

A Fast Algorithm for Depth Segmentation

Christopher R. Wren

Cambridge Research Laboratory
Mitsubishi Electric Research Laboratories
Cambridge, MA 02139

Yuri Ivanov

Media Laboratory
Massachusetts Institute of Technology
Cambridge, MA 02139

Abstract

We present a fast algorithm for depth segmentation. The algorithm uses pre-computed disparity maps to detect regions of the scene that are not at predetermined depths. The form of the algorithm allows it to take advantage of the single instruction, multiple data (SIMD) instruction set extensions that have recently become commonly available in consumer-grade microprocessors. We also present the application of the algorithm to the problem of detecting contact between a foreground object (a hand) and a geometrically static background (a display surface) in real-time video sequences. Due to the geometric nature of the segmentation algorithm, the touch detector is invariant to lighting, color, and motion. It therefore is applicable to interactive front- and back-projected displays. A simple extension to the algorithm allows for the background surface to be given analytically.

1 Introduction

The increasing availability of cheap cameras and high-quality projectors is enabling a wide array of novel interactive space applications. These interfaces present a particular challenge for computer vision algorithms since surfaces lit by bright, high-contrast projection displays may have a continuously changing visual appearance. This means that standard background subtraction techniques that rely on static background appearance will not work. Even worse, front projected displays also cast light on foreground objects, making color tracking and other appearance-based methods difficult or impossible to use.

One of the few constraints on the background that remains in these situations is the geometric configuration of the projection surfaces. By utilizing multiple calibrated cameras, it is possible to take advantage of these geometric constraints to segment such a scene using stereo disparity. Furthermore, we will show that it is possible to perform this operation without computing a dense depth map. Our algorithm instead generates depth segmentation maps directly

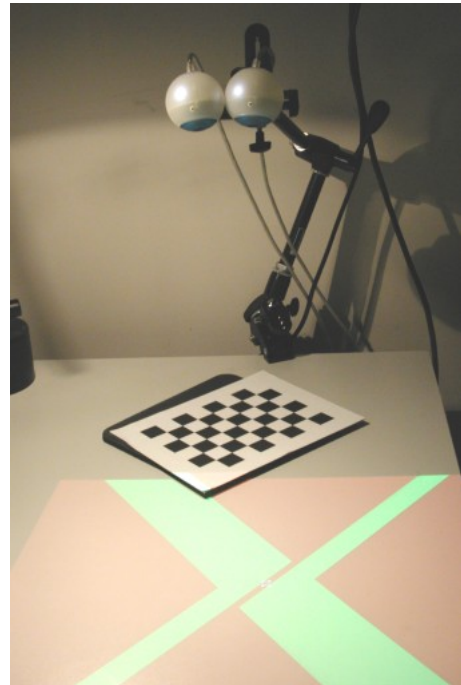


Figure 1: The stereo iBot camera rig, test area with projection display, and calibration widget.

by using precomputed disparity maps to rectify the input images prior to a direct image subtraction. Pixel pairs that are imaging the same small part of the projection surface will exhibit a disparity that satisfies the pre-computed geometric constraint. The precomputed disparity map will then warp those pixels into the same rectified image location[1]. If the cameras have similar luminance and chrominance imaging characteristics, then the difference between these pixels will be small, since they are imaging the very same physical surface patch (assuming non-specular surfaces materials). Areas with high absolute differences indicate regions of the scene where the geometric constraints are not satisfied: regions of the scene that correspond to objects that lie outside the precomputed depth.

Of course, it is not necessary for the disparity maps to correspond to the depths of real surfaces that appear in the scene. It is possible to construct disparity maps that segment virtual surfaces that may correspond to empty space in the real scene. Notions of foreground and background don't make sense in this context since the depth segmentation is selecting areas of the scene that are within some neighborhood of the virtual surface. The rest of the scene is simply outside this neighborhood, either in front of, or behind the virtual surface. We use this fact to move beyond simple background subtraction. Specifically, we present a system called TouchIt that can detect touch events between foreground objects (the user's hand for example) and the background surface. TouchIt uses two depth segmentation maps and simple logical operations to combine them into a touch map.

We implement the depth segmentation algorithm using the Intel Performance Library functions: `iplRemap`, `iplSubtract`, and `iplThreshold`[4]. By using these optimized functions we are able to generate 320x240 depth segmentation maps in less than 10ms on a 1GHz Intel PentiumIII. This allows us to compute several depth segmentations on each image pair in real time, and allows the TouchIt application to run in real time on an off-the-shelf PC. The physical configuration of the TouchIt apparatus is depicted in Figure 1.

The details of the depth segmentation algorithm itself are discussed in Section 3. TouchIt is described in Section 4. The next section reviews the relevant literature.

2 Related Work

The most common approach to segmenting objects from the background is some form of background subtraction. For example, in [11, 10, 2] authors use statistical texture properties of the background observed over extended period of time to construct a model of the background, and use this model to decide which pixels in an input image do not fall into the background class. The fundamental assumption of the algorithm is that the background is static, or at most slowly varying, in all respects: geometry, reflectance, and illumination. These algorithms show low sensitivity to slow lighting changes, to which they need to adapt.

[9] uses a subtraction method which has an explicit illumination model. The model in this case is the eigenspace, describing a range of appearances of the scene under a variety of lighting conditions. The method works extremely well for outdoor environments with static geometry, but unforeseen illumination conditions, which are not taken into account while building the model, will degrade the performance of the algorithm.

None of the above algorithms are designed to handle rapid lighting changes, such as a dynamic, high-contrast

projection display.

More closely related to the technique presented in this paper are those based upon geometry. Gaspar, et. al. use geometrical constraints of a ground plane in order to detect obstacles on the path of a mobile robot [3]. Okutomi and Kanade employ special purpose multi-baseline stereo hardware to compute dense depth maps in real-time [8]. Provided with a background disparity value, the algorithm can perform real-time depth segmentation or "z-keying" [7]. The only assumption of the algorithm is that the geometry of the background does not vary. However, the computational burden of computing dense, robust, real-time stereo maps requires great computational power.

The most closely related work was done by Ivanov and Bobick when they employed a similar technique for segmentation in the presence of theatrical lighting [6]. They do take advantage of precomputed disparity for segmentation, but did not move beyond segmentation to more general depth-aware applications as we do with the TouchIt application.

3 Fast Depth Segmentation

Typically, to estimate stereo disparity corresponding to an image location (x, y) in the left image of a stereo pair, one must search for the image location (x', y') in the other (right) image of the pair where the image data $I_r(x', y')$ corresponds to the image data in the first image $I_l(x, y)$. The estimated stereo disparity $\hat{d}(x, y)$ is the difference between these two locations:

$$\hat{d}(x, y) = \begin{bmatrix} \hat{x}' - x \\ \hat{y}' - y \end{bmatrix} \quad (1)$$

and together with the calibration data can be used to hypothesize that these two image patches correspond to the same surface patch at a calculable distance from the cameras. The fast depth segmentation (FDS) algorithm works in exactly the opposite way. FDS requires an image pair plus a pre-computed disparity map. The disparity map indicates, for every location in one image of the pair, the estimated offset (disparity) $\hat{d}(x, y)$ to the location in the other image of the pair where the correspondence will be found, if the surface of some object currently occupies the point in space at the implied distance from the cameras. The set of all such disparities for a given image pair is \hat{D} :

$$\hat{D} = \begin{bmatrix} \hat{d}(x_1, y_1) & \hat{d}(x_2, y_2) & \cdots & \hat{d}(x_m, y_m) \end{bmatrix} \quad (2)$$

\hat{D} specifies a map that may be used to warp one image of a pair, rectifying it with respect to the other image of the

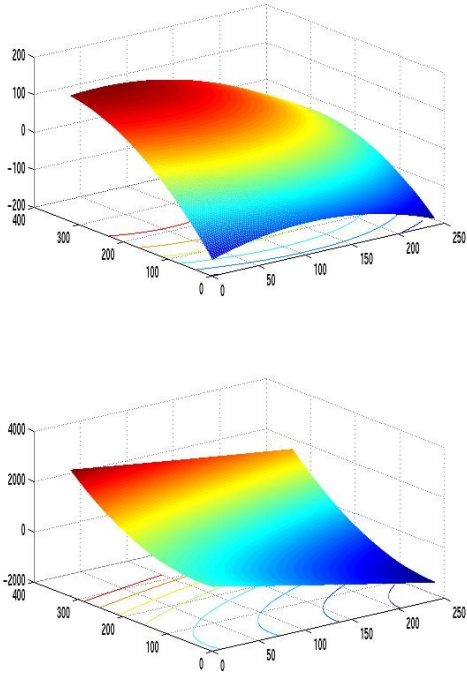


Figure 2: An example of the pixel displacement map generated by the polynomial estimate: **Top**: x displacements on the z -axis as a function of pixel location. **Bottom**: y displacements.

pair such that points at the predetermined depth will map to identical image locations. For a visual example of this process see the rectified and composited images in the middle of Figures 4, 5, and 6.

We construct this dense map by fitting a two-dimensional polynomial to a relatively small set of known correspondences. We use the Intel Open Computer Vision Library[5] chess board finder functions to acquire these correspondences by placing a chess board at the the desired depth plane. The form of the polynomial is:

$$\hat{\mathbf{d}}(x, y) = \hat{\mathbf{A}}\mathbf{x}(x, y) \quad (3)$$

where $\mathbf{x}(x, y)$ is the power expansion:

$$\mathbf{x}(x, y) = \begin{bmatrix} x^2 \\ y^2 \\ xy \\ x \\ y \\ 1 \end{bmatrix} \quad (4)$$

Given a set of m correspondence points we construct the

matrix of powers:

$$\mathbf{X} = \begin{bmatrix} \mathbf{x}(x_1, y_1) & \mathbf{x}(x_2, y_2) & \cdots & \mathbf{x}(x_m, y_m) \end{bmatrix} \quad (5)$$

and a matrix of estimated disparities for those points:

$$\hat{\mathbf{D}} = \begin{bmatrix} \hat{\mathbf{d}}(x_1, y_1) & \hat{\mathbf{d}}(x_2, y_2) & \cdots & \hat{\mathbf{d}}(x_m, y_m) \end{bmatrix} \quad (6)$$

Then $\hat{\mathbf{A}}$ can be recovered by least squares:

$$\hat{\mathbf{A}} = (\mathbf{X}\mathbf{X}^T)^{-1}\mathbf{X}^T\hat{\mathbf{D}} \quad (7)$$

Equation 3 can then be applied to each image location to compute an approximate dense disparity map as shown in Figure 2.

The Intel Image Processing Library provides an optimized function that performs general image warps[4]. Given an input image and modified form of $\hat{\mathbf{D}}$, this function will produce the rectified image that we set out to compute.

The remainder of the FDS algorithm consists of simply subtracting one image of a pair from the rectified version the the other image. If the scene satisfies the geometric constraint, then the magnitude of the difference will be near zero everywhere. Where the geometric constraint fails to hold, this magnitude will be large.

It is unimportant how the disparity map is calculated. So long as $\hat{\mathbf{D}}$ is sufficiently accurate, the warp and subtract will generate a measure of how well the scene satisfies the encoded geometric constraint. The above estimation algorithm avoids requiring accurate estimation of extrinsic and intrinsic camera geometry by directly estimating the warp induced on an observed planar calibration widget. Given sufficiently accurate camera calibration it would be possible to construct disparity maps corresponding to arbitrary virtual surfaces, including piecewise-planar or even non-planar surfaces. These generalizations would have no impact on run-time efficiency. They only change the form of the off-line computation of $\hat{\mathbf{D}}$.

4 Touch It

TouchIt utilizes FDS to do two things. First: segment the hand from the projection surface even in the presence of arbitrary projected light. Second: determine if the user is touching the display surface with the tip of their finger. Figure 3 shows TouchIt in action. The white circle in the bottom image of Figure 3 indicates the detection and localization of a touch event. Figure 7 shows TouchIt operating in the presence of high-contrast projected light.

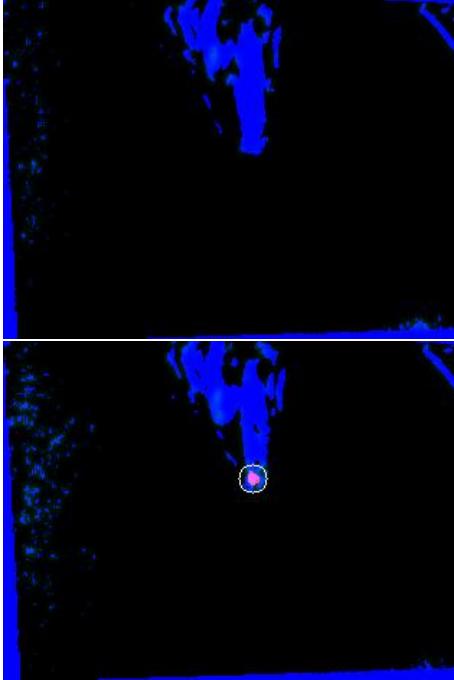


Figure 3: **Top:** segmented hand above the surface. **Bottom:** contact point marked on lowered finger tip..

The first task is a direct application of FDS. The calibration widget is placed flat on the projection surface, and this induces a plane approximately coincident with the surface. In practice, observed surfaces somewhat near this plane will be marked as satisfying the constraint even if they are not strictly coincident with it. Areas that do not satisfy the constraint are unambiguously part of the foreground because they are not in the plane of the projection surface, and they cannot be behind it (the projection surface is solid and opaque).

The second task is accomplished by running a second instance of FDS in parallel with the first implementing a plane constraint that lies just above the surface of the table. To accomplish this, the calibration widget is placed on a mouse pad (approximate thickness of 4mm) to facilitate building the disparity map.

The output of the two instances of FDS are compared. Pixels that are determined to be above the projection surface, but are not marked as being outside the second, slightly higher plane, must correspond to surfaces within a very narrow band just above the projection surface. The top of the finger, when the finger is resting on the projection surface, satisfies this constraint.

Well understood methods were sufficient to locate the single large region of non-overlap between the two maps. These areas are automatically marked with a white circle and appear red in Figure 3 and Figure 7.

5 Results and Conclusions

On a 1GHz Intel Pentium IIIEB with 320x240 images, the fundamental computation requires 7ms per depth segmentation, averaged over 1000 frames. That includes the remap call, the difference magnitude computation, a 3x3 blur and a threshold operation. For the TouchIt application we found it necessary to perform morphological operations to remove noise from the segmentation map. These operations add another 2ms to the operation, for a total of 9ms. An example of the algorithms robustness to projected displays can be seen in Figure 7.

We look forward to refining the construction of the disparity map by taking advantage of lens distortion calibration[5], as well as more sophisticated models of projective geometry[1]. The implementation would also benefit from color calibration of the cameras. Being able to treat the color channels separately in the difference magnitude computation would provide better discrimination, and therefore cleaner segmentation maps, compared to the greyscale process. Despite the need for improvement in these areas, the algorithm produces usable depth segmentation maps within a small fraction of the computational cost that would be required to produce the same answer using an approach based on full stereo.

The fast depth segmentation algorithm presented here provides a computationally cheap way to take advantage of stereo disparity constraints in the environment. The TouchIt application takes advantage of this algorithm to extract a foreground segmentation in a visual environment that would break most, if not all commonly used segmentation algorithms. TouchIt then goes beyond just segmentation to recover a measure of physical proximity between observed objects that would be difficult to compute using other visual routines within similar computational bounds.

References

- [1] Olivier Faugeras and Quang-Tuan Luong. *The Geometry of Multiple Images*. MIT Press, 2001. with contributions from Theo Papadopoulos.
- [2] Nir Friedman and Stuart Russell. Image segmentation in video sequences: A probabilistic approach. In *Thirteenth Conference on Uncertainty in Artificial Intelligence*, 1997.
- [3] Jose Gaspar, Jose Santos-Victor, and Joao Sentieiro. Ground plane obstacle detection with a stereo vision system. *International workshop on Intelligent Robotic Systems*, 1994.
- [4] Intel Corporation. *Intel Image Processing Library Reference Manual*, 2000. document number 663791-005.

- [5] Intel Corporation. *Open Source Computer Vision Library Reference Manual*, 2001.
- [6] Yuri Ivanov, Aaron Bobick, and John Liu. Fast lighting independent background subtraction. *International Journal of Computer Vision*, 37(2):199–207, June 2000.
- [7] T. Kanade. A stereo machine for video-rate dense depth mapping and its new applications. In *Proc. of Image Understanding Workshop*, pages 805 – 811, Palm Springs, California, 1995.
- [8] M. Okutomi and Takeo Kanade. A multiple-baseline stereo. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 15(4):353–363, April 1993.
- [9] N. Oliver, B. Rosario, and A. Pentland. A bayesian computer vision system for modeling human interactions. In *Proceedings of ICVS99*, Gran Canaria, Spain, 1999.
- [10] Chris Stauffer and W.E.L. Grimson. Adaptive background mixture models for real-time tracking. In *Proc. of CVPR–99*, pages 246–252, Ft. Collins, CO, 1999.
- [11] C. Wren, A. Azarbayejani, T. Darrell, and A. Pentland. Pfunder: Real-time tracking of the human body. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 19(7):780–785, 1997.

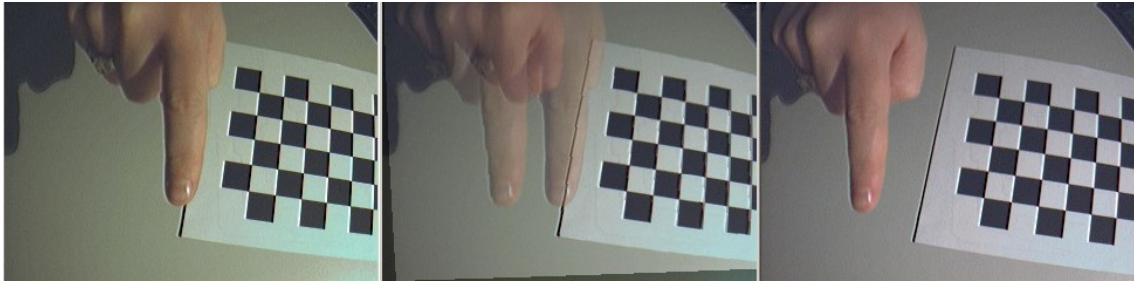


Figure 4: **Left:** left camera view. **Middle:** rectified, fused images. **Right:** right camera view.

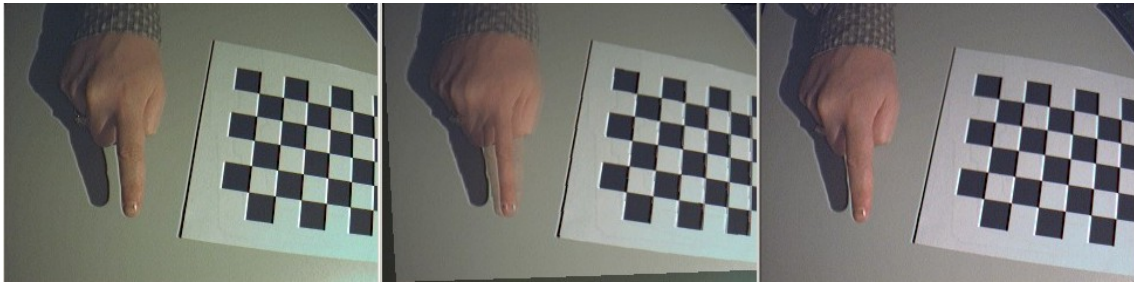


Figure 5: A hand just above the segmentation surface.

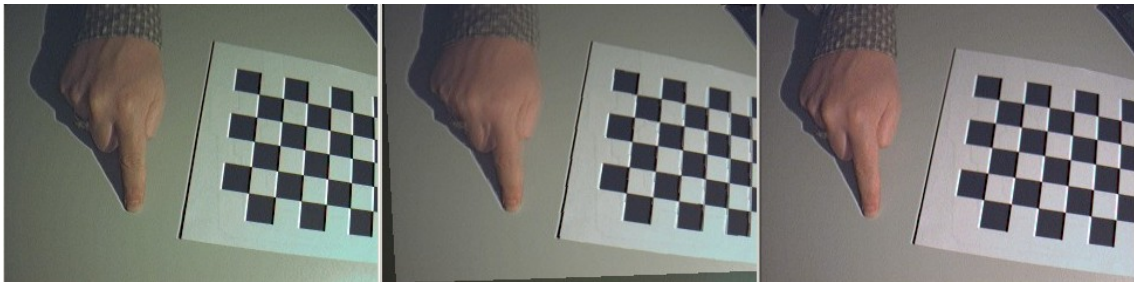


Figure 6: A touch event.

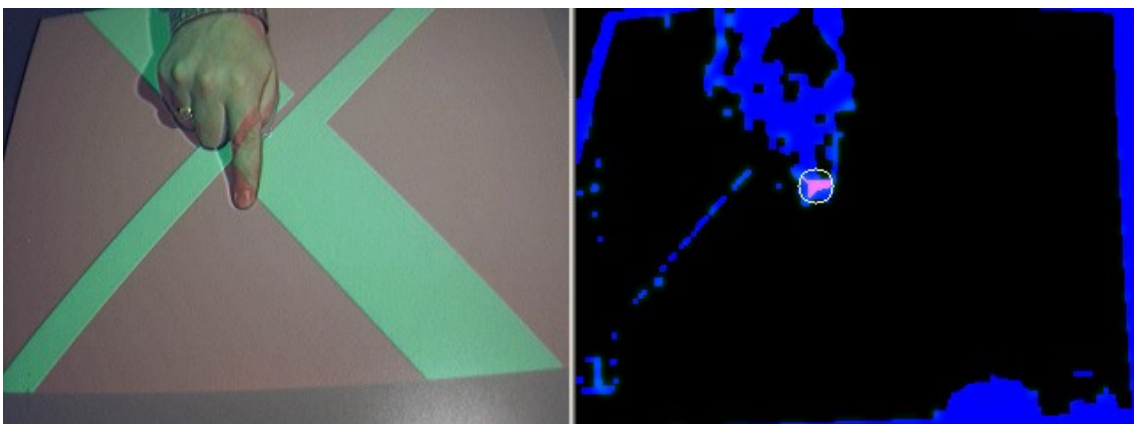


Figure 7: Image pair showing performance under projected illumination.