Semi-Dense Stereo Correspondence with Dense Features

Olga Veksler
NEC Research Institute, 4 Independence Way Princeton, NJ 08540
olga@research.nj.nec.com

Abstract

We present a new feature based algorithm for stereo correspondence. Most of the previous feature based methods match sparse features like edge pixels, producing only sparse disparity maps. Our algorithm detects and matches dense features between the left and right images of a stereo pair, producing a semi-dense disparity map. Our dense feature is defined with respect to both images of a stereo pair, and it is computed during the stereo matching process, not a preprocessing step. In essence, a dense feature is a connected set of pixels in the left image and a corresponding set of pixels in the right image such that the intensity edges on the boundary of these sets are stronger than their matching error (which is basically the difference in intensities between corresponding boundary pixels). Our algorithm produces accurate semi-dense disparity maps, leaving featureless regions in the scene unmatched. It is robust, requires little parameter tuning, can handle brightness differences between images, and is fast (linear complexity).

1 Introduction

Stereo correspondence is one of the oldest problems in computer vision, with numerous applications. Despite significant progress through the years, the accuracy and reliability of the existing stereo algorithms can be improved. Any stereo algorithm must face the following: while textured regions in a scene are relatively easy to match (although not in the case of repeated texture), textureless regions are hard to deal with. Many different approaches were developed to address this problem.

The feature-based approaches [15, 8, 16, 18, 20] detect and match only "feature" pixels. These are the physically significant image pixels, such as pixels on an intensity edge. Feature pixels can be matched reliably. Textureless regions are left unmatched. The advantage of these methods is that they produce accurate results. The results are rather sparse, though, which is a disadvantage. Many applications require dense measurements, and interpolation is a difficult problem in itself.

Since sparse depth maps are insufficient, many methods were developed to match all pixels, not just those in edge features. These methods must deal with homogeneous regions. The basic idea behind all of them is to encourage "cooperation" between pixels, so that pixels in homogeneous regions get assigned smoothly varying disparities. We roughly divided such methods in a few groups below.

Area correlation methods [17, 19, 7] assume that a pixel is surrounded by a window of pixels with the same disparity, and windows of pixels are matched. Cooperation is encouraged because close-by pixels are matched with only slightly different windows, and thus are likely to be assigned the same disparity. However choosing an appropriate window is a difficult problem, only a few researchers have addressed it [11, 5, 13]. Area correlation methods produce dense stereo maps, but can be quite unreliable not only in homogeneous regions, but also in textured regions for an inappropriately chosen window size.

Cooperative methods [14, 24] directly encourage nearby pixels to cooperate by local iterative schemes which propagate information from a pixel to its neighbors. Energy minimization methods [1, 6, 9, 3, 21] also directly promote cooperation but use global optimization. They design and minimize energy functions which reward smooth or almost smooth disparity maps. The cooperative and energy minimization methods frequently have parameters which are difficult to set, and they tend to be inefficient. Another drawback is that it may still be difficult to assess whether a homogeneous region was assigned the correct disparity, since in some cases all pixels in a homogeneous region may be assigned the same, but nevertheless wrong disparity.

In the last group are the segmentation based methods. Their underlying idea is to use the results of an image segmentation algorithm to locate regions which are likely to belong to the same object, and match those regions [4, 12, 23], The actual details vary significantly between these algorithms. Finding good regions to match through image segmentation is, of course, a difficult problem.

All of the above methods have certain disadvantages. We propose a new feature based approach to stereo correspondence. Most of the previous feature based algorithms match thin features (edge pixels), producing accurate but sparse results. We would like to retain the accuracy of the feature based methods, but detect and match dense features, thus producing semi-dense stereo maps. In essence, our dense feature is a connected set of pixels in the left image and the corresponding set of pixels in the right image with intensity boundaries stronger than the error of matching the left and right boundary pixels. We call this the "boundary" condition, and it is the main enabling idea of our algorithm: the intensity change on the boundary must be more significant than the noise level of the pixels being matched, otherwise the boundary does not carry any useful information, its significance is destroyed by noise. If this principle is not enforced, then any textured region could be matched to any other textured region. Unlike the previous feature based methods, the detection of our dense features is an integral part of the algorithm, not a preprocessing stage. Furthermore, the threshold to detect a feature is adaptive, it depends on the noise of pixels being matched.

To find a dense feature at disparity d, we first overlap the left and the right images at disparity d and compute the error surface, see Section 2.1. The error surface is basically the absolute difference in intensities between the pixels of the left and right image. We expect it to be small for pixels which are likely to correspond. Then we use the error surface to find the binary match surface which is 1 for pixels that may have disparity d, and 0 otherwise. Then we prune the match surface leaving only the regions satisfying the "boundary" condition, that is the regions with borders on the intensity boundaries larger than the error surface on that boundary. When pruning, we take into consideration the brightness changes between the left and the right images, see Section 2.3. The connected regions which survive the pruning are our dense features. Due to image structure some features may overlap. In the final stage of our algorithm, if a pixel belongs to several features (in case of overlapping features), we choose the "densest" feature for that pixel, see Section 2.4. Thus the algorithm has four main stages: computing the error surface, computing the match surface, detecting dense features in the match surface, and choosing among dense features when necessary.

Besides feature-based approaches, our algorithm is similar to segmentation based stereo. Our dense features can be thought of as the appropriate segments to match. However segmentation of the dense features is an integral part of our stereo algorithm, not a separate preprocessing stage.

Our algorithm has many good properties. Its complexity is linear in the number of pixels times the number of disparities searched, so it is very fast, taking 1 second for smaller images and 7 seconds for larger images. It is even more efficient in its memory usage, which is linear in the number of pixels. It produces accurate results as tested by real data with ground truth, see Section 3. It can handle brightness differences between the left and the right images. Even though it is feature based, a large percentage of pixels is matched, from 40 to 95 percent in our experiments. Occlusions do not need to be handled, since most of the occluded pixels do not belong to any dense feature. Our parameters have intuitive meaning, and we do not tune them separately for each stereo pair. The algorithm can handle not only homogeneous regions but also repeated texture regions. We do not need to produce a separate "uncertainty" map for the disparity map. Only the "certain" pixels, i.e. pixels belonging to some dense feature are assigned a disparity.

2 Description of the Algorithm

We assume that the images are rectified so that the epipolar lines are the scanlines. We search in the disparity range $\{0, \ldots, max_d\}$, where max_d is the maximum possible disparity, the only parameter provided by the user in our implementation. Right now we search with pixel precision, that is only integer disparity d are considered, although the algorithm is easily extended to search in the subpixel range.

The algorithm is organized as follows. We cycle through all $d \in \{0, \ldots, max_d\}$. For each d there are four main steps. First we overlap the left and the right images at disparity d, and compute the error surface, see Section 2.1. The second step is to compute the match surface, see Section 2.2. The third step is to find all the dense features $\{f_d^1, \ldots f_d^n\}$ in the match surface, see Section 2.3. The last step is to go through all $p \in f_d^i$ and assign disparity d to p if the disparity of p is still uninitialized, or if f_d^i is "denser" for pixel p than $f_{d'}^j$, where d' is the current disparity assigned to pixel p and $f_{d'}^j$ is the feature containing p at disparity d'. This final step is explained in Section 2.4. The summary of our algorithm is in Fig. 1.

2.1 First Stage: Computing the Error Surface

In this section we explain how we find the error surface. We denote the intensity of pixel p in the left image by $L(p)$ and the intensity of pixel p in the right image by $R(p)$. We denote the error surface at disparity d and pixel p by $E_d(p)$.

To compute the error surface, we need a similarity measure between pixel p in the left image and pixel $p - d$ in the right image, where $p - d$ is the pixel with coordinates of p shifted by d to the right. Roughly $E_d(p)$ should be $|L(p) - R(p - d)|$. However even in the absence of noise, this error measure is not accurate for pixels overlapping the surface with rapidly changing intensity when the pixel's true intensity is not integer. This happens because of image sampling artifacts, see [2] for more details. Computing disparity at subpixel accuracy helps to solve this problem, but as [2] points out, the additional computation time may not

```
for all pixels $p$ do
    $disparity(p) = NONE$
for $d = 0, \dots max_d$ do
    1. Compute the error surface $E_d(p)$
    2. Compute the match surface $M_d(p)$ from $E_d(p)$
    3. Find dense features $\{f_d^1, \dots f_d^n\}$ in $M_d(p)$
    4. for $i = 0, \dots n$
        for $p \in f_d^i$ do
            if $disparity(p) = NONE$
                disparity(p) = d
                $FeatureDensity(p) = density(f_d^i, p)$
            else if $density(f_d^i, p) > FeatureDensity(p)$
                disparity(p) = d
                $FeatureDensity(p) = density(f_d^i, p)$
```

Figure 1. Overview of the algorithm.

be worth it. Instead we use the method in [2] to construct the error surface insensitive to image sampling.

First we define $\hat{R}$ as the linearly interpolated function between the sample points on the right scanline, and then we measure how well the intensity at p in the left image fits into the linearly interpolated region surrounding pixel $p - d$ in the right image

$$e_d^l(p) = \min_{q \in \left[p-d-\frac{1}{2}, p-d+\frac{1}{2}\right]} |L(p) - \hat{R}(q)|.$$

For symmetry,

$$e_d^r(p) = \min_{q \in \left[p-\frac{1}{2}, p+\frac{1}{2}\right]} |\hat{L}(q) - R(p-d)|.$$

Thus our error surface is the symmetric measure of similarity between pixels p in the left image and pixel $p - d$ in the right image:

$$E_d(p) = \min\left\{e_d^l(p), e_d^r(p)\right\}.$$

2.2 Second Stage:Computing the Match Surface

With the error surface defined, we are ready to compute the binary match surface. We denote this surface for disparity d and pixel p by $M_d(p)$. At this stage, we want to set $M_d(p) = 1$ for pixels for which disparity d might be the right disparity, and set $M_d(p) = 0$ for the rest.

Left and right images of a stereo pair sometimes have significant brightness differences, due to different camera gains or changed light conditions, for example. We want to allow pixels with significant brightness difference to match, provided that nearby pixels experience similar brightness differences. At the same time we need to exclude the unlikely matches. To satisfy these two goals simultaneously,

```
for all $p$
    $M_d(p) = 0$
Sort $p$ in the order of increasing $E_d(p)$
for all $p$ in order of increasing $E_d(p)$ do
    if $M_d(q) = 0 \quad \forall q \in N_p$
        $M_d(p) = 1$
    else if $|E_d(q) - E_d(p)| < \epsilon \quad \forall q \in N_p$ s.t. $M_d(q) = 1$,
        $M_d(p) = 1$
```

Figure 2. Match surface computation.

we detect the regions in the error surface with smoothly varying errors, and set the match surface to 1 for those regions. This way we allow matching only between two regions differing by a smooth surface. For example matching between two regions with smoothly varying or constant intensities is allowed, even if these regions are of different brightness, as long as the difference surface is smooth. Matching a smoothly varying region with a textured region is not allowed, and matching between two regions with different textures is also not allowed.

Now we can explain why we detect dense features in the match surface rather than in the error surface. Recall that our "boundary" condition for dense features says that the intensity change on the boundary must be greater than the error on the boundary. Notice that this definition treats the boundary pixels differently from others, enabling efficient detection of dense features. However if we use boundary condition on the error surface, then we are checking only that the boundary pixels are good matches. If due to image structure and noise we find some false "good" boundary in the error surface, all the pixels inside are automatically matched, even if the inside pixels form two unrelated textures. This does not happen when we detect dense features in the match surface, since the match surface contains only the pixels which we consider a good match, as explained in the previous paragraph.

The algorithm to compute the match surface is in Fig. 2. We start by initializing the match surface to 0. Then we sort all pixels in order of increasing $E_d(p)$. It can be done in linear time since the range of $E_d(p)$ is small. The next step is to go over all pixels p in order of increasing $E_d(p)$ and set $M_d(p) = 1$ if either of two conditions hold. First condition is that $M_d(p) = 0$ for all nearest neighbors q of p, where the nearest neighbors of p are just the pixels above, to the left, to the right, and below p. We denote the nearest neighbors of p by N_p. The second condition is that if $M_d(q) = 1$ for some $q \in N_p$, then $|E_d(p) - E_d(q)| \le \epsilon$. The first condition initializes some "seeds" from which to grow the match surface. It makes sense to take the pixels with the smaller errors for the seeds, that is why we sort $E_d(p)$'s.

```
for all $p$ s.t. $M_d(p) = 1$ and $M_d(p-1) = 0$
  while $|E_d(p) - avr(p,d)| + \sigma > |L(p) - L(p-1)|$ or
      $|E_d(p) - avr(p,d)| + \sigma > |R(p-d) - R(p-d-1)|$
      do
        $M_d(p) = 0$
        $p = p + 1$
```

Figure 3. Pruning the left boundary.

The second condition makes sure that all the regions which are set to 1 in the match surface correspond to smoothly varying error surface.

We set $\epsilon = 3$ for all the experiments. This value might seem rather small, but keep in mind that this is not the largest matching error that we allow, rather it is the largest difference between matching errors that we allow. Thus the two matched regions can differ significantly in intensity, but this difference must be smooth. In textured regions two neighboring pixels may have significantly different errors not explained by just the smooth brightness differences between the regions. However in textured regions $E_d(p)$ is smaller than the absolute difference of intensities due to the similarity measure we use, recall section 2.1. So this value of ϵ works well even in the textured regions.

In practice, of course, there are always a few pixels which do not obey our assumptions. To deal with these pixels, we compute connected components in the matching surface and patch all holes of small size, which we set to 5 in our implementation. The reason for 5 is the following: if some pixel has an error not handled by our assumptions, it can lead to a wrong choice for its 4 nearest neighbors.

2.3 Third Stage: Detecting Dense Features

In this section we explain how we find dense features in the match surface. Recall that to locate a dense feature, we need to find a region in the match surface which satisfies our "boundary" condition, i.e. the boundaries must be on intensity edges larger than error surface. This implies that we can match only regions where change in disparity occurs together with a change in intensity. In reality, of course, there are frequently uniform surfaces which straddle several disparities. To deal with such surfaces, we need to detect features which straddle several disparities. We plan to do so in the future, see section 4. However in the current implementation, we can deal with some of such uniform surfaces by enforcing our "boundary" condition only on the left and right boundaries of a region. We do not enforce it on the top or the bottom boundary. Thus we can at least match low-texture surfaces sloping horizontally, like the ground plane.

We begin by pruning pixels on the left boundary of the match surface until the the error of pixels on that boundary is smaller than the intensity edge on that boundary in the left and the right image. However we also want to correct for the brightness differences between the images. Therefore we subtract the average brightness difference around pixel p in the left image and pixel $p-d$ in the right image from $E_d(p)$. We denote this difference by $avr(p,d)$, and it is computed in the 3 by 3 window, i.e. if W_p is the 3 by 3 window around p, then

$$avr(p,d) = \sum_{q \in W_p} \left(L(q) - R(q-d)\right).$$

The algorithm to prune the left boundary is given in Fig. 3. We begin by taking some pixel p which does not satisfy our boundary condition. That is p is in the match surface, pixel to the left of p is not in the match surface, and $|E_d(p) - avr(p,d)| + \sigma$ is larger than the intensity boundary between p and $p-1$ in the left image or $|E_d(p) - avr(p,d)| + \sigma$ is larger than the intensity boundary between $p-d$ and $p-d-1$ in the right image. We remove p from the match surface, i.e. set $M_d(p) = 0$, and continue this process until all left boundary pixels p satisfy the "boundary" condition. We set $\sigma = 5$ for all our experiments, and this is the second and last significant parameter of our algorithm. It is used to make sure that the intensity on the boundary is not only larger, but significantly larger (by σ) than the error on the boundary.

We do similar pruning for the right boundary. Since we treat each scanline independently for the dense feature construction, there may be some inconsistencies between the the horizontal intervals of our dense feature. That is a few horizontal lines may stick in and out our dense feature. In principle, a better way to extract a dense feature from the match surface would be to use some boundary extraction algorithm, for example the one in [10]. However even our simple algorithm works very well with the following filtering step. If pixel p is in a dense feature f_d, but the pixels above and below p are not in f_d, then p is also removed from f_d. Also if pixel p is not in f_d, but pixels above and below p are in f_d then p is also placed in f_d.

After the filtering step, we find connected components in the pruned match surface, and remove components less than some minimum size, which we set to 25. The rest of the connected components are our dense features.

Figs. 4(a,b) show the match surface and the dense features at disparity 14 for the scene in Fig. 7(a). This is the correct disparity of the lamp. Pixels for which the match surface is 1 and pixels which belong to a dense feature are shown with bright intensity. Notice that the match surface is 1 for the majority pixels in the scene. However the match surface is 0 for the majority of the intensity edges for which the correct disparity other than 14. That is why when the

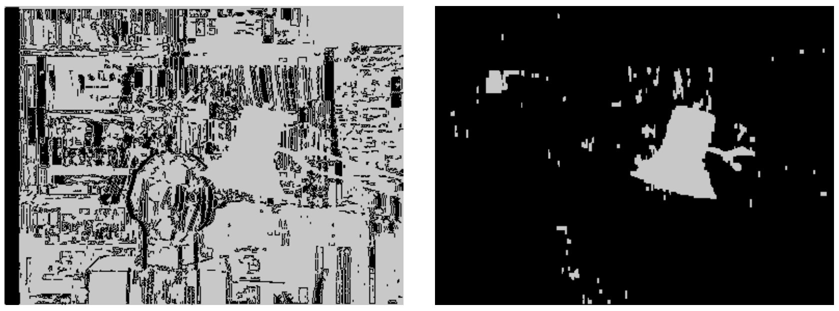

(a) match surface at $d = 14$ (b) dense features at $d = 14$

Figure 4. Comparison of match surface and dense features

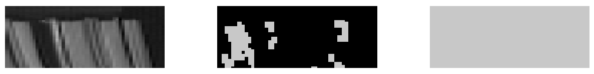

(a) left image (b) wrong d (c) right d

Figure 5. Overlap due to repeated texture

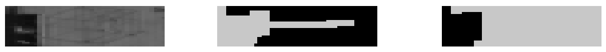

(a) left image (b) can disparity (c) wall disparity

Figure 6. Overlap due to spurious texture

match surface is pruned using our "boundary" condition, most of the pixels are not in any dense feature. The remaining regions correspond to the lamp, few regions of repeated texture, and few spurious small regions. Unfortunately, we loose most of lamp handle because the intensity edge there is not strong enough.

2.4 Forth Stage: Choosing Dense Features

Some pixel p can be a part of two dense features. One reason is repeated texture. Consider Fig. 5(a). This is a cut out of a repeated texture region from the scene in Fig. 7(a). This region are the books right above the lamp. Figs. 5(b,c) shows dense features for this region at the wrong and the right disparities, respectively. Pixels which belong to a dense feature are shown in bright color, pixels which do not belong to any dense feature are shown in black color. Notice that for the right disparity, all the pixels are in a dense feature. Thus the feature at the right disparity is "denser" for the pixels which have a choice of features.

However most often features overlap is due not to repeated texture. It happens because some dense feature is joined by a few extra regions, usually fairly small and textured. Consider Fig. 6(a). It shows a region occupied by a corner of the soda can to the right and slightly above the lamp in Fig. 7(a). The background has texture due to the wall poster. Fig. 6(b) shows the dense feature at the disparity of the soda can. The can grabbed a thin horizontal region due to the wall texture. Fig. 6(c) shows dense feature at the correct disparity for that thin wall region. Notice that the correct disparity is "denser" for that region.

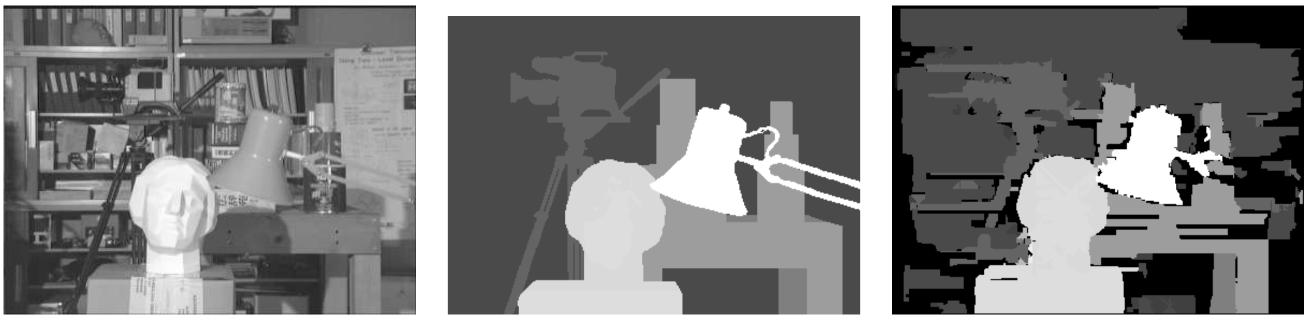
Now we will formalize what we mean by "denser". We need to estimate how many pixels are there in the immediate surrounding of p in a dense feature. Let M_d be our match surface, and let $H_d(p)$ be the length of the horizontal interval consisting only of pixels in the same dense feature and containing p. For example if there are l consecutive pixels to the left of p in the same dense feature as p and r pixels to the right of p in the same dense feature as p, then $H_d(p) = l + r + 1$. Similarly we define $V_d(p)$, $D^1_d(p)$, $D^2_d(p)$, where these quantities are, respectively, the number of pixels in the vertical and and two diagonal intervals containing p and consisting of consecutive pixels of the same dense feature that p is in. We can compute $H_d(p)$, $V_d(p)$, $D^1_d(p)$, $D^2_d(p)$ for all p in just four passes over the dense features. If f^i_d is a feature at disparity d containing p, then

$$density(p, f^i_d) = H_d(p) + V_d(p) + D^1_d(p) + D^2_d(p) - \\ - \max\left\{H_d(p), V_d(p), D^1_d(p), D^2_d(p)\right\}$$

We subtract the maximum for robustness against occasional thin horizontal structure like in Fig. 6(b). Notice that $density(p, f^i_d)$ can be different from $density(q, f^i_d)$. With this definition of density, the regions in Figs. 5 and 6 are placed at the correct disparities, as can be checked in Fig. 7.

3 Experimental Results

In this section we present our experimental results on real stereo pairs, two of which have known ground truth.



(a) left image (b) ground truth (c) our algorithm

Figure 7. Real imagery with dense ground truth

For all the experiments, the parameters were fixed as follows: $\epsilon = 3$, $\sigma = 5$, and minimum feature size was set to 25. On the disparity maps, brighter pixels have larger disparity. Pixels for which no disparity was found (i.e. pixels which do not belong to any dense feature) are in black.

3.1 Head and Lamp Stereo Pair

Fig. 7(a) shows the left image of a stereo pair from the university of Tsukuba. For this stereo pair the dense ground truth is known, and it is in Fig. 7(b). The disparity map our algorithm computes is in Fig. 7(c). The size of these images is 384 by 288, maximum disparity we search is 14. The running time is 1 sec, and 75% of pixels are matched. The three largest regions which our algorithm leaves unmatched are the upper right corner, the lower right corner, and the upper part of the region under the table. Some parts of on the table are also not matched.

Out of the pixels that we match, 88.2% are found correctly, 98.5% are off by not more than 1 pixel, and only 1.5% of pixels are more than 1 disparity away from the correct answer. The absolute average error is 0.13. Algorithms in [24, 3] report 1.4% and 1.8% of pixels with error more than 1 disparity from the correct answer. However both of these methods tune their parameters to achieve the best performance, and they are much less efficient than our algorithm. The work in [22] gives a more comprehensive comparison of the performance algorithms on this stereo pair. However out of all algorithms evaluated in [22], algorithms [24, 3] achieve the best accuracy.

3.2 Two Planes Stereo Pair

Fig. 8(a) shows the left image of another stereo pair for which the dense ground truth is also known. This stereo pair is from Microsoft. Figs. 8(b,c) show the ground truth and our answer, respectively. The image sizes are 284 by 216, and the maximum disparity we search for is 28. The running time was 2 seconds, with 89% of pixels matched. Since the ground truth was computed for the right image, we also computed our answer for the right image. Notice that the black region on the right is not matched, this region corresponds to the occluded pixels. Only three small regions which should be occluded are matched erroneously. Out of the pixels which our algorithm matches, 84.6% are matched correctly, 99.1% are off by not more than one pixel from the correct disparity, and the average absolute error is 0.27.

3.3 Birch Stereo Pair

Fig. 9(a,b) shows the left and right birch tree images from SRI. The image size is 320 by 242, and the maximum disparity we searched for is 28. The running time was 2 seconds, with 41% of pixels matched. The right image is approximately 15% brighter than the left image. This difference is easily noticeable to the eyes, that is why we show both the left and the right images. In addition, the texture of the grass in the front part of the left image is almost all lost. Only the two bright spots in the very front and three bright spots further in the back retain texture, the majority of other grass pixels in the frontal half have intensity 0. This makes stereo correspondence very challenging. Our algorithm, however, successfully matches the trees and the five spots on the grass which have not lost texture. It does not match the grass which lost its texture because it cannot match the textureless regions in the left image to the textured ones in the right image. It is hard to notice in out displays, but the two trees in the front have smoothly varying disparity, the closer one changes disparity from 22 on the bottom to 26 on top, and the one to the left of it changes disparity from 20 to 21. If we enforced our “boundary” condition on the whole boundary, and not just on the left and the right boundary as we do now, we would not be able to get these results, due to the lack of sufficient horizontal texture on these trees.

3.4 Shoe Stereo Pair

Fig. 10(a) shows the left image of another challenging stereo pair from CMU. The size is 512 by 480, and the maximum disparity is 14. The running time was 7 seconds, and 95% of pixels are matched by our algorithm. This stereo pair is difficult because of the repeated texture floor. Our algorithm was able to place almost all of the floor at the disparity 9. A manual inspection of the left and right images suggests that two plausible disparities of the floor are 9 and 3. However the disparity of the shoe varies from 13 to 11, so disparity 3 would place the floor too far, the shoe would have to float over it. So the disparity of 9 our algorithm produces is most likely right.

4 Discussion

Our results on the real stereo data, including the data with ground truth, show that our algorithm produces accurate results, can handle brightness differences between images, repeated texture, homogeneous image regions. It is robust with respect to parameters, they do not need to be tuned. It is also very fast and efficient with memory.

The biggest current limitation is that it cannot handle homogeneous sloped surfaces with slant other than horizontal. We plan to address this in our future work but allowing dense features to form across different disparities. When we do that, we will need to check our “boundary” condition across the whole border, not just the left and right boundary as we do right now. With this extension we also plan to apply our algorithm to motion sequences.

The second biggest limitation is the way we extract the dense features from the match surface. Instead of processing each scanline independently as we do now, it would be better to use a boundary extraction algorithm which is less local.

Acknowledgments

We would like to thank Prof. Y. Ohta from the University of Tsukuba and Dr. R. Szeliski from Microsoft research for providing the images with ground truth.

References

[1] S. Barnard. Stochastic stereo matching over scale. *IJCV*, 3(1. May 1989):17–32, May 1989.

[2] S. Birchfield and C. Tomasi. A pixel dissimilarity measure that is insensitive to image sampling. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(4):401–406, April 1998.

[3] Y. Boykov, O. Veksler, and R. Zabih. Fast approximate energy minimization via graph cuts. In *International Conference on Computer Vision*, pages 377–384, 1999.

[4] L. Cohen, L. Vinet, P. Sander, and A. Gagalowicz. Hierarchical region based stereo matching. In *CVPR89*, pages 416–421, 1989.

[5] A. Fusiello and V. Roberto. Efficient stereo with multiple windowing. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 858–863, 1997.

[6] D. Geiger, B. Ladendorf, and A. Yuille. Occlusions and binocular stereo. *International Journal of Computer Vision*, 14:211–226, 1995.

[7] D. Gennery. Modelling the environment of an exploring vehicle by means of stereo vision. In *Ph. D.*, 1980.

[8] W. Grimson. A computer implementation of a theory of human stereo vision. *Royal*, B-292:217–253, 1981.

[9] H. Ishikawa and D. Geiger. Occlusions, discontinuities, and epipolar lines in stereo. In *ECCV98*, pages xx–yy, 1998.

[10] I. Jermyn and H. Ishikawa. Globally optimal regions and boundaries. In *ICCV99*, pages 904–910, 1999.

[11] T. Kanade and M. Okutomi. A stereo matching algorithm with an adaptive window: Theory and experiment. *PAMI*, 16(9):920–932, September 1994.

[12] J. Ma and N. Ahuja. Region correspondence by global configuration matching and progressive delaunay triangulation. In *CVPR00*, pages II:637–642, 2000.

[13] R. Maas, B. ter Haar Romeny, and M. Viergever. Area-based computation of stereo disparity with model-based window size selection. In *CVPR99*, pages I:106–112, 1999.

[14] D. Marr and T. Poggio. Cooperative computation of stereo disparity. *Science*, 194(4262):283–287, October 15, 1976, October 1976.

[15] D. Marr and T. Poggio. A computational theory of human stereo vision. *RoyalP*, B-204:301–328, 1979.

[16] G. Medioni and R. Nevatia. Segment-based stereo matching. *CVGIP*, 31(1):2–18, July 1985.

[17] K. Mori, M. Kidode, and H. Asada. An iterative prediction and correction method for automatic stereocomparison. *CGIP*, 2:393–401, 1973.

[18] N.Ayache and B. Faverjon. Efficient registration of stereo images by matching graph descriptions of edge segments. *International Journal of Computer Vision*, 1, 1987.

[19] D. Panton. A flexible approach to digital stereo mapping. *PhEngRS*, 44(12):1499–1512, December 1978.

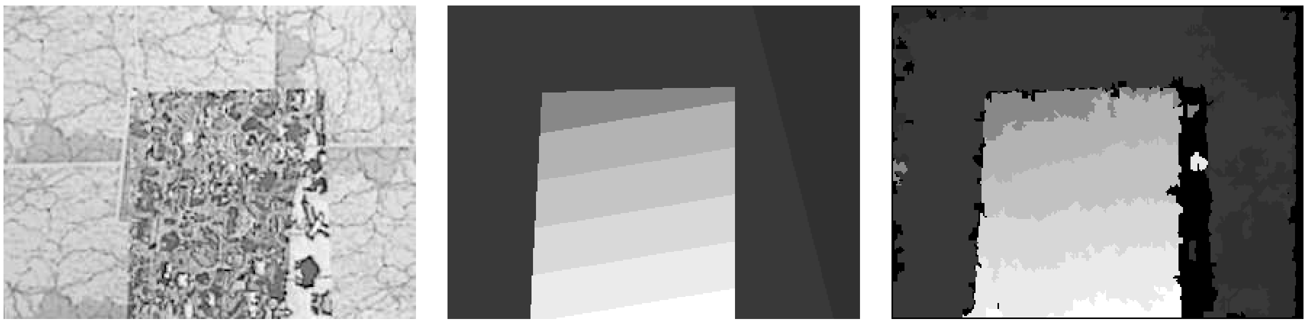
[20] L. Robert and O. Faugeras. Curve-based stereo: Figural continuity and curvature. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 57–62, 1991.

[21] S. Roy. Stereo without epipolar lines: A maximum-flow formulation. *IJCV*, 34(2/3):1–15, August 1999.

[22] R. Szeliski and R. Zabih. An experimental comparison of stereo algorithms. In *IEEE Workshop on Vision Algorithms*, September 1999.

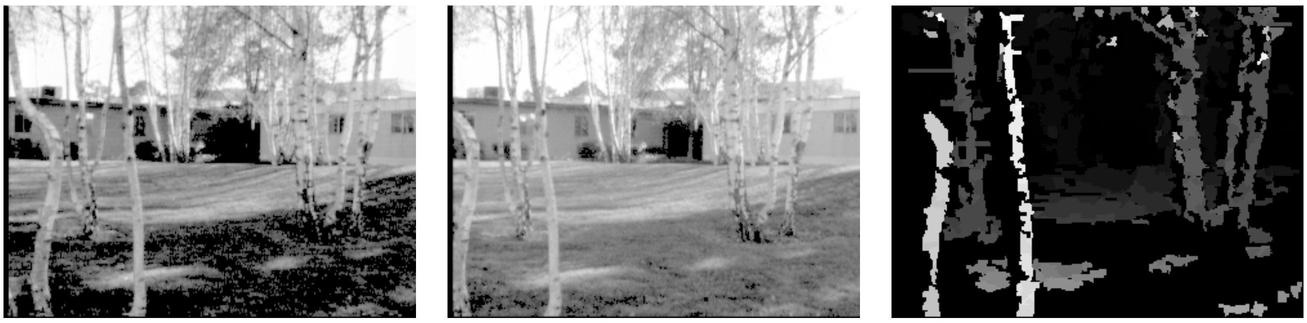
[23] H. Tao and H. Sawhney. Global matching criterion and color segmentation based stereo. In *WACV00*, pages 246–253, 2000.

[24] C. Zitnick and T. Kanade. A cooperative algorithm for stereo matching and occlusion detection. *PAMI*, 22(7):675–684, July 2000.



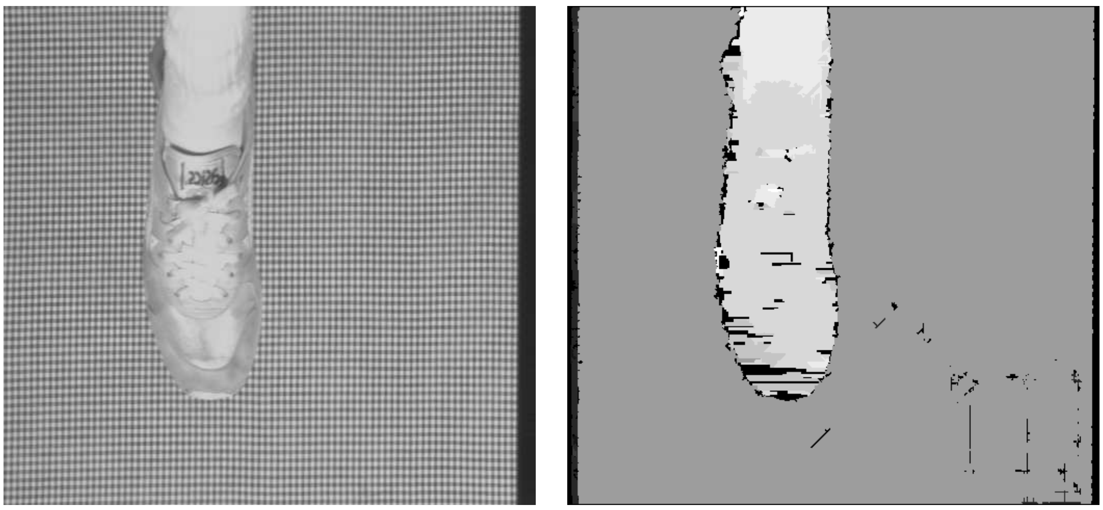
(a) left image (b) ground truth (c) our algorithm

Figure 8. Slanted planes stereo pair



(a) left image (b) right image (c) our algorithm

Figure 9. Birch Sequence from SRI



(a) left image (b) our algorithm

Figure 10. Shoe sequence from CMU