massachusetts institute of technology — artificial intelligence laboratory

Range Segmentation Using Visibility Constraints

Leonid Taycher and Trevor Darrell

AI Memo 2001-024 September 2001

Abstract

Visibility constraints can aid the segmentation of foreground objects observed with multiple range images. In our approach, points are defined as foreground if they can be determined to occlude some *empty space* in the scene. We present an efficient algorithm to estimate foreground points in each range view using explicit epipolar search. In cases where the background pattern is stationary, we show how visibility constraints from other views can generate virtual background values at points with no valid depth in the primary view. We demonstrate the performance of both algorithms for detecting people in indoor office environments.

1. Introduction

Object segmentation is an important preliminary step for many high-level vision tasks, including person detection and tracking. State-of-the-art systems [16, 3, 1, 7] use foreground/background classification followed by pixel clustering and analysis. These systems commonly maintain a background model and label all pixels that differ significantly from this model as foreground.

Ideally, these systems should be robust to rapid illumination variation, such as from outdoor weather or indoor video projection systems. Several segmentation methods have been proposed which use background models based on color/intensity [16, 15, 14], stereo range [9, 1] or both [8]. Generally, non-adaptive color-based models suffer from varying illumination. Adaptive color models [14] are more stable under lighting changes, but can erroneously incorporate objects that stop moving into the background model. Range-based background models can be illumination invariant, but are usually sparse. To avoid ambiguity at undefined background values (and the resulting illumination dependence [4]), they have been either used in conjunction with color models [8], or are built using observations from widely varying illumination and imaging conditions [4]. In this paper we show how visibility constraints from other range images can aid segmentation.

Our approach to foreground segmentation is to combine *free space* constraints found from multiple stereo range views. We decide if a given pixel is "foreground" by checking whether there is any *free space* behind it, as seen from other range views. We scan the set of epipolar lines in the other views corresponding to the given pixel, and test whether there are range points on the epipolar lines which indicate empty space behind the given point.

This is a similar computation to algorithms proposed for the rendering of image-based visual hulls [11]. The key difference is that our method takes as input unsegmented noisy range data and evaluates 3-D visibility per ray, while the visual hull method presumes segmented color images as input and simply identifies non-empty pixels along the epipolar lines in other views. Also related are space carving and coloring methods [10, 13], which split the space into *voxels* and use color consistency across multiple cameras to locate opaque voxels and to detect *free space*. These methods are quite general, and work with an arbitrary set of monocular views. They also require the construction of a volumetric representation of the scene for reconstruction or segmentation. We are interested in algorithms that perform segmentation solely in the image domain, without computing a volumetric reconstruction. We believe ours is the first method for range image segmentation using image-based (non-voxel) freespace computation.

In this paper we develop two complimentary segmentation algorithms that use visibility constraints. The first is an instantaneous foreground detection algorithm, which is independent of previous time points and does not presume scene or illumination constancy. The second assumes a stationary scene and a background range model per view, and generates virtual background values at pixels which would otherwise have had insufficient contrast to have valid range.

In the next section we describe our method for using "complimentary" camera(s) to determine whether a single 3-D point, visible by a "primary" camera, occludes any *free space*. We describe how to cluster such points, and to determine whether two clusters provably belong to separate objects. We then propose a method for creating dense virtual backgrounds for stationary scenes. Finally, we demonstrate the results using our algorithm tracking people in an indoor office environment.

2. Foreground segmentation

We wish to segment objects that are not attached to any "background" surface other than the floor, by detecting pixels that occlude some *empty space*.

When a 3D point $\mathbf{P}$ is imaged by a range sensing device (e.g. stereo camera) C^1 (Figure 2.1), we know that there exists a nontransparent material at that point, and that all points between it and the camera's center of projection are transparent. But C^1 is unable to provide any information about what lies behind $\mathbf{P}$ on the same projection ray. On the other hand, we can use the observation $\mathbf{B}$ of the appropriately located range sensor C^2 to discover that $\overline{\mathbf{P}}$, which is occluded by $\mathbf{P}$ in C^1's view is transparent.

An ideal (rectified) stereo rig may be completely described by the baseline B, focal length f and the image coordinates of the principal point (c_x, c_y). The following equations describe a relation between point (x, y) in the disparity image I_D and the corresponding 3-D location (X, Y, Z).

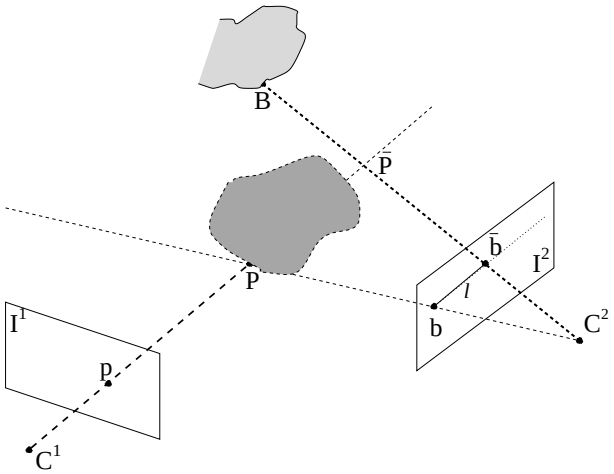


Figure 2.1: Visibility-based segmentation. Observation of $\mathbf{B}$ in C^2 allows us to infer that $\mathbf{P}$ in C^1 is foreground. Point $\mathbf{B}$ visible in I^2 (projecting to C^2 disparity point $\overline{\mathbf{b}}$) lies behind point $\overline{\mathbf{P}}$ relative to C^2, and thus provides evidence for existence of *free space* behind $\mathbf{P}$ (projecting to $\mathbf{b}$) by demonstrating that $\overline{\mathbf{P}}$ is transparent. Line l contains the oversilhouette for the part of an object lying along ray $[\mathbf{C^1}, \mathbf{P})$

$$\begin{cases} \overline{x} = x - c_x = f\dfrac{X}{Z} \\ \overline{y} = y - c_y = f\dfrac{Y}{Z} \\ d = I_D(x,y) = f\dfrac{B}{Z} \end{cases} \tag{1}$$

As has been shown in [5], the disparity space is a projective space, and we can write the transformation from disparity to camera-centered Euclidean projective coordinates as

$$\begin{pmatrix} X \\ Y \\ Z \\ W \end{pmatrix} = \mathbf{T_C}(f, B, c_x, c_y) \begin{pmatrix} x \\ y \\ d \\ w \end{pmatrix} \tag{2}$$

where

$$\mathbf{T_C}(f, B, c_x, c_y) = \begin{pmatrix} B & 0 & 0 & -c_x B \\ 0 & B & 0 & -c_y B \\ 0 & 0 & 0 & fB \\ 0 & 0 & 1 & 0 \end{pmatrix} \tag{3}$$

In the rest of the paper we refer to $(x\ y\ d)^T$ and $(X\ Y\ Z)^T$ ($(x\ y\ d\ w)^T$ and $(X\ Y\ Z\ W)^T$) as disparity and Euclidean (projective) spaces, and denote them D and E respectively.

The general setup of the imaging system assumed in our algorithm is presented in the Figure 2.1. There are two calibrated stereo rigs, C^1 ("primary") and C^2 ("complimentary"), with disparity-to-Euclidean camera coordinate transforms $\mathbf{T}_{\mathbf{D^1}}^{\mathbf{E^1}} = \mathbf{T_C}(f^1, B^1, c_x^1, c_y^1)$ and $\mathbf{T}_{\mathbf{D^2}}^{\mathbf{E^2}} = \mathbf{T_C}(f^2, B^2, c_x^2, c_y^2)$, image planes I^1 and I^2, and disparity images I_D^1 and I_D^2 respectively. The Euclidean coordinate transform between views is $\mathbf{T}_{\mathbf{E^1}}^{\mathbf{E^2}}$.

2.1. Pixel-level Segmentation

The first stage of our foreground segmentation algorithm applied to the disparity image of the camera C^i is to determine which 3-D points visible by C^i occlude some *free space*. That is, for each disparity point $\mathbf{p} = (p_x\ p_y\ I_D^i(p_x, p_y)\ 1)^T$, where

$I^i_D(p_x, p_y)$ is valid, we check whether there is some *free space behind* the point $\mathbf{P}$,

$$\begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} = \mathbf{P} \simeq \mathbf{T}^{\mathbf{E^i}}_{\mathbf{D^i}} \mathbf{p} \tag{4}$$

where $\simeq$ denotes equality up to a scale factor.

We say that point $\overline{\mathbf{P}}$ is *behind* point $\mathbf{P}$ relative to rig C^i, if $\overline{\mathbf{P}}$ lies on the optical ray $[\mathbf{C^i}, \mathbf{P})$, and $|\overline{\mathbf{P}} - \mathbf{C^i}| > |\mathbf{P} - \mathbf{C^i}|$.

Let point $\overline{\mathbf{P}}$ be *behind* $\mathbf{P}$ relative to camera C^i. We project $\overline{\mathbf{P}}$ into $D^j, i \neq j$ to obtain point $\overline{\mathbf{b}}$, such that

$$\begin{pmatrix} \overline{b}_x \\ \overline{b}_y \\ \overline{b}_d \\ 1 \end{pmatrix} = \overline{\mathbf{b}} \simeq (\mathbf{T}^{\mathbf{E^j}}_{\mathbf{D^j}})^{-1} \mathbf{T}^{\mathbf{E^j}}_{\mathbf{E^i}} \overline{\mathbf{P}} \tag{5}$$

The points $(\overline{b}_x, \overline{b}_y)$, corresponding to all possible $\overline{\mathbf{P}}$s, form a ray of the line epipolar to (p_x, p_y) in I^j (passing through projection of the $\mathbf{C^i}$ to I^j and $\mathbf{b}$).

If the disparity value $I^j_D(\overline{b}_x, \overline{b}_y)$ corresponding to any of such $\overline{\mathbf{b}}$s is *valid* and is *smaller* than $\overline{b}_d$ (i.e. some point $\mathbf{B}$ *behind* $\overline{\mathbf{P}}$ relative to camera C^j is visible by C^j), then point $\overline{\mathbf{P}}$ is transparent, and may be assumed to belong to *free space*. When we can find cases where

$$I^j_D(\overline{b}_x, \overline{b}_y) < \overline{b}_d \tag{6}$$

we consider that point to be *evidence* for $\mathbf{P}$ (correspondingly $\mathbf{p}$) belonging to foreground. The fast algorithm to generate all points $\overline{\mathbf{b}}$ corresponding to a particular $\mathbf{p}$ is detailed in Section 4.

In the current implementation of the algorithm, we use the number of found *evidence* pixels as a measure of certainty that point $\mathbf{p}$ belongs to foreground. If more than one "complimentary" camera is available, then the results from each of them may be combined to provide more robust output. We compute a map of the number of observed occluded free-space points:

$$\begin{aligned} \theta(\overline{\mathbf{b}}) &= \begin{cases} 1 & I^j_D(\overline{b}_x, \overline{b}_y) \text{ is valid and } \lambda I^j_D(\overline{b}_x, \overline{b}_y) < \overline{b}_d \\ 0 & \text{otherwise} \end{cases} \\ OFS(\mathbf{p}) &= \sum_{\substack{\overline{\mathbf{b}}, \\ \text{for all } \overline{\mathbf{P}}}} \theta(\overline{\mathbf{b}}) \end{aligned} \tag{7}$$

Where the factor $\lambda > 1$ is introduced to deal with noise inherent in disparity computation. Since we expect the stereo-based range to be less robust for locations that are far from the camera, we can classify $\mathbf{p}$ as foreground if $OFS(\mathbf{p}) > T_{OFS}(p_d)$, where $T_{OFS}(d) \sim 1/d$.

2.2. Range Cluster Generation

The method described in the previous section provides us with a measure of how much *free space* is occluded by each pixel in a given view. We use this information to estimate the extent and connectivity of foreground regions in each view, and then link regions across views based on their projected overlap. Individual pixels are first clustered in each view, and we then determine whether two clusters belong to separate objects (Figure 2.2).

A naive approach would be to cluster the points based on proximity in either disparity or Euclidean space, and assume that each such cluster corresponds to a separate object. Such assumptions are correct in cases such as one in Figure 2.3(a), but lead to oversegmentation in the example in Figure 2.3(b), where components S^1 and S^2 actually correspond to parts of the same object. To resolve this ambiguity, we use the visibility information computed for each pixel (Section 2.1).

In Figure 2.2 we cannot separate O^1 and O^2 using only information from I^1 (S^1 and S^2). If the actual silhouettes of the objects in the second image were available, we could see that in fact they are non-overlapping, and thus objects are unconnected. In practice we do not have a set of complete silhouettes when range data is sparse. Instead, we compute an approximation of the silhouette in I^2 using the *free space* visibility constraints found for I^1. We define an "oversilhouette" to be the projection into I^2 of 3-D line segments formed by observed points P from I^1 and the first confirmed freespace point

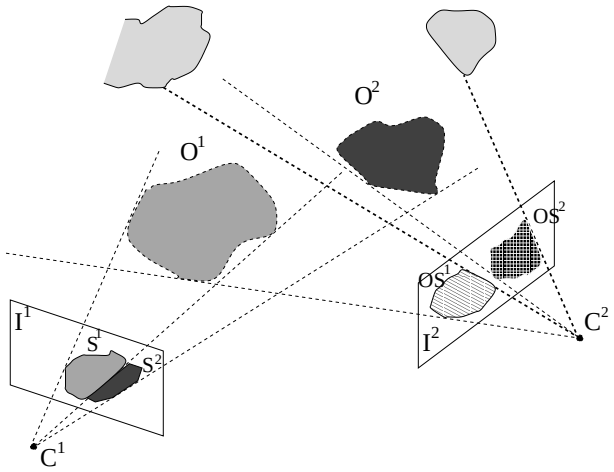


Figure 2.2: Segmenting multiple objects. The silhouettes S^1 and S^2 of objects O^1 and O^2 are adjacent in I^1, but the I^2 oversilhouettes, OS^1 and OS^2, which are computed from S^1 and S^2 (and the first freespace points found behind them) do not overlap, so we may conclude that O^1 and O^2 are indeed separate objects.

they occlude. If such "oversilhouettes" of two components do not overlap in I^2, then we conclude these components belong to different objects (Figure 2.2).

If the components' "oversilhouettes" overlap, we assume that components correspond to parts of the same object (Figure 2.3(b)). This can lead to undersegmentation (e.g. Figure 2.3(c)) if there are insufficient views to observe the segmentation between disjoint objects. With additional cameras this could be resolved as shown in Figure 2.3(d).

3. Virtual Background Generation

While the algorithm described in the previous section is capable of semi real-time performance (2fps on full resolution images) on current hardware, our tracking applications require much faster segmentation algorithms. The common range background subtraction algorithms provide high-speed performance, but are unreliable in the absence of the dense range data [4, 1]. While some improvement may be obtained using statistical training, the range images obtained in the indoor environment would generally be sparse (Figures 5.1(d), 5.2(d)). In this section we describe a method for generating dense virtual background images.

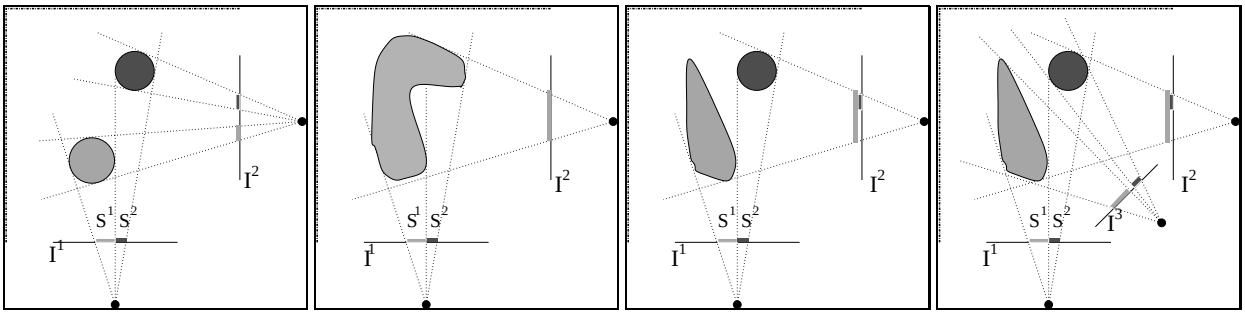
When the common range background subtraction methods are used, each pixel in the "background" image represents an upper limit on the depth (lower limit on the disparity) of *free space* visible along the corresponding optical ray when no foreground objects are present. Such upper limit may be obtained by, for example, taking the minimum of the observed valid disparity values at the pixel over time [4].

If no range data is available at the point, we can estimate this limit from visibility constraints obtained from "complimentary" cameras. For each point in I^i with invalid range we search the corresponding optical ray to detect all *free space* points along it that are visible by other cameras C^js, and select the one with the greatest depth as the virtual background.

In order to simplify the algorithm we inverse the order of computation. Instead of searching along the optical rays of C^i, we compute all *free space* points visible by C^j, $i \neq j$, using the computation described in the next section.

For each valid range point $\mathbf{p} = (p_x\ p_y\ I^j_D(p_x, p_y)\ 1)^T$, all points on the optical ray between $\mathbf{P} \simeq \mathbf{T}^{\mathbf{E^j}}_{\mathbf{D^j}}\mathbf{p}$ and $\mathbf{C^j}$ are transparent and may be assumed to belong to *free space*. Thus any point $\overline{\mathbf{P}} \in (\mathbf{P}, \mathbf{C^j})$ is a candidate virtual background for the corresponding point in I^i, $(\overline{b}_x, \overline{b}_y)$, such that

$$\begin{pmatrix} \overline{b}_x \\ \overline{b}_y \\ \overline{b}_d \\ 1 \end{pmatrix} = \overline{\mathbf{b}} \simeq (\mathbf{T}^{\mathbf{E^i}}_{\mathbf{D^i}})^{-1}\mathbf{T}^{\mathbf{E^i}}_{\mathbf{E^j}}\overline{\mathbf{P}} \tag{8}$$



(a) (b) (c) (d)

Figure 2.3: Connected components configurations. (a) the components S^1, and S^2 belong to separate objects, as the oversilhouettes computed from them in I^2 do not overlap. (b) the components belong to the same object. (c) The algorithm undersegments the scene, assuming that S^1 and S^2 belong to the same object, since their "oversilhouettes" overlap in I^2, but the same scene may be correctly segmented (d) if an extra view (I^3) is available.

After a set of candidates for a single (discrete) image location ($\{\tilde{\mathbf{b}}_i = (\tilde{b}_x\ \tilde{b}_y\ \tilde{b}_{d_i}\ 1)^T\}$) is computed, we select

$$v = \min_i \tilde{b}_{d_i} \tag{9}$$

as the virtual background value at the location $(\tilde{b}_x, \tilde{b}_y)$.

If both virtual and statistically trained background images are available, we may combine them to increase robustness by using data from statistically trained background when it is available (as it represents the true limit), and using virtual background data otherwise (Figure 5.5(a-f)).

4. Fast Computation of Visibility Constraints

In this section we detail our method for discretizing disparity-space projections of optical rays. Since this is the most time consuming part of the algorithms described in the previous sections, it is imperative that the point generation be extremely fast.

For each valid disparity point $\mathbf{p} = (p_x\ p_y\ I^i_D(p_x, p_y)\ 1)^T$, the corresponding optical ray in the camera C^i is $[\mathbf{C^i}, \mathbf{P})$, where $\mathbf{P} \simeq \mathbf{T}^{\mathbf{E^i}}_{\mathbf{D^i}}\mathbf{p}$. The image of this ray in D^j is $[\mathbf{\Gamma C^i}, \mathbf{\Gamma P})$, where $\mathbf{\Gamma}$ is the transformation from the E^i to D^j, i.e.

$$\mathbf{\Gamma} = (\mathbf{T}^{\mathbf{E^j}}_{\mathbf{D^j}})^{-1}\mathbf{T}^{\mathbf{E^j}}_{\mathbf{E^i}} = \left(\begin{array}{c|c} \mathbf{\Gamma}_{4\times 3} & \begin{matrix}\gamma_x\\ \gamma_y\\ \gamma_d\\ \gamma_w\end{matrix}\end{array}\right) \tag{10}$$

.

Since $\mathbf{C^i}$ is the origin of E^i (i.e $\mathbf{C^i}_{E^i} \simeq (0\ 0\ 0\ 1)^T$), the ray in D^j is $[\boldsymbol{\gamma}, \mathbf{b})$, where $(b_x\ b_y\ b_d\ 1)^T = \mathbf{b} \simeq \mathbf{\Gamma P}$.

We can represent any point $\overline{\mathbf{b}} = (\overline{b}_x\ \overline{b}_y\ \overline{b}_d)^T$ on this ray as $\overline{\mathbf{b}} = (b_x\ b_y\ b_d)^T + t\mathbf{u}$. Where

$$\begin{aligned} \mathbf{u}' &= \begin{pmatrix} b_x\gamma_w - \gamma_x \\ b_y\gamma_w - \gamma_y \\ b_d\gamma_w - \gamma_d \end{pmatrix} \\ \mathbf{u} &= \pm\frac{\mathbf{u}'}{\max\{|u'_x|, |u'_y|\}} \end{aligned} \tag{11}$$

with sign selected so that $\mathbf{u}$ points "away" from $\boldsymbol{\gamma}$ when applied at $\mathbf{b}$. In this case the positive values of t correspond to the direction away from the center of projection, and should be used for foreground segmentation (Section 2), while the negative ts should be used in generating virtual backgrounds (Section 3).

The region where the points on the ray are valid is determined by the extend of available portion of I^{j}_{D} (the range image), and the requirements that the point $\mathbf{P}$ should be in front of both image planes I^{i} and I^{j}. Applying these constraints is similar to line clipping, commonly used in computer graphics [6]. As the result of clipping we obtain values t_{MIN} and t_{MAX}, such that if $t_{MIN} \leq t \leq t_{MAX}$, then $\overline{\mathbf{b}} = (b_x\ b_y\ b_d)^T + t\mathbf{u}$ is valid.

After $\mathbf{b}$, $\mathbf{u}$, t_{MIN} and t_{MAX} are computed, we use Bresenham's algorithm [2] to discretize the ray relative to I^{j}, i.e. compute all points $\overline{\mathbf{b}} = \mathbf{b} + t\mathbf{u}, t \in \Im, t_{MIN} \leq t \leq t_{MAX}$. These point can then processed as described in Sections 2 and 3.

Note that while ray processing is the most time consuming operation of the presented algorithms, each ray is independent of all other rays, and the algorithm can take advantage of parallel processing capabilities if they are available.

5. Experimental Results

The foreground segmentation algorithm as implemented currently consists of several parts. The first part performs per-pixel computation described in Section 2.1 with $\lambda = 1.05$. We then cluster the pixels using techniques from Section 2.2, and finally pass the connected components through a size filter (accepting components greater than 1% of the image). The algorithm, running on 700MHz Pentium III, achieved the performance of 2 frames per second on the full resolution images.

To test our algorithms we used an installation with two Point Grey Digiclops cameras [12]. One camera used 6mm lenses, and another had 3.8mm lenses (wide-angle) (Figures 5.1 and 5.2), with approximately perpendicular viewing directions. The cameras were calibrated offline. We used the Triclops SDK [12] to produce rectified reference and disparity images. High surface and texture validation thresholds were specified to produce much more reliable (although more sparse) disparity output.

In Figures 5.3(a, b) we show the results of nonconservative background subtraction (i.e. labeling new pixel as foreground if a valid range was detected where the background model is invalid) between range views of the same scene under different lighting conditions. Figures 5.3(c, d) show the results of conservative background subtraction (i.e. labeling new pixel as foreground only if background model contained valid value different from the new one) when trying to segment two people in the room. As can be seen neither method produces acceptable results under the conditions we expect the segmentation algorithm to handle. Nonconservative background subtraction produces large number of false positives when illumination changes, and contrast (and thus valid range data) become available on previously uniform background regions. A conservative approach, on the other hand, never detects foreground objects where no valid range data is available in the background model.

Figure 5.4 demonstrates the results of applying our foreground segmentation algorithm to the same data as in figures 5.3(c, d) . Note that the algorithm was able to correctly segment people where no background range data was available (cf. Figures 5.3(c) and 5.4(a)). Classifying parts of the table as foreground is, in fact, correct behavior of the algorithm, as there is empty space detectable behind them.

The output of the virtual background generation algorithm applied to the same data is shown in Figure 5.5. The dense pure virtual background images (a, b) were generated from the disparity images in Figures 5.1(d), 5.2(d). The background models (c, d) used in our conservative background subtraction, were obtained by combining direct observations and virtual range images. The resulting (unfiltered) segmentations are shown in (e, f).

6. Conclusions

We have presented two novel range-based segmentation algorithms, that take advantage of availability of multiple, widely spaced stereo views. The semi real-time foreground segmentation algorithm relies on the visibility information obtained from other views to locate points that occlude *free space*. Since the algorithm does not maintain an explicit background model and uses only immediately available reliable range information, it is able to handle variable lighting conditions. We further extended the algorithm to use visibility constraints to improve clustering of the object points. The virtual backgrounds algorithm uses the visibility information to create dense range background images which can then be used with common real-time conservative background subtraction methods.

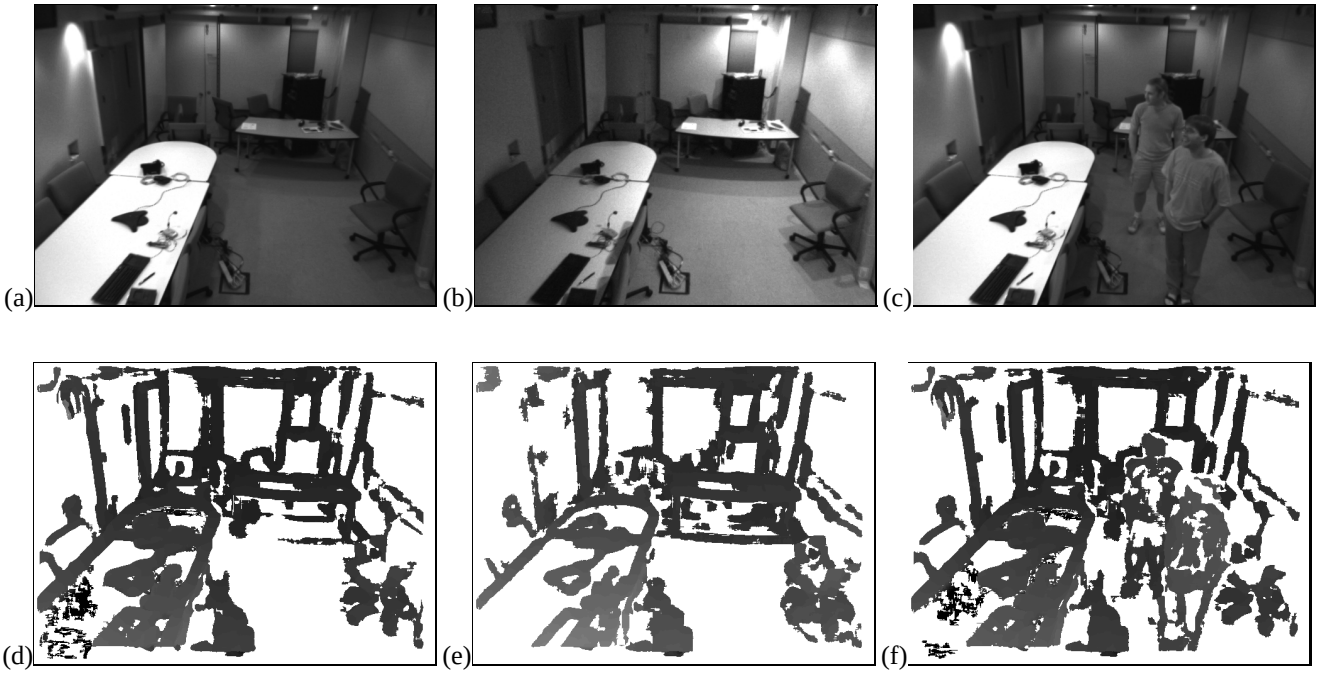


Figure 5.1: Intensity and disparity pairs obtained from camera C^1. (a, d) – empty room. (b, e) – empty room under different lighting conditions. (c, f) room with two people. The pixels with invalid disparity are shown in white.

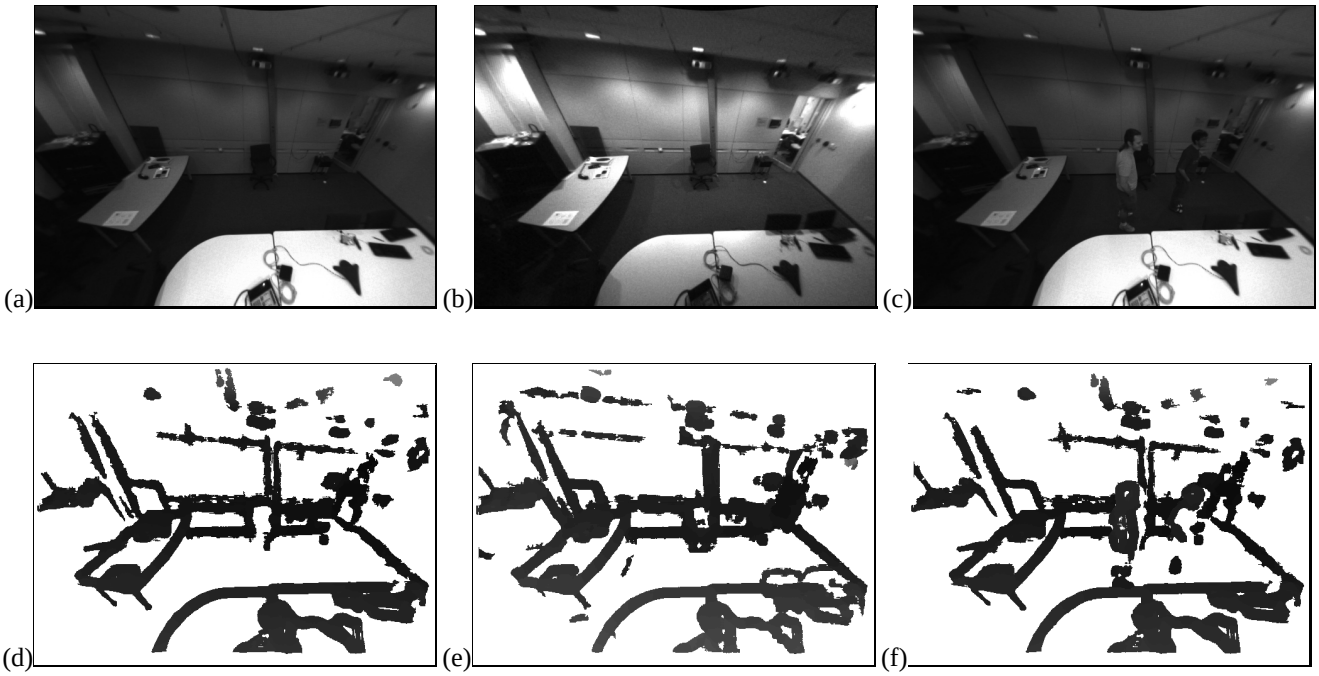


Figure 5.2: Intensity and disparity pairs obtained from camera C^2. (a, d) – empty room. (b, e) – empty room under different lighting conditions. (c, f) room with two people. The pixels with invalid disparity are shown in white.

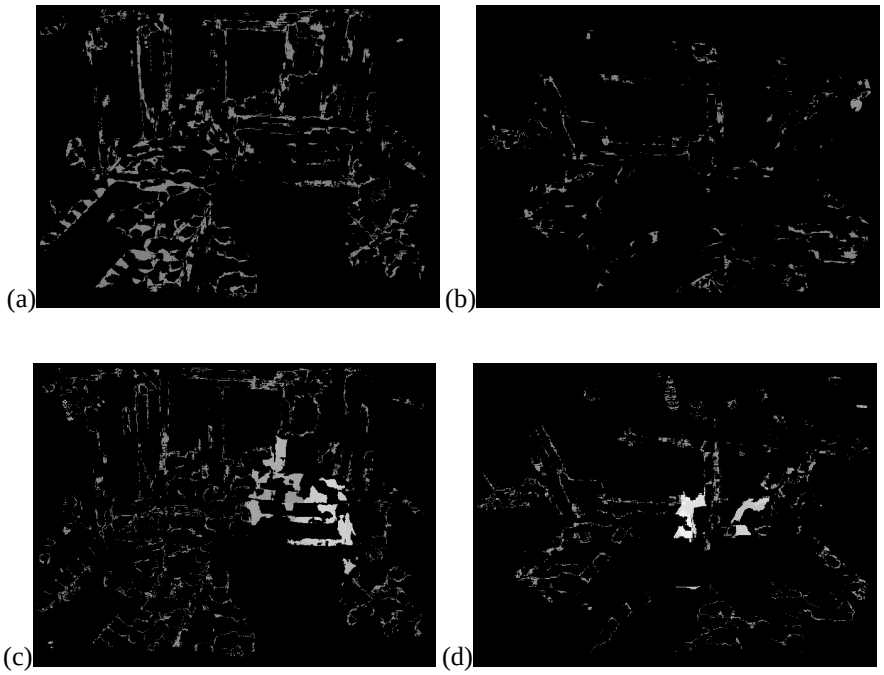


Figure 5.3: The problems with conventional background modeling with sparse range data (the difference values are scaled for display). (a, b) Non-conservative disparity background subtraction result for the views of the empty room under different illumination. This approach detects many false positives, when new valid range data becomes available in the background regions as illumination changes. (c, d) Conservative disparity background subtraction result between views of empty room and room with two people. This approach never detects foreground points when they appear in parts of the model with invalid range data.

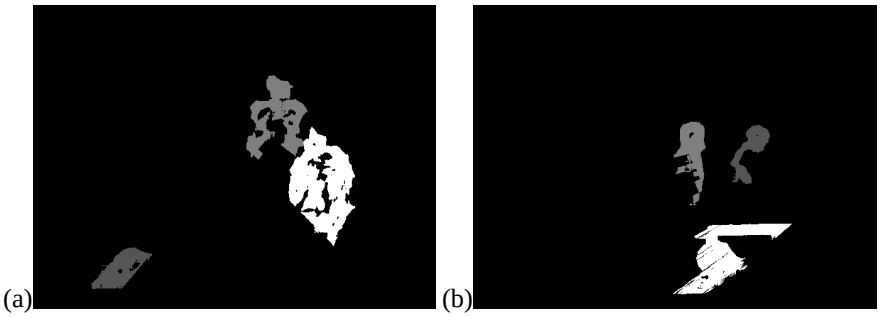


Figure 5.4: Results of applying foreground segmentation algorithm. Note that only data from Figures 5.1(f) and 5.2(f) – instantaneous range views is used. The connected components are shown in different colors. (a) C^1 view, and (b) C^2 view.

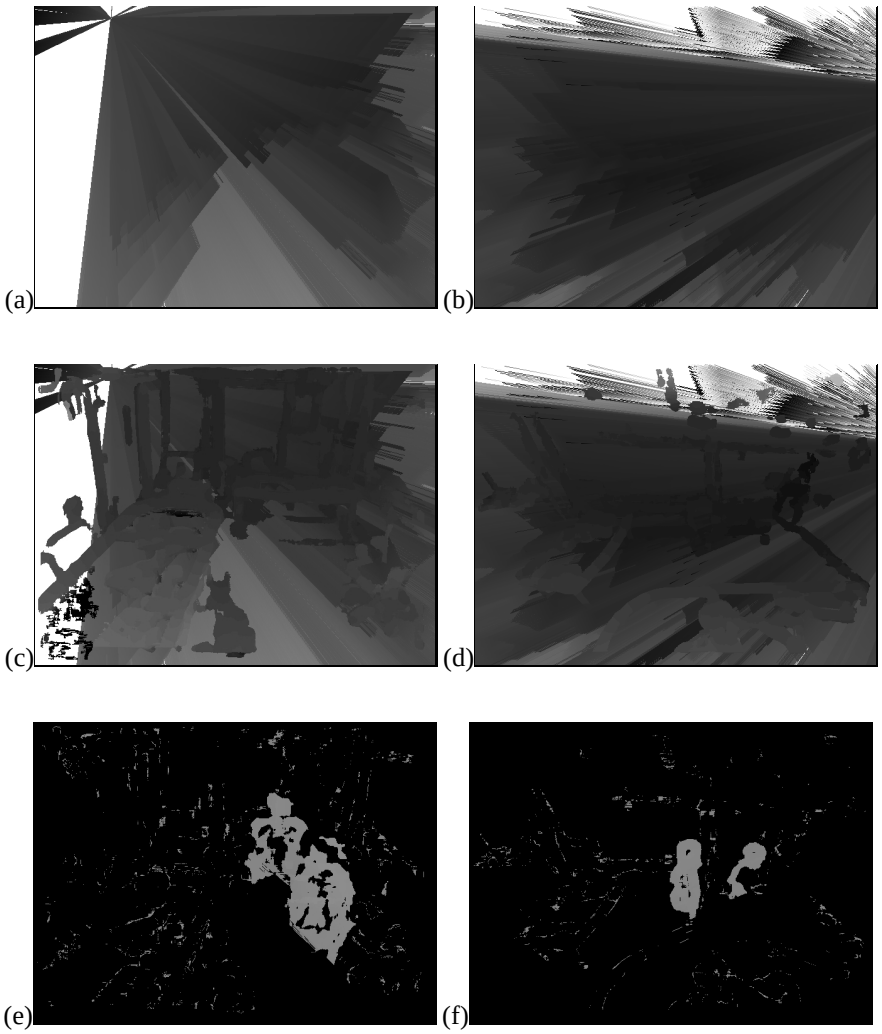


Figure 5.5: Results of applying virtual backgrounds algorithm. The background disparity images from Figures 5.1(d) and 5.2(d) were used to generate virtual backgrounds (a) and (b). (c) and (d) are the background images obtained by combination of direct observations and virtual backgrounds.The (unfiltered) results of applying conservative background subtraction to images 5.1(f) and 5.2(f) are shown in (e) and (f) respectively. (a), (c) and (e) correspond to C^1 view, and (b), (d) and (f) to C^2 view. Compare the results from our foreground segmentation algorithm (Figure 5.4), and conservative background subtraction results using background model obtained via direct observations (Figure 5.3(c, d)) and combined direct and virtual observations (e, f).

Bibliography

[1] D. Beymer and K. Konolige. Real-time tracking of multiple people using continous detection. In *Proc. International Conference on Computer Vision (ICCV'99)*, 1999.

[2] J. E. Bresenham. Algorithm for computer control of digital plotter. *IBM Syst. J.*, 4(1):25–30, 1965.

[3] B. Brumitt, B. Meyers, J. Krumm, A. Kern, and S. Shafer. Easyliving: Technologies for intelligent environments. In *Proceedings of Second International Symposium on Handheld and Ubiquitous Computing, HUC 2000*, pages 12–29, September 2000.

[4] T. Darrell, D. Demirdjian, N. Checka, and P. Felzenszwalb. Plan-view trajectory estimation with dense stereo background models. In *Proc. International Conference on Computer Vision (ICCV'01)*, 2001.

[5] D. Demirdjian and T. Darrell. Motion estimation from disparity images. In *Proc. International Conference on Computer Vision (ICCV'01)*, 2001.

[6] J. D. Foley, A. van Dam, S. K. Feiner, J. F. Hughes, and Phillips. *Introduction to Computer Graphics*. Addison-Wesley, Reading, MA, 1993.

[7] W.E.L Grimson, C. Stauffer, R. Romano, and L. Lee. Adaptive background mixture models for real-time tracking. In *Proceedings of CVPR'98*, 1998.

[8] M. Harville, G. Gordon, and J. Woodfill. Foreground segmentation using adaptive mixture models in color and depth. In *Workshop on Detection and Recognition of Events in Video.*, 2001.

[9] J. Krumm, S. Harris, B. Meyers, B. Brumitt, M. Hale, and S. Shafer. Multi-camera multi-person tracking for easyliving. In *IEEE Workshop on Visual Surveillance*, July 2000.

[10] K. N. Kutulakos and S. M. Seitz. A theory of shape by space carving. *Int. Journal of Computer Vision*, 38(3):199–218, July 2000.

[11] Wojciech Matusik, Chris Buehler, Ramesh Raskar, Steven J. Gortler, and Leonard McMillan. Image-based visual hulls. In Kurt Akeley, editor, *Siggraph 2000, Computer Graphics Proceedings*, Annual Conference Series, pages 369–374. ACM Press / ACM SIGGRAPH / Addison Wesley Longman, 2000.

[12] B.C. Point Grey Research, Vancouver. http://www.ptgrey.com/.

[13] G. Slabaugh, B. Culbertson, and T. Malzhender. A survey of methods for volumetric schene reconstruction from photographs. In *VG'01*, 2001.

[14] C. Stauffer and W.E.L Grimson. Adaptive background mixture models for real-time tracking. In *Proceedings of CVPR'99*, 1999.

[15] K. Toyama, J. Krumm, B. Brumitt, and B. Meyes. Wallflower: Principles and practice of background maintenance. In *In Proc. International Conference on Computer Vision (ICCV'99)*, 1999.

[16] C. Wren, A. Azarbayejani, T. Darrell, and A. Pentland. Pfinder: Real-time tracking of the human body. In *Photonics East, SPIE, volume 2615*, 1995.