice, however, this does not work well due to interreflections in the scene and “fogging” inside the projector (adding a low-frequency average of intensities to the projected pattern), which causes increased brightness near bright areas. We have found that the only reliable way of thresholding pixels into on/off is to project both the code pattern *and its inverse*. We can then label each pixel according to whether the pattern or its inverse appears brighter. This avoids having to estimate the local albedo altogether. The obvious drawback is that twice as many images are required. Figure 2 shows examples of thresholded Gray-code images.

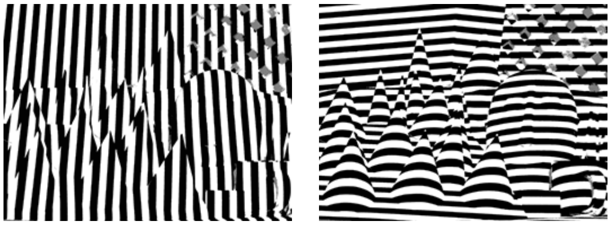


Figure 2. *Examples of thresholded Gray-code images. Uncertain bits are shown in gray. (Full-size versions of all images in this paper are available at http://www.middlebury.edu/stereo/.)*

Unfortunately, even using patterns and their inverses may not be enough to reliably distinguish light patterns on surfaces with widely varying albedos. In our experiments, we have found it necessary to use two different exposure times (0.5 and 0.1 sec.). At each pixel, we select the exposure setting that yields the largest absolute difference between the two illuminations. If this largest difference is still below a threshold (sum of signed differences over all color bands < 32), the pixel is labeled “unknown” (gray pixels in Figure 2), since its code cannot be reliably determined. This can happen in shadowed areas or for surfaces with very low albedo, high reflectance, or at oblique angles.

The initial code values we obtain by concatenating the bits from all the thresholded binary images need to be cleaned up and potentially interpolated, since the camera resolution is typically higher than projector resolution. In our case, the projector has a 1024×768 resolution, and the camera has 2048×1536. Since the camera only sees a subset of the illuminated scene (i.e., it is zoomed in) and illumination pixels can appear larger on slanted surfaces, we get even more discrepancy in resolution. In our setup, each illumination pixel is typically 2–4 camera pixels wide. We clean up the Gray code images by filling small holes caused by unknown bit values. We then interpolate (integer) code values to get a higher resolution and avoid multiple pixels with the same code. Interpolation is done in the prominent code direction, i.e., horizontally for u and vertically for v. We currently compute a robust average over a sliding 1D window of 7 values. The results of the entire decoding process are shown in Figure 4a.

3.2. Sine waves

Binary Gray-code patterns use only two different intensity levels and require a whole series of images to uniquely determine the pixel code. Projecting a continuous function onto the scene takes advantage of the gray-level resolution available in modern LCD projectors, and can thus potentially require fewer images (or alternatively, result in greater precision for the same number of images). It can also potentially overcome discretization problems that might introduce artifacts at the boundaries of binary patterns [6].

Consider for example projecting a pure white pattern and a gray-level ramp onto the scene. In the absence of noise and non-linearities, the ratio of the two values would give us the position along the ramp of each pixel. However, this approach has limited effective spatial resolution [11, 22]. Projecting a more quickly varying pattern such as a sawtooth alleviates this, but introduces a phase ambiguity (points at the same phase in the periodic pattern cannot be distinguished), which can be resolved using a series of periodic patterns at different frequencies [13]. A sine wave pattern avoids the discontinuities of a sawtooth, but introduces a further two-way ambiguity in phase, so it is useful to project two or more waves at different phases

Our current algorithm projects sine waves at two different frequencies and 12 different phases. The first frequency has a period equal to the whole (projector) image width or height; the second has 10 periods per screen.

Given the images of the scene illuminated with these patterns, how do we compute the phase and hence (u, v) coordinates at each pixel? Assuming a linear image formation process, we have the following (color) image formation equation

$$\vec{I}_{kl}(x, y) = \vec{A}(x, y) B_{kl}[\sin(2\pi f_k u + \phi_l) + 1], \quad (1)$$

where $\vec{A}(x, y)$ is the (color) albedo corresponding to scene pixel (x, y), B_{kl} is the intensity of the (k, l)th projected pattern, f_k is its frequency, and ϕ_l is its phase. A similar equation can be obtained for horizontal sine wave patterns by replacing u with v.

Assume for now that we only have a single frequency f_k and let $c_l = \cos\phi_l$, $s_l = \sin\phi_l$, $c_u = \cos(2\pi f_k u)$, $s_u = \sin(2\pi f_k u)$, and $\vec{C} = \vec{A}(x, y)B$. The above equation can then be re-written (for a given pixel (x, y)) as

$$\vec{I}_{kl} = \vec{C}[s_u c_l + c_u s_l + 1]. \quad (2)$$

We can estimate the illuminated albedo value $\vec{C}$ at each pixel by projecting a mid-tone grey image onto the scene. The above equation is therefore linear in the unknowns (c_u, s_u), which can be (optimally) recovered using linear least squares [10], given a set of images with different (known) phases ϕ_l. (In the least squares fitting, we ignore any color values that are saturated, i.e., greater than 240.) An estimate of the u signal can then be recovered using

$$u = p_u^{-1}(\frac{1}{2\pi} \tan^{-1} \frac{s_u}{c_u} + m), \quad (3)$$

where $p_u = W/f_u$ is the sine period (in pixels) and m is the (unknown) integral phase wrap count. To solve the phase wrapping problem, we first estimate the value of u using

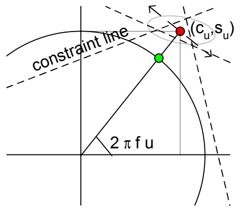


Figure 3. *Phase estimation from (c_u, s_u) least squares fit. The red dot is the least squares solution to the constraint lines, and the ellipse around it indicates the two-dimensional uncertainty.*

a single wave ($f_1 = 1$), and then repeat the estimation with $f_2 = 10$, using the previous result to disambiguate the phase.

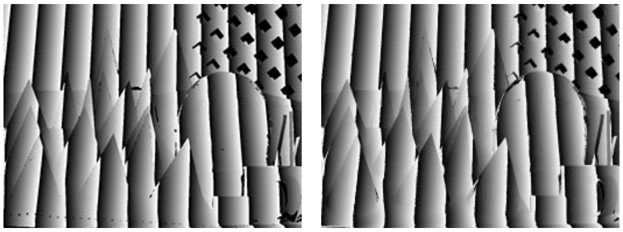
Since we are using least squares, we can compute a *certainty* for the u estimate. The normal equations for the least squares system directly give us the *information matrix* (inverse covariance) for the (c_u, s_u) estimate. We can convert this to a variance in u by projecting along the direction normal to the line going through the origin and (c_u, s_u) (Figure 3). Furthermore, we can use the distance of the fitted point (c_u, s_u) from the unit circle as a sanity check on the quality of our sine wave fit. Computing certainties allows us to merge estimates from different exposures. At present, we simply pick the estimate with the higher certainty.

Figure 4b shows the results of recovering the u positions using sine patterns. For these experiments, we use all 12 phases ($\phi = 0°, 30°, \ldots, 330°$) and two different exposures (0.1 and 0.5 sec). In the future, we plan to study how the certainty and reliability of these estimates varies as a function of the number of phases used.

3.3. Comparison

Figure 4 shows examples of u coordinates recovered both from Gray code and sine wave patterns. The total number of light patterns used is 80 for the Gray codes (10 bit patterns and their inverses, both u and v, two exposures), and 100 for the sine waves (2 frequencies and 12 phases plus 1 reference image, both u and v, two exposures). Visual inspection shows that the Gray codes yield better (less noisy) results. The main reason is that by projecting binary patterns and their inverses, we avoid the difficult task of estimating the albedo of the scene. Although recovering the phase of sine wave patterns potentially yields higher resolution and could be done with fewer images, it is also more susceptible to non-linearities of the camera and projector and to interreflections in the scene.

In practice, the time to take the images of all structured light patterns is relatively small compared to that of setting up the scene and calibrating the cameras. We thus use the Gray code method for the results reported here.



(a): Gray code (b): sine wave

Figure 4. *Computed u coordinates (only low-order bits are shown).*

4. Disparity computation

Given N illumination sources, the decoding stage described above yields a set of labels $(u_{ij}(x, y), v_{ij}(x, y))$, for each illumination $i \in \{0, \ldots, N-1\}$ and view $j \in \{L, R\}$. Note that these labels not only uniquely identify each scene point, but also encode the coordinates of the illumination source. We now describe how high-accuracy disparities can be computed from such labels corresponding to one or more illumination directions.

4.1. View disparities

The first step is to establish correspondences between the two views L and R by finding matching code values. Assuming rectified views for the moment, this amounts to a simple 1D search on corresponding scanlines. While conceptually simple, several practical issues arise:

- Some pixels may be partially occluded (visible only in one view).
- Some pixels may have unknown code values in some illuminations due to shadows or reflections.
- A perfect matching code value may not exist due to aliasing or interpolation errors.
- Several perfect matching code values may exist due to the limited resolution of the illumination source.
- The correspondences computed from different illuminations may be inconsistent.

The first problem, partial occlusion, is unavoidable and will result in unmatched pixels. The number of unknown code values due to shadows in the scene can be reduced by using more than one illumination source, which allows us to establish correspondences at all points illuminated by *at least* one source, and also enables a consistency check at pixels illuminated by more than one source. This is advantageous since at this stage our goal is to establish only high-confidence correspondences. We thus omit all pixels whose disparity estimates under different illuminations disagree. As a final consistency check, we establish disparities d_{LR} and d_{RL} independently and cross-check for consistency. We now have high-confidence view disparities at points visible in both cameras and illuminated by at least one source (see Figures 6b and 7b).

Before moving on, let us consider the case of unrectified views. The above method can still be used, except that a 2D search must be used to find corresponding codes. The resulting set of high-quality 2D correspondences can then be used to rectify the original images [15].

4.2. Illumination disparities

The next step in our system is to compute another set of disparities: those between the cameras and the illumination sources. Since the code values correspond to the image coordinates of the illumination patterns, each camera-illumination pair can be considered an independent source of stereo disparities (where the role of one camera is played by the illumination source). This is of course the idea behind traditional structured lighting systems [3].

The difference in our case is that we can register these *illumination disparities* with our rectified *view disparities* d_{LR} without the need to explicitly calibrate the illumination sources (video projectors). Since our final goal is to express all disparities in the rectified two-view geometry, we can treat the view disparities as a 3D reconstruction of the scene (i.e., *projective depth*), and then solve for the projection matrix of each illumination source.

Let us focus on the relationship between the left view L and illumination source 0. Each pixel whose view disparity has been established can be considered a (homogeneous) 3D scene point $S = [x \; y \; d \; 1]^T$ with projective depth $d = d_{LR}(x, y)$. Since the pixel's code values (u_{0L}, v_{0L}) also represent its x and y coordinates in the illumination pattern, we can write these coordinates as homogenous 2D point $P = [u_{0L} \; v_{0L} \; 1]^T$. We then have

$$P \cong M_{0L}\, S,$$

where M_{0L} is the unknown 4×3 projection matrix of illumination source 0 with respect to the left camera. If we let m_1, m_2, m_3 denote the three rows of M_{0L}, this yields

$$\begin{aligned} u_{0L}\, m_3\, S &= m_1\, S, \quad \text{and} \\ v_{0L}\, m_3\, S &= m_2\, S. \end{aligned} \tag{4}$$

Since M is only defined up to a scale factor, we set $m_{34} = 1$. Thus we have two linear equations involving the 11 unknown entries of M for each pixel whose disparity and illumination code are known, giving us a heavily overdetermined linear system of equations, which we solve using least squares [10].

If the underlying disparities and illumination codes are correct, this is a fast and stable method for computing M_{0L}. In practice, however, a small number of pixels with large disparity errors can strongly affect the least-squares fit. We therefore use a robust fit with outlier detection by iterating the above process. After each iteration, only those pixels with low residual errors are selected as input to the next iteration. We found that after 4 iterations with successively lower error thresholds we can usually obtain a very good fit.

Given the projection matrix M_{0L}, we can now solve Equation (4) for d at each pixel, using again a least-squares fit to combine the two estimates. This gives us the *illumination disparities* $d_{0L}(x, y)$ (see Figures 6c and 7c). Note that these disparities are available for all points illuminated by source 0, even those that are not visible from the right camera. We thus have a new set of disparities, registered with the first set, which includes half-occluded points. The above process can be repeated for the other camera to yield disparities d_{0R}, as well as for all other illumination sources $i = 1 \ldots N-1$.

4.3. Combining the disparity estimates

Our remaining task is to combine the $2N + 2$ disparity maps. Note that all disparities are already registered, i.e., they describe the horizontal motion between views L and R. The first step is to create combined maps for each of L and R separately using a robust average at pixels with more than one disparity estimate. Whenever there is a majority of values within close range, we use the average of this subset of values; otherwise, the pixel is labeled unknown. In the second step, the left and right (combined) maps are checked for consistency. For unoccluded pixels, this means that

$$d_{LR}(x, y) = -d_{RL}(x + d_{LR}(x, y), y),$$

and vice versa. If the disparities differ slightly, they are adjusted so that the final set of disparity maps is fully consistent. Note that since we also have disparities in half-occluded regions, the above equation must be relaxed to reflect all legal visibility situations. This yields the final, consistent, and highly accurate pair of disparity maps relating the two views L and R (Figures 6d and 7d).

The two final steps are cropping and downsampling. Up to this point, we are still dealing with full-size (2048×1536) images. In our setup, disparities typically range from about 210 to about 450. We can bring the disparities closer to zero by cropping to the joint field of view, which in effect stabilizes an imaginary point just behind farthest surface in the scene. This yields a disparity range of 0–240, and an image width of 1840. Since most current stereo implementations work with much smaller image sizes and disparity ranges, we downsample the images and disparity maps to quarter size (460×384). The disparity maps are downsampled using a majority filter, while the ambient images are reduced

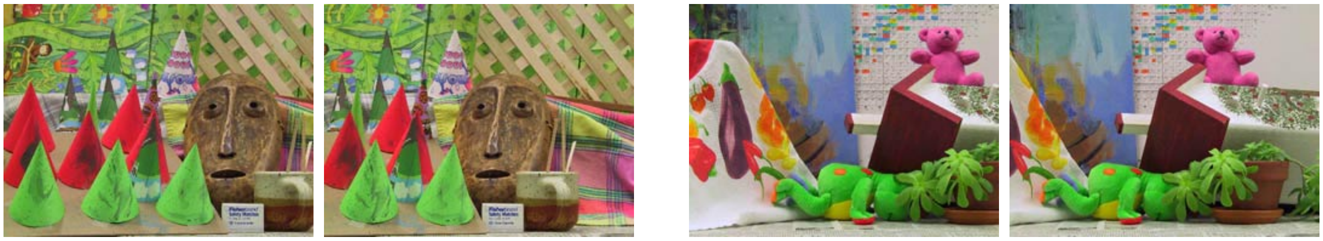


Figure 5. *The two image pairs:* Cones *(left) and* Teddy *(right).*

with a sharp 8-tap filter. Note that for the downsampled images, we now have disparities with quarter-pixel accuracy.

A remaining issue is that of holes, i.e., unknown disparity values, which are marked with a special value. While small holes can be filled by interpolation during the above process, large holes may remain in areas where no illumination codes were available to begin with. There are two main sources for this: (1) surfaces that are highly specular or have very low albedo; and (2) areas that are shadowed under all illuminations. Special care must be taken to avoid both situations when constructing test scenes whose disparities are to be estimated with the method described here.

5. Results

Using the method described in the previous sections, we have acquired two different scenes, *Cones* and *Teddy*. Figure 5 shows views L and R of each scene taken under ambient lighting. L and R are actually views 3 and 7 out of a total of 9 images we have taken from equally-spaced viewpoints, which can be used for prediction-error evaluation [21].

The *Cones* scene was constructed such that most scene points visible from either view L and R can be illuminated with a single light source from above (see Figure 6a). The exception is the wooden lattice in the upper right quadrant in the image. (Its shadow falls on a planar surface, however, so that the missing disparities could be filled in by interpolation.) For the *Teddy* we used two illumination directions (see Figure 7a). Due to the complex scene, however, several small areas are shadowed under both illuminations.

Figures 6 and 7 also show the recovered view disparities (b) and illumination disparities (c), as well as the final disparity maps combined from all sources (d). The combination step not only fills in half-occluded and half-shadowed regions, but also serves to detect outlier disparities (e.g., due to errors in the projection matrix estimation).

In order to verify that our stereo data sets are useful for evaluating stereo matching algorithms, we ran several of the algorithms from the Middlebury Stereo Page [21] on our new images. Figure 8 shows the results of three algorithms (SSD with a 21×21 shiftable window, dynamic programming, and graph cuts) on the cropped and downsampled image pairs, as well as the corresponding ground-truth data. Table 1 shows the quantitative performance in non-occluded areas (percentage of “bad” pixels with large disparity errors) for two error thresholds $t = 1$ and $t = 2$. (We ignore points whose true disparities are unknown.)

The results clearly indicate that our data sets are challenging, yet not unrealistically so. Difficulties posed to the matching algorithms include a large disparity range, complex surface shapes, textureless areas, narrow occluding objects and ordering-constraint violations. Of the algorithms tested, the graph-cut method performs best, although it clearly cannot handle some of the complex occlusion situations and some highly-slanted surfaces.

6. Conclusion

In this paper we have developed a new methodology to acquire highly precise and reliable ground truth disparity measurements accurately aligned with stereo image pairs. Such high-quality data is essential to evaluate the performance of stereo correspondence algorithms, which in turn spurs the development of even more accurate algorithms. Our new high-quality disparity maps and the original input images are available on our web site at http://www.middlebury.edu/stereo/. We plan to add these new data sets to those already in use to benchmark the performance of stereo correspondence algorithms [21].

Our novel approach is based on taking stereo image pairs illuminated with active lighting from one or more projectors. The structured lighting enables us to uniquely code each scene pixel, which makes inter-camera correspondence much easier and more reliable. Furthermore, the encoded positions enable the recovery of camera-projector disparities, which can be used as an auxiliary source of information to increase the reliability of correspondences and to fill in missing data.

We have investigated two different kinds of structured light: binary Gray codes, and continuous sine waves. At present, the Gray codes give us more reliable estimates of projector coordinates, due mainly to their higher insensitivity to effects such as photometric nonlinearities and interreflections. In future work, we plan to develop the sine-wave approach further, to see if we can reduce the total number of acquired images necessary to recover high-

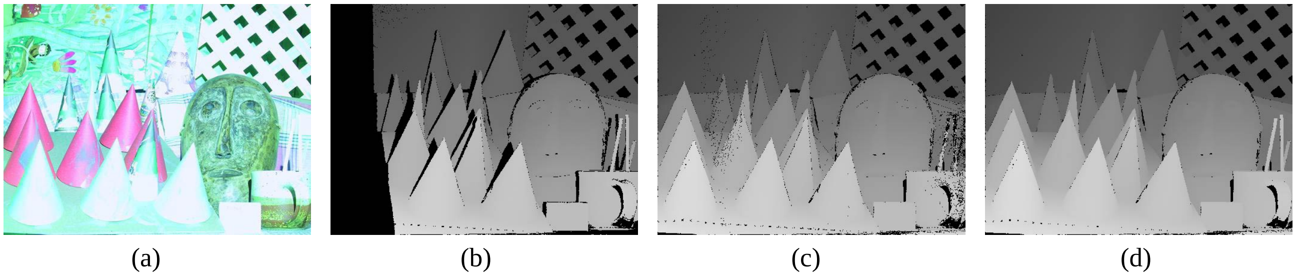


Figure 6. *Left view of* Cones *(one illumination source is used): (a) scene under illumination (note absence of shadows except in upper-right corner); (b) view disparities; (c) illumination disparities; (d) final (combined) disparity map. Unknown disparities are shown in black.*

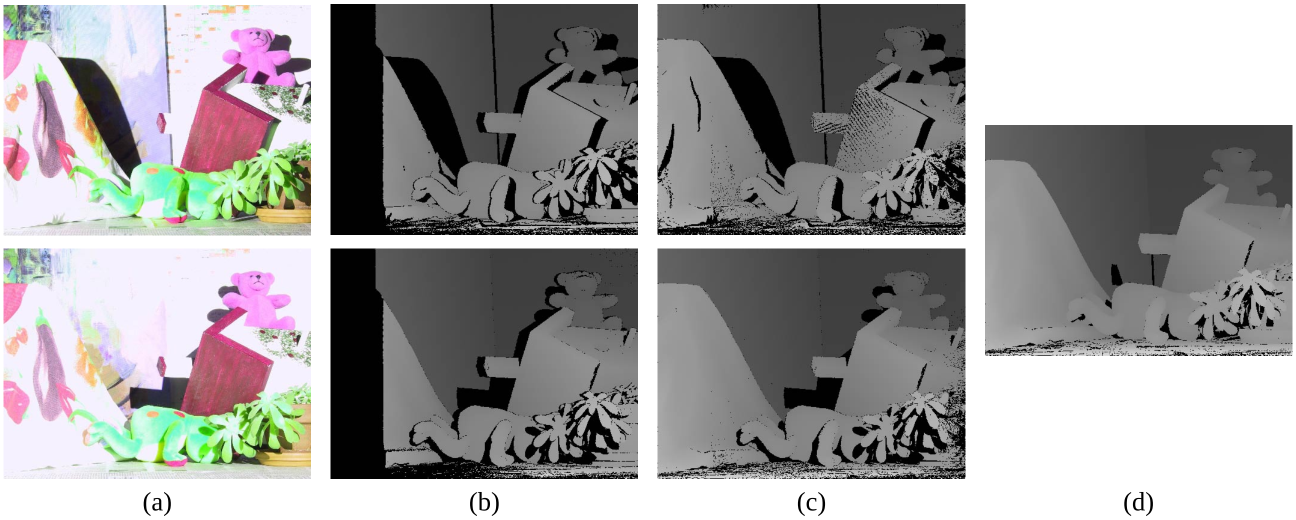


Figure 7. *Left view of* Teddy *(two illumination sources are used): (a) scene under the two illuminations; (b) view disparities; (c) illumination disparities; (d) final (combined) disparity map. Unknown disparities are shown in black.*

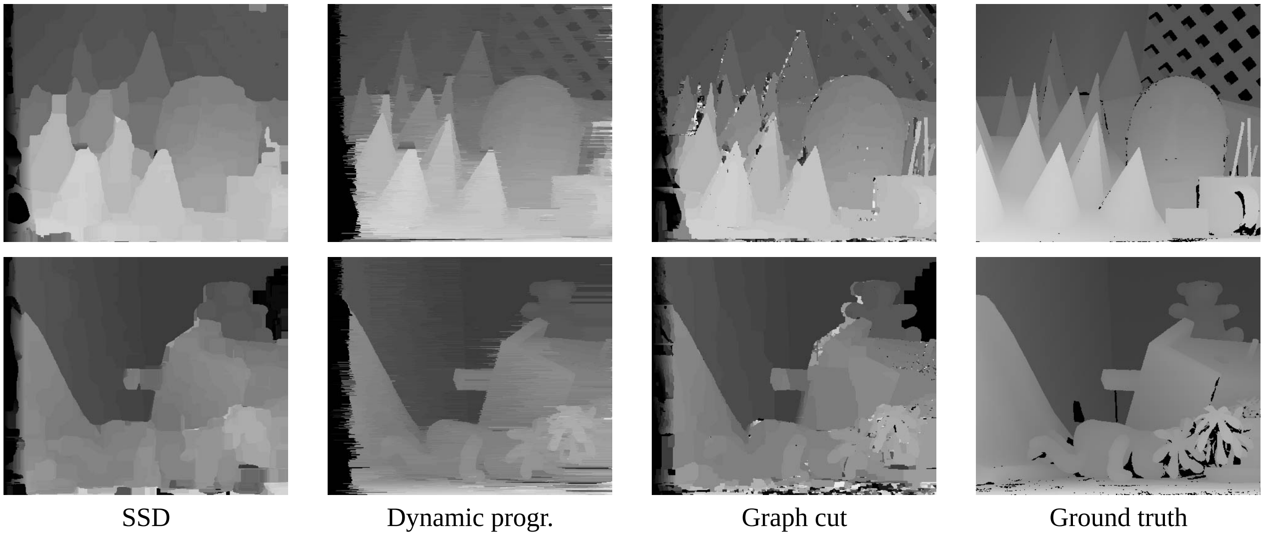


Figure 8. *Stereo results on cropped and downsampled images:* Cones *(top) and* Teddy *(bottom).*

Bad %	Cones		Teddy	
	$t=1$	$t=2$	$t=1$	$t=2$
SSD	17.8%	9.3%	26.5%	12.8%
DP	17.1%	9.8%	30.1%	10.5%
GC	12.6%	7.0%	29.3%	11.4%

Table 1. *Performance of SSD, dynamic programming, and graph cut stereo methods on our data sets. The table shows the percentage of pixels whose disparity error is greater than threshold t for $t = 1, 2$.*

quality pixel encodings. We also plan to fill in the remaining pixels currently marked as unknown in our disparity maps, using a combination of semi-automated and manual methods. It may also be possible to co-locate the cameras and projectors using mirrors and to use Zickler *et al.*'s beautiful results on reciprocity to deal with highlights [25].

While doing our research, we became aware of concurrent work aimed at acquiring high-quality correspondences with active illumination that is applicable to both static and dynamic scenes [8, 24]. Instead of decoding the projected light patterns to yield pixel addresses in the projector, these alternative methods simply temporally sum up the correlation or error measures of all frames to directly compute stereo disparities. This results in a simpler approach that produces high-quality inter-camera correspondences. Unlike our method, however, these techniques are not able to fill in semi-occluded areas using projected disparities.

We close the paper with the following challenge. Can one devise a comparable (or any) technique to acquire high-quality ground truth data for real-time two-dimensional motion? The existence of such data would be of invaluable use to the motion estimation community, just as we hope the data presented here will aid in developing better stereo matching algorithms.

Acknowledgments

Thanks to Alexander Vandenberg-Rodes who was instrumental in creating the scenes and capturing the images.

References

[1] G. J. Agin and T. O. Binford. Computer description of curved objects. *IEEE Trans. Comp.*, C-25(4):439–449, 1976.

[2] J. Batlle, E. Mouaddib, and J. Salvi. Recent progress in coded structured light as a technique to solve the correspondence problem: a survey. *Pat. Recog.*, 31(7):963–982, 1998.

[3] P. Besl. Active optical range imaging sensors. In Jorge L.C. Sanz, editor, *Advances in Machine Vision*, pp. 1–63, 1989.

[4] J.-Y. Bouguet and P. Perona. 3D photography on your desk. In *ICCV'98*, pp. 43–50, 1998.

[5] C. Chen, Y. Hung, C. Chiang, and J. Wu. Range data acquisition using color structured lighting and stereo vision. *Image and Vision Computing*, 15(6):445–456, 1997.

[6] Y.-Y. Chuang et al. Environment matting extensions: towards higher accuracy and real-time capture. In *SIGGRAPH 2000*, pp. 121–130, 2000.

[7] B. Curless and M. Levoy. Better optical triangulation through spacetime analysis. In *ICCV'95*, pp. 987–994, 1995.

[8] J. Davis, R. Ramamoorthi, and S. Rusinkiewicz. Spacetime stereo: a unifying framework for depth from triangulation. In *CVPR 2003*, 2003.

[9] T. Frohlinghaus and J. M. Buhmann. Regularizing phase-based stereo. In *ICPR'96*, vol. A, pp. 451–455, 1996.

[10] G. Golub and C. F. Van Loan. *Matrix Computation, third edition*. The John Hopkins University Press, 1996.

[11] G. Häusler and D. Ritter. Parallel three-dimensional sensing by color-coded triangulation. *Applied Optics*, 32(35):7164–7169, 1993.

[12] W. Hoff and N. Ahuja. Surfaces from stereo: integrating feature matching, disparity estimation, and contour detection. *IEEE Trans. Pat. Anal. Mach. Int.*, 11(2):121–136, 1989.

[13] E. Horn and N. Kiryati. Toward optimal structured light patterns. In *Intl. Conf. Recent Advances in 3D Digital Imaging and Modeling*, pp. 28–35, 1997.

[14] S. B. Kang, J. Webb, L. Zitnick, and T. Kanade. A multibaseline stereo system with active illumination and real-time image acquisition. In *ICCV'95*, pp. 88–93, 1995.

[15] C. Loop and Z. Zhang. Computing rectifying homographies for stereo vision. In *CVPR'99*, volume I, pp. 125–131, 1999.

[16] J. Mulligan, V. Isler, and K. Daniilidis. Performance evaluation of stereo for tele-presence. In *ICCV 2001*, vol. II, pp. 558–565, 2001.

[17] Y. Nakamura, T. Matsuura, K. Satoh, and Y. Ohta. Occlusion detectable stereo - occlusion patterns in camera matrix. In *CVPR'96*, pp. 371–378, 1996.

[18] M. Proesmans, L. Van Gool, and F. Defoort. Reading between the lines - a method for extracting dynamic 3D with texture. In *ICCV'98*, pp. 1081–1086, 1998.

[19] K. Pulli *et al.* Acquisition and visualization of colored 3D objects. In *ICPR'98*, pp. 11-15, 1998.

[20] K. Sato and S. Inokuchi. Three-dimensional surface measurement by space encoding range imaging. *J. Robotic Systems*, 2:27–39, 1985.

[21] D. Scharstein and R. Szeliski. A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. *Intl. J. Comp. Vis.*, 47(1):7–42, 2002.

[22] E. Schubert. Fast 3D object recognition using multiple color coded illumination. In *Proc. IEEE Conf. Acoustics, Speech, and Signal Processing*, pp. 3057–3060, 1997.

[23] P. Vuylsteke and A. Oosterlinck. Range image acquisition with a single binary-encoded light pattern. *IEEE Trans. Pat. Anal. Mach. Int.*, 12(2):148–164, 1990.

[24] L. Zhang, B. Curless, and S. M. Seitz. Spacetime stereo: shape recovery for dynamic scenes. In *CVPR 2003*, 2003.

[25] T. Zickler, P. N. Belhumeur, and D. J. Kriegman. Helmholtz stereopsis: exploiting reciprocity for surface reconstruction. In *ECCV 2002*, v. III, pp. 869–884, 2002.