Volumetric Operations with Surface Margins

Christopher R. Wren
Yuri A. Ivanov

TR-2001-47   January 2002

Abstract

Cheap cameras and fast processors have made it possible to visually exploit geometric constraints in real time. It has been shown that the fast depth segmentation (FDS) algorithm successfully exploits geometric constraints to perform visual foreground/background segmentation in environments where other vision routines fail. This paper presents new insights into the operation of the FDS algorithm that lead to the concept of a virtual surface margin. We then show how surface margins can be used to extend the FDS algorithm and thereby enable a class of logical volume operations that go far beyond simple background segmentation tasks. An example application called TouchIt is demonstrated. Touchit utilizes surface margins to create a virtual volume configuration that is useful for detecting physical proximity to a surface. We also present refinements in the the implementation of the FDS algorithm that make these volumetric computations practical for interactive applications by taking advantage of the single instruction, multiple data (SIMD) instruction set extensions that have recently become commonly available in consumer-grade microprocessors.

IEEE Computer Vision and Pattern Recognition Technical Sketches, December 2001. Kauai, Hawaii.

Submitted September 2001, revised and released December 2001.

Volumetric Operations with Surface Margins

Christopher R. Wren
Cambridge Research Laboratory
Mitsubishi Electric Research Laboratories
Cambridge, MA 02139
wren@merl.com

Yuri A. Ivanov
Media Laboratory
Massachusetts Institute of Technology
Cambridge, MA 02139
yivanov@media.mit.edu

Abstract

Cheap cameras and fast processors have made it possible to visually exploit geometric constraints in real time. It has been shown that the fast depth segmentation (FDS) algorithm successfully exploits geometric constraints to perform visual foreground/background segmentation in environments where other vision routines fail. This paper presents new insights into the operation of the FDS algorithm that lead to the concept of a virtual surface margin. We then show how surface margins can be used to extend the FDS algorithm and thereby enable a class of logical volume operations that go far beyond simple background segmentation tasks. An example application called TouchIt is demonstrated. Touchit utilizes surface margins to create a virtual volume configuration that is useful for detecting physical proximity to a surface. We also present refinements in the the implementation of the FDS algorithm that make these volumetric computations practical for interactive applications by taking advantage of the single instruction, multiple data (SIMD) instruction set extensions that have recently become commonly available in consumer-grade microprocessors.

1 Introduction

By utilizing multiple calibrated cameras, it is possible to take advantage of geometric constraints to segment scenes. Indeed if the geometry of the background is known one can establish depth at every pixel in the image and compare that to the static geometry of the empty scene. However, this process involves computing a dense depth map of each pair of frames coming from the stereo camera pair. This can be very computationally expensive.

The fast depth segmentation (FDS) algorithm[5, 2] performs depth segmentation without the need to compute full depth maps of the scene. The algorithm essentially amounts to a table lookup and can be performed at frame rate. The algorithm instead generates depth segmentation maps directly by using pre-computed disparity maps to rectify the input images prior to an image subtraction. Subtraction results in a depth residual image that may then be thresholded. The FDS algorithm is described briefly in Section 3.

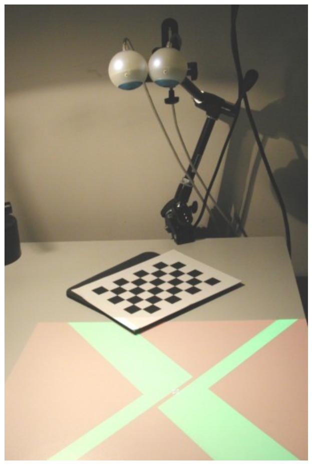
Figure 1: The stereo iBot camera rig, test area with projection display, and calibration widget.

Theoretically, these disparity maps define virtual surfaces in physical space. Practically, due to the nature of real cameras and real scenes, the threshold can never be set small enough to realize an infintesmally thin virtual surface. We call this resulting thickness the *virtual surface margin*, and we explicitly use it to enable arbitrary volumetric operations. The virtual surface margin is described in more detail in Section 4.

Of course, it is not necessary for these virtual surfaces to correspond to real surfaces that appear in the scene. It is possible to construct disparity maps that segment arbitrary virtual surfaces that may correspond to empty space in the real scene. Section 5 describes some useful techniques for building such disparity maps.

Combining these arbitrary virtual surfaces with the idea of virtual surface margins and simple logical operations, we move beyond simple background subtraction to more complex geometric constraint operations on volumes. 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 create a touch map in real time on an off the shelf PC. The TouchIt apparatus is depicted in Figure 1, and described in more detail in Section 6.

2 Related Work

Most computer vision algorithms dealing with objects and recognition of their motion patterns begin the processing with some form of object segmentation. The literature focuses on building statistical models of appearance of the scene[10, 9, 1, 8]. All of these algorithms assume the scene is basically stationary with respect to geometry, reflectance, and illumination. They generally are not designed to handle rapid lighting changes, as observed in the presence of a dynamic, high-contrast projection display, for example.

Techniques based upon geometry only rely on the geometric stability of the scene. Gaspar, et. al. use geometrical constraints of a ground plane in order to detect obstacles on the path of a mobile robot [2]. Okutomi and Kanade employ special purpose multi-baseline stereo hardware to compute dense depth maps in real-time [7] and to perform real-time depth segmentation[6]. Closely related to this work is the fast depth segmentation work of Ivanov and Bobick in the context of segmenting a scene in the presence of theatrical lighting [5].

All of these geometric algorithms treat the reality of the virtual surface margin as noise, and therefore are not able to 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 main image of a stereo pair, one must search for the image location (x^r, y^r) in the other (reference) image of the pair where the image data $\mathbf{I}^r(x^r, y^r)$ corresponds to the image data in the main image $\mathbf{I}(x, y)$. The estimated stereo disparity $\mathbf{d}(x, y)$ is the difference between these two locations, 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 predicted offset, $\mathbf{d}(x, y)$ to the location in the other image of the pair where the correspondence will be found, if the surface of some object currently occupies

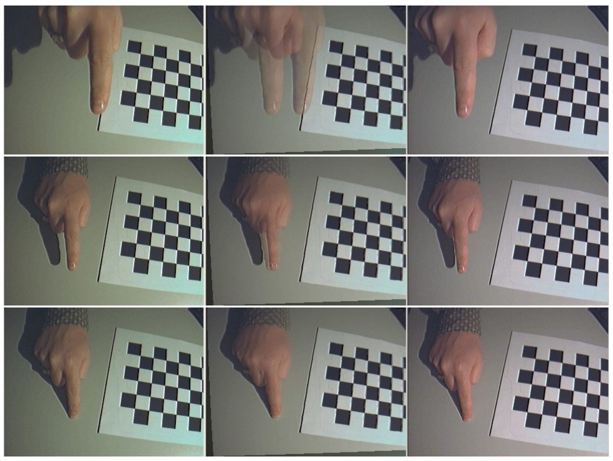

Figure 2: **Left**: left camera view. **Middle**: rectified, fused images. **Right**: right camera view. **Top**: A hand far from the surface. **Middle**:A hand just above the segmentation surface. **Bottom**: A touch event.

the point in space at the implied distance from the cameras. The set of all such disparities for a given image pair is $\mathbf{D}$.

$\mathbf{D}$ specifies a map that may be used to warp one image of a pair, rectifying it with respect to the other image of the 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 Figure 2.

After the reference image is warped to correspond to the main camera image, pixel-by-pixel subtraction results in a depth residual image.

4. Virtual Surface Margin

If there is a real surface coincident with the virtual surface defined by the disparity map $\mathbf{D}$, then a pixel from the main image and a corresponding pixel from the reference image will image the exact same physical surface patch. In this case the pixel measurements will be substantially identical, and the residual will be dominated by imaging noise. This is the case for the surface of the table table illustrated in Figure 2,

For the case where the real surface is slightly nearer or farther from the cameras than the virtual surface, similar to the situation of the fingertip in the bottom row of Figure 2, pairs of pixels will image slightly different parts of the surface, and the pixel measurements therefore differ slightly. Consequently, the residual will be greater than in the above case.

As the real surface moves farther from the virtual surface, less overlap exists in a pair, until the case where a pair of pixels image completely different patches of the surface, or possibly different surfaces altogether, and the residual is dominated by surface properties.

Therefore, for any given threshold, noise, geometry, and surface properties combine to form a margin surrounding the virtual surface. There will be thin slice of physical space, a virtual volume, that will be perceived as part of a single unified surface.

Practically this thickness means that if these surfaces are constructed to exist near each other in physical space, and the FDS outputs are combined with Boolean operations, then it is possible to perform complex volumetric depth segmentation operations.

5 Disparity Calculations

In order to compute the disparity map and perform the fast depth segmentation we can proceed in one of two ways: a) from known correspondence points, compute the map directly, using the known point-correspondences and the smoothness constraints on the resulting surface; b) compute the disparity map analytically from the Intrinsic and extrinsic parameters of each camera in the stereo pair. We show each technique in the remainder of this section.

5.1 Direct Interpolation Technique

By a variety of methods we can obtain a sparse set of point correspondences from the camera pair (we use the Intel Open Computer Vision Library[4] chess board finder functions to acquire these correspondences by placing a chess board at the the desired depth plane). Since the smooth surface will have a smooth disparity map, we can use a smooth continuous approximation of the point set to calculate the dense disparity map between the cameras in the pair.

We construct this dense map by a polynomial interpolation of the known set of point correspondences. The disparity, $\mathbf{d}(x, y)$ is approximated by the following linear system:

$$\mathbf{d}(x, y) = \Lambda \widetilde{\mathbf{x}}(x, y) \tag{1}$$

where Λ is the unknown matrix of coefficients, and $\widetilde{\mathbf{x}}(x, y)$ is the power expansion of $\mathbf{x} = [x, y]^T$: $\widetilde{\mathbf{x}}(x, y) = \left[x^2 y^2 1\right]$. Given a set of m correspondence points it is possible to estimate Λ, via least squares. Equation 1 can then be applied to each image location to compute an approximate dense disparity map.

5.2 Analytic Technique

The analytic method is useful for constructing virtual surfaces in arbitrary locations in space relative to the cameras. We begin with introducing some notation used in the rest of this section. Let $\mathbf{m}$ be a $2D$ location in image coordinates, $\tilde{\mathbf{m}}$ - an $3D$ location in homogeneous image coordinates, $\mathbf{M}$ a $3D$ physical coordinates of a point on the imageing surface in the "world" coordinate system, and $\tilde{\mathbf{M}}$ its $4D$ homogeneous version.

Widely available camera calibration techniques (which are not the focus of this paper and, therefore, are not discussed) typically make available a set of matrices, A - the intrinsics, and a pair of extrinsic matrices R - the rotation, and $\mathbf{t}$ - the translation vector, that relate the world coordinate system to the coordinate system centered at the optical center of the camera, $\mathbf{O}$. Under these transformations the following relation maps points in the physical world to the image pixel positions:

$$\tilde{\mathbf{m}} = A\,[R|\mathbf{t}]\,\tilde{\mathbf{M}} \tag{2}$$

Without loss of generality, let us assume that we are interested in computing the disparity map for a plane which has a constant value of $Z = C$ in the world coordinate system. We start by expressing a pixel position in the world coordinate frame:

$$\mathbf{r}_w = (AR)^{-1}\,\tilde{\mathbf{m}} - R^{-1}\mathbf{t} \tag{3}$$

This induces a ray from $\mathbf{O}_w$, the optical center of the camera, through $\mathbf{r}_w$. We are interesting in the the surface that is imaged by this optical ray:

$$L(s) = \mathbf{r}_w + s\,(\mathbf{r}_w - \mathbf{O}_w) \tag{4}$$

We solve for this point in this case by intersecting the ray with the point on the plane $Z = C$, so the point where $L^z(s) = C$, which we will call $\mathbf{M}_C$.

Forward application of the camera model allows us to compute the image location, $\mathbf{m}^r$, of this point in the reference camera:

$$\tilde{\mathbf{m}}^r_C = A^r\,[R^r|\mathbf{t}^r]\,\tilde{\mathbf{M}}_C \tag{5}$$

And, finally, the disparity for the point $\mathbf{m}$ in the main camera view is computed:

$$\mathbf{D}_C = \mathbf{m}^r_C - \mathbf{m} \tag{6}$$

In order to compute the dense disparity map we perform this calculation for every pixel in the view of the main camera. This computation is performed only once, before the run of the depth segmentation algorithm.

6 Touch It

TouchIt utilizes the Fast Disparity Segmentation (FDS) algorithm to analyze two virtual planes, a lower plane P_L and an upper plane P_U. P_L is the plane $Z = 0$, where the world coordinate system is defined such that $Z = 0$ is approximately coincident with the surface of a table top projection

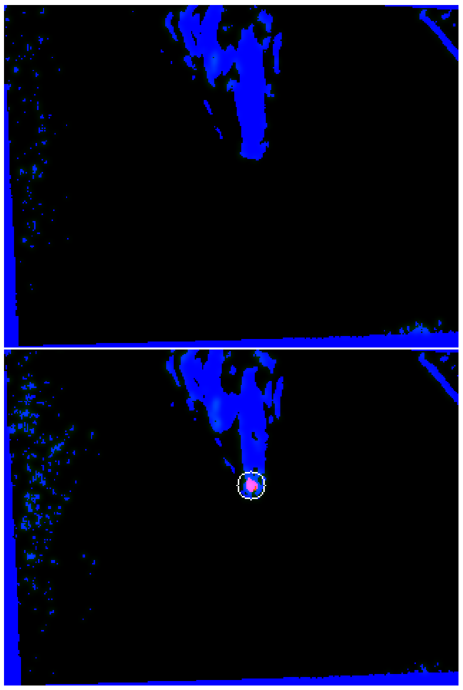

Figure 3: **Top**: segmented hand above the surface. **Bottom**: contact point marked on lowered finger tip..

surface. The second plane P_U is defined to be a small distance above the table surface, $Z = \epsilon > 0$.

If FDS_L and FDS_U are the Boolean result of the FDS algorithm for the planes P_L and P_U, respectively, and the FDS output is true where the depth constraint is satisfied, then the fundamental operation of the touch detector is to locate areas marked false in FDS_L but true in FDS_U. This relationship identifies regions where a non-table object must occupy the margin.

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

7 Results and Conclusions

We implement the depth segmentation algorithm using the Intel Performance Library functions: `iplRemap`, `iplSubtract`, and `iplThreshold`[3]. On a 1GHz Intel Pentium IIIEB with 320x240 images, the per-frame depth segmentation computation requires 7ms. 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 per virtual surface.

We look forward to refining the implementation relative to issues such as lens distortion precise geometric calibration, and color calibration. 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. TouchIt then goes beyond just segmentation to recover a measure of physical proximity between observed objects that would be difficult to compute using other vision routines within similar computational bounds.

References

[1] Nir Friedman and Stuart Russell. Image segmentation in video sequences: A probabilistic approach. In *Thirteenth Conference on Uncertainty in Artificial Intelligence*, 1997.

[2] Jose Gaspar, Jose Santos-Victor, and Joao Sentieiro. Ground plane obstacle detection with a stereo vision system. *International workshop on Intelligent Robotic Systems*, 1994.

[3] Intel Corporation. *Intel Image Processing Library Reference Manual*, 2000. document number 663791-005.

[4] Intel Corporation. *Open Source Computer Vision Library Reference Manual*, 2001.

[5] Yuri Ivanov, Aaron Bobick, and John Liu. Fast lighting independent background subtraction. *International Journal of Computer Vision*, 37(2):199–207, June 2000.

[6] 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.

[7] M. Okutomi and Takeo Kanade. A multiple-baseline stereo. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 15(4):353–363, April 1993.

[8] N. Oliver, B. Rosario, and A. Pentland. A bayesian computer vision system for modeling human interactions. In *Proceedings of ICVS99*, Gran Canaria, Spain, 1999.

[9] 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.

[10] C. Wren, A. Azarbayejani, T. Darrell, and A. Pentland. Pfinder: Real-time tracking of the human body. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 19(7):780–785, 1997.