Dense Features for Semi-Dense Stereo Correspondence

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 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, nonlinear errors, and is fast (linear complexity).

1 Introduction

Stereo correspondence is one of the oldest problems in computer vision, with numerous applications. Due to noise and image structure, establishing correspondence is an ambiguous task. The very first designers of stereo algorithms recognized this fact. A way to disambiguate the problem is to make additional assumptions about the data. One natural assumption made implicitly or explicitly by most stereo algorithms is that disparity varies smoothly almost everywhere except the object boundaries. This assumption leads to a variety of quite distinct algorithms but with the following common principle. The disparity a pixel gets assigned should be influenced by the neighbors of that pixel, that is image pixels should cooperate in the stereo computation.

Stereo algorithms differ not only in the way they encode cooperation among pixels, but also in how much of the image data is used. When designing a stereo algorithm, one soon discovers that the textured regions in a scene are relatively easy to match (although not in the case of repeated texture), while textureless regions are hard to match accurately. Stereo algorithms roughly break down into two groups, depending on whether they match textureless regions or not.

The first approach is generally called feature based. See [16, 9, 17, 19, 22] for some examples. In this approach, only the "feature" pixels are detected and matched. These are the physically significant image pixels, such as intensity edges or corners. Textureless regions are left unmatched. The motivation is that the pixels in textureless regions cannot be matched reliably anyway. The advantage of the feature based methods is that they produce accurate results. The results are rather sparse, though. Many applications require dense measurements, and measurement interpolation is a difficult problem in itself. Feature based methods were especially popular in the early days of computer vision because image quality was generally poor and so comparing the raw image intensities of pixels was not a reliable measure for the likelihood of their correspondence.

The second approach is to match all (or almost all) image pixels to produce dense disparity estimates. The intuition is that by propagating disparity estimates from the high texture areas, the disparity in low texture areas can be inferred. The requirement of dense disparity estimates in many applications and the improvements in image quality lead to a greater popularity of the dense stereo algorithms in the recent years. We briefly summarize dense stereo algorithms in section 2. For a comprehensive overview and comparison of dense stereo correspondence methods, see [25, 24].

The motivation behind the sparse stereo methods is that only the pixels in the textured image areas can be matched reliably. The motivation behind the dense stereo methods is that all or almost all image pixels can be matched (some dense algorithms attempt to explicitly

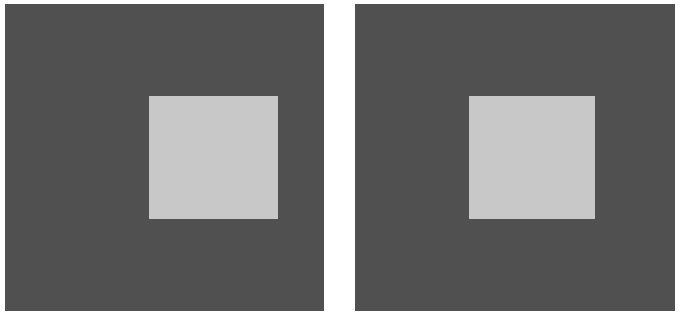


Figure 1. An artificial stereo pair. Textureless square in front of textureless background. Disparity of the square is easy to determine, while the disparity of the background is ambiguous.

detect and leave unmatched the occluded image pixels). The reality might be a mixture of the two assumptions. That is the disparity of textureless regions can be recovered in some cases, but not the others. Consider an artificial example in Figure 1. This stereo pair shows a bright textureless rectangle on a dark textureless background. The square shifts by several pixels to the left between the left and the right images of this stereo pair. Even though the square completely lacks texture, its disparity is easy to determine. The edges of the square give us a clear cue for matching. However most people would agree that the disparity of the background cannot be determined with any certainty.

We propose a new approach to stereo correspondence. The view that we will hold is that not all regions of a stereo pair can be matched reliably. Thus our aim is to detect only those regions of a stereo pair which are easy to match accurately. We will call such regions dense features. For example in Figure 1 the central square in both images should be a good candidate for a dense feature. 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 the difference in intensities between corresponding boundary pixels). We give our motivation behind this informal definition of dense features in Section 3. For now notice that our dense features are defined with respect to both images of the stereo pair overlapped at some disparity. Thus each dense feature has some disparity attached.

After all dense features are computed, pixels that belong to some dense feature get assigned the disparity of that dense feature. We only need to disambiguate disparity assignment for pixels which belong to more than one dense feature. Due to descriptiveness of the dense features, there is not a lot of ambiguity. Purely local, and therefore efficient disambiguating step works well enough. Each pixel which happens to be in more than one dense feature chooses the disparity of the feature which is "densest" with respect to that pixel. This procedure is described in Section 4.4.

Our approach is only superficially similar to the previous feature based methods. Most of the previous feature based algorithms match thin structures like edge pixels or corners, producing only sparse results. In addition, matching with thin features still presents a lot of ambiguities. Our dense features have several advantages over the previously used features. The first obvious advantage is that the disparity maps we produce are semi-dense, that is we match more pixels. The second advantage is that our dense features help to reduce ambiguities in correspondence because their structure is more descriptive. Other advantages are that 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 how noisy are the pixels being matched.

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 dense features is an integral part of our stereo algorithm, not a separate preprocessing stage.

Another algorithm we share some similarities with is the variable window algorithm [3]. In [3] a set of connected components or "windows" is computed at each disparity. These connected components contain only the pixels for which that disparity is likely. The disparity which gives the largest window is assigned to each pixel. Our dense features can also be looked at as windows. However [3] is not robust, its windows can grow to include many pixels which do not in fact belong to the

disparity of the window. The boundary condition that we enforce helps to avoid this problem in our dense features.

Some dense stereo algorithms [12, 30] compute a confidence or certainty map. This maps assigns to each pixel the confidence in its disparity estimate. Thresholding disparity estimates at some confidence level results in semi-dense disparity map which is presumably more accurate. It is important to compare our semi-dense algorithm with such thresholded disparity maps, which we do in section 5.2. However a confidence map is usually a byproduct of the correspondence algorithm, and we found that thresholding it does not necessarily improves the results significantly.

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 5. It can handle brightness differences and monotonic errors in intensities between the regions being matched. Even though it is feature based, a large percentage of pixels is matched, from 40 to 95 percent in our experiments. The exact percentage of the pixels matched depends on the particular imagery, of course. 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, that is pixels belonging to some dense feature are assigned a disparity.

This paper is organized as follows. We start by discussing related work in Section 2. In Section 3 we describe the motivation behind our dense features. In Section 4 we explain how we compute and choose among the dense features. We conclude with experimental results in Section 5.

2 Related Work

In this section we briefly sketch most common dense approaches to stereo correspondence. These methods must deal with textureless regions. The basic idea behind all of them is to encourage "cooperation" among pixels, so that pixels in textureless regions get assigned smoothly varying disparities. We roughly divided such methods in a few groups below, according to the way they encode cooperation among pixels.

Area correlation methods [18, 21, 8] 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 [12, 6, 14, 28, 20]. 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 [15, 30] directly encourage nearby pixels to cooperate by local iterative schemes which propagate information from a pixel to its neighbors. Energy minimization methods [1, 7, 10, 4, 23] 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 [5, 13, 26, 27], The actual details vary significantly between these algorithms. Finding good regions to match through image segmentation is, of course, a difficult problem.

3 Dense Features for Stereo Correspondence

In this section we give motivation behind our dense features and describe them in detail. There are two main questions in designing dense features. First we need an intuitive understanding of what kind of properties make some region a good candidate for a dense feature. That is what kind of region is easy to match reliably. Secondly we need to transfer this intuition into a computationally feasible solution.

Our answer to the first question is as follows. Clearly the presence of intensity edges on the boundary of a region gives a good cue for matching and we want to incorporate this cue in a dense feature. When dealing with intensity edges, the usual problem is the selection of an appropriate threshold. The threshold should be above the noise level for a good performance. For stereo correspondence, there is a natural way of estimating when

an intensity edge is strong enough. It is strong enough if the intensity difference on the edge is larger than the error of matching the edge pixels. Thus 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 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.

Consider an example in Figure 2. The first two columns are a stereo pair consisting of a foreground object (a square with a hole) in front of a background. The intensity of the foreground is 170, and the intensity of the background is 140. The background and the foreground are corrupted by $N(0, 25)$ noise. The noise is strong, but it is not strong enough to destroy the edges of the foreground object in the left and the right images, and these edges give us a good cue for correspondence. The disparity of the foreground is 10. Consider the error surface at disparity 10, where the error surface is just the absolute difference between the left and the right images overlapped at disparity 10. We marked with bright color all pixels in this error surface where the error is smaller than the edge strength at that pixel. The resulting image is shown in the last column of figure 2. Observe that the foreground region is surrounded by the bright edges. Thus we are now facing the problem of segmentation, where the task is to segment a region surrounded by the bright edges.

We actually detect dense features not in the error surface, but in another 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 all the other pixels, thus enabling efficient detection of dense features. However if we apply the boundary condition to the error surface, then we are checking only that the boundary pixels form good matches at the disparity in question. 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 these pixels contain two completely unrelated textures in the left and the right images.

To avoid the problem described in the previous paragraph, we apply the boundary condition to a surface which we call the match surface. The match surface at disparity d is set to 1 for pixels that are likely to have disparity d and it is 0 otherwise. When enforcing the boundary condition, dense features are allowed to consist only of pixels which are set to 1 in the match surface. Thus not only the boundary pixels have to be checked as a good match, but also the pixels inside the boundary.

We want to allow for significant brightness changes between the corresponding regions, and thus pixels which differ in intensity significantly should be allowed to match. At the same time we wish to exclude from consideration regions with unrelated textures. Setting the match surface to 1 for smoothly varying regions of the error surface seems to work well for achieving these two purposes simultaneously. Section 4.2 explains how we compute the match surface in details.

We now come to the second question, which is how to compute the dense features. We do not have a very good answer to this question, it is a topic for further improvements. Ideally we need dense features which can overlap several disparities, since a textureless region can straddle many disparities. Also we would like to use a good contour based segmentation algorithm. Currently though we have only experimented with a very simple segmentation approach and only search for dense features which do not overlap several disparities. Furthermore we only test the left and the right boundaries of a region satisfy the "boundary" condition. Even with such a naive approach to dense feature computation we achieve impressive results, see Section 5. This give us some evidence that we are pursuing a useful direction in stereo correspondence.

Here is our current simple approach to segment a dense feature from the match surface. We independently prune the left and right boundary of the match surface until what remains satisfies the boundary condition. Then we compute connected components in the pruned match surface, and these connected components are our dense features. This algorithm is explained in details in Section 4.3.

4 Description of the Dense Feature Stereo Algorithm

In this section we give a complete description of our dense feature based stereo 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 disparities 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

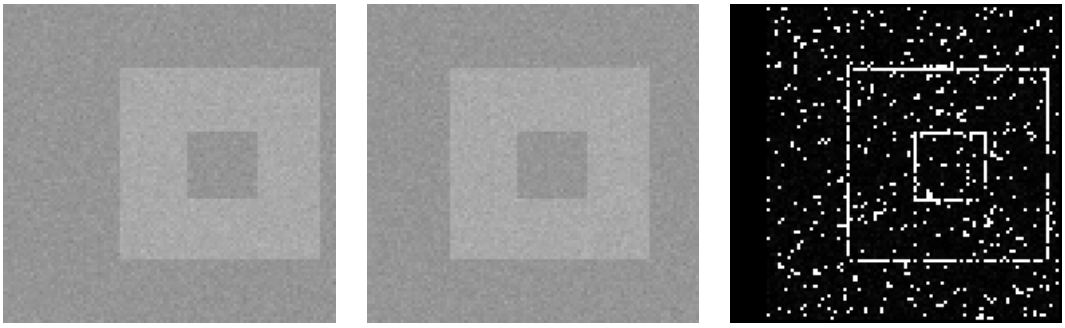


Figure 2. Square with a hole in front of background stereo pair. The last image shows in white the pixels at the disparity of the foreground for which the matching error is less than edge intensity at that pixel.

for all pixels p **do**
  $disparity(p) = NONE$
for $d = 0, \ldots max_d$ **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 $\{f_d^1, \ldots f_d^n\}$ in $M(d, p)$
  4. **for** $i = 0, \ldots n$
    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 3. Overview of the algorithm.

Section 4.1. The second step is to compute the match surface, see Section 4.2. The third step is to find all the dense features $\{f_d^1, \ldots f_d^n\}$ in the match surface, see Section 4.3. 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'. Note that this "density" can be different for distinct pixels of the same dense feature, that is density is a function of pixel. The importance of this property will be explained later. This final step is in Section 4.4. The summary of our algorithm is in Figure 3.

4.1 First Stage: Computing the Error Surface

In this section we explain how we find the error surface. Let us 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 will use notation $E(d, p)$ for the error surface at disparity d and pixel 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. We will keep the sign of that difference, because we will need it for the match surface computation. Our error surface $E(d, p)$ has two components, namely $E^r(d, p)$ and $E^s(d, p)$. The first component $E^r(d, p)$ just measures the difference in intensities between the two images, that is

$$E^r(d, p) = L(p) - R(p - d).$$

However even in the absence of noise, $E^r(d, p)$ measure is not accurate for pixels overlapping a 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 be worth it. Instead we use the method in [2] to construct the the second component of error surface $E^s(d, p)$ to be 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 $E^s(p,d)$ is the symmetric measure of similarity between pixels p in the left image and pixel $p-d$ in the right image:

$$E^s(d,p) = \text{sign}(E^r(d,p)) \cdot \min\left\{e_d^l(p), e_d^r(p)\right\}.$$

Here $sign()$ is the sign function, we use it to retain the right sign of the error surface.

4.2 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 of pixels.

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 from the match surface. To satisfy these two goals simultaneously, 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.

The algorithm to compute the match surface is in Figure 4. We start by initializing the match surface to 0. Then we sort all pixels in order of increasing $E^s(d,p)$. It can be done in linear time since the range of $E^s(d,p)$ is small. The next step is to go over all pixels p in order of increasing $E^s(d,p)$ 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. This 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 $E^s(d,p)$'s.

The second condition is that if $M(d,q) = 1$ for some $q \in N_p$, then the distance between the two intervals $[E^s(d,p), E^r(d,p)]$ and $[E^s(d,q), E^r(d,q)]$ is less than ϵ. This condition makes sure that all the regions which are set to 1 in the match surface correspond to smoothly varying error surface. Notice that we use intervals of values $[E^s(d,p), E^r(d,p)]$ instead of using just $E^r(d,p)$ or $E^s(d,p)$. The reason for using the whole interval is that due to the image sampling error we do not know where the matching error lies within this interval, we just know that it is somewhere in the interval. Using an interval works better in image regions with rapidly changing intensity.

The value of ϵ is related to the noise level in the images. However we found that in the imagery we experimented with, larger errors occur in textured regions, and are related to the image sampling artifacts rather, than to the camera noise. 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 the interval $[E^s(d,p), E^r(d,p)]$ is larger than in the low textured regions, and larger intervals are more likely to overlap. Thus 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.

4.3 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, that is the boundaries must be on intensity edges larger than the 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 6. However in the current implementation, we can deal with some of such uniform surfaces by

for all p
  $M(d,p) = 0$
Sort p in the order of increasing $E^s(d,p)$
for all p in order of increasing $E^s(d,p)$ **do**
  if $M(d,q) = 0 \quad \forall q \in N_p$
    $M(d,p) = 1$
  else if $\forall q \in N_p$ s.t. $M(d,q) = 1$ intervals $[E^s(d,p), E^r(d,p)]$ and $[E^s(d,q), E^r(d,q)]$ overlap within ϵ,
    $M(d,p) = 1$

Figure 4. Algorithm to compute the match surface at disparity d.

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

Figure 5. Algorithm to prune the left boundary.

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.

Recall that we have an interval $[E^s(d,p), E^r(d,p)]$ for the error surface. So we have a choice of which value in this interval to use when applying our boundary condition. We chose $E^r(d,p)$ because it is the larger of the two and thus gives a more conservative estimate.

We begin by pruning pixels on the left boundary of the match surface until 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 $E^r(d,p)$. We denote this difference by $avr(p,d)$, and it is computed in the 3 by 3 window, that is if W_p is the 3 by 3 window around p, then

$$avr(p,d) = \frac{1}{9} \sum_{q \in W_p} (L(q) - R(q-d)) .$$

The algorithm to prune the left boundary is given in Figure 5. 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 $|E^r(d,p) - avr(p,d)| + \sigma$ is larger than the intensity boundary between p and $p-1$ in the left image or $|E^r(d,p) - avr(p,d)| + \sigma$ is larger than the intensity boundary between $p-d$ and $p-d-1$ in the right image.[1] We remove such p from the match surface, that is we set $M(d,p) = 0$. Then we continue this pruning process until all left boundary pixels p satisfy the boundary condition.

Notice a new parameter σ, which we set to 5 for all our experiments. This is the second and the 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. The value of σ should be above the noise level, that is above ϵ, but the best choice for both parameters is a topic for future research.

We do similar pruning for the right boundary. Since we treat each scanline independently for the dense feature detection, there may be some inconsistencies between 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 [11, 29]. However even our simple algorithm works quite 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 compo-

[1] Notice that the left and the right images are treated in the same way by our algorithm

nents less than some minimum size, which we set to 25. The rest of the connected components are our dense features.

Figures 6(a,b) show the match surface and the dense features at disparity 14 for the scene in Figure 9(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 is other than 14. That is why when the 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.

4.4 Forth Stage: Choosing Dense Features

Some pixel p can be a part of two dense features (notice that these features are at different disparities, since dense features at the same disparity do not overlap). One reason is repeated texture. Consider Figure 7(a). This is a cut out of a repeated texture region from the scene in Figure 9(a). This region are the books right above the lamp. Figures 7(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 we do not enforce the boundary condition on the top and the bottom border. This allows some dense feature to be joined by a few extra regions, usually fairly small. Consider Figure 8(a). It shows a region occupied by a corner of the soda can to the right and slightly above the lamp in Figure 9(a). The background has texture due to the wall poster. Figure 8(b) shows the dense feature at the disparity of the soda can. The can grabbed a thin horizontal region due to the wall texture. Figure 8(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. We found the following definition of density to work well in practice. Let $M(d,p)$ be the match surface, and let $H^{nw}(d,p)$ be Manhattan distance from p in the north west direction to the nearest pixel q such that $M(d,q) = 0$. $H^{nw}(d,p)$ can be computed in one pass over the image for all pixels p. Similarly define H^{ne}, H^{sw}, and H^{se} to be the Manhattan distance from p to the nearest pixel q such that $M(d,q)$=0 in the north east, south west, and south east directions. If f_d^i is a feature at disparity d containing p, then

$$density(p, f_d^i) = H^{ne} + H^{nw} + H^{sw} + H^{se}.$$

Notice that $density(p, f_d^i)$ can be different from $density(q, f_d^i)$. This is important since in many cases some pixels of a dense feature do belong to the disparity of that dense features while others do not. With this definition of density, it is possible to break off the wrong pixels from the dense feature in the final assignment, while leaving the other pixels. With this definition of density, the regions in Figures 7 and 8 are placed at the correct disparities, as can be checked in Figure 9.

5 Experimental Results

In this section we present our experimental results on real stereo pairs, several of which have known 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 (that is the pixels which do not belong to any dense feature) are in black.

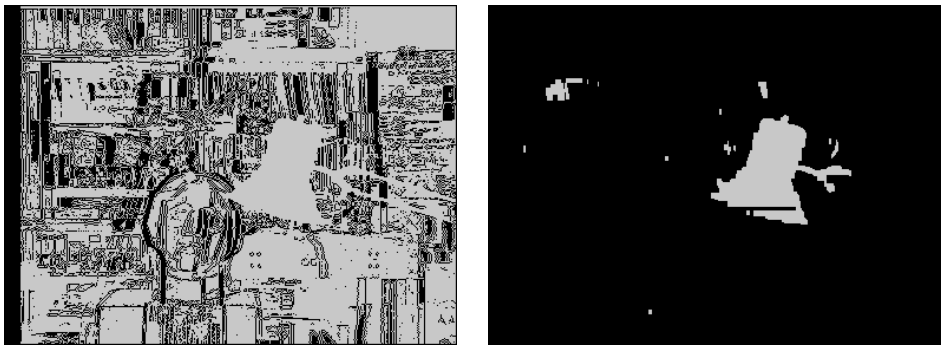
5.1 Handling of Nonlinear Errors

Before presenting our experimental results, we describe the last component, which is the handling of nonlinear errors in stereo imagery. In finely textured regions when the baseline between cameras is large, the noise is highly non linear. For such regions, even the sampling insensitive error measure $E^s(d,p)$ does not help. However we can use another error measure developed in [28], which seems to work well in textured regions. The basic idea is that intensity monotonicity is preserved in these regions, and it should be exploited instead of raw image intensities.

Let us measure local differences in intensities, retaining only their signs and not the magnitude. We define functions $sgn_l(p)$, $sgn_r(p)$, $sgn_a(p)$, and $sgn_b(p)$ as follows:

$$sgn_i(p) = \begin{cases} -1 & \text{if } L(p) - L(p \to i) < 0 \\ 1 & \text{if } L(p) - L(p \to i) > 0 \\ 0 & \text{if } L(p) - L(p \to i) = 0 \end{cases}$$

Here $p \to i$ stands for the pixel to the left, right, above, or below of p if $i = l, r, a, b$ correspondingly.



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

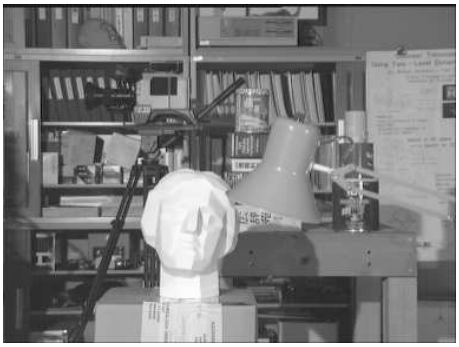
Figure 6. Comparison of match surface and dense features

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

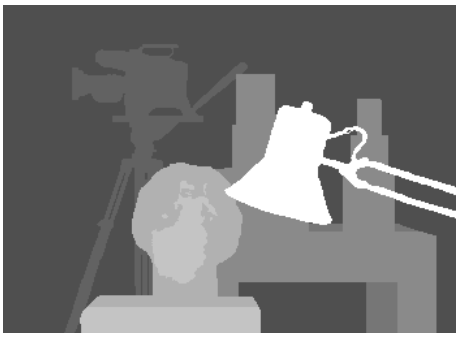
Figure 7. Dense feature overlap due to repeated texture

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

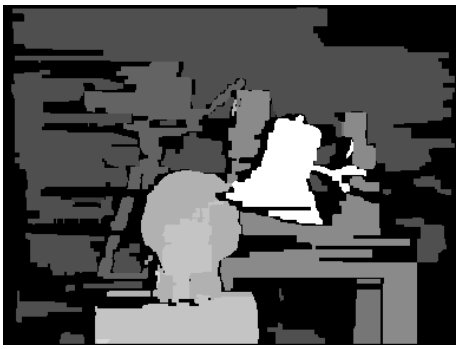
Figure 8. Dense feature overlap due to spurious texture



(a) left image



(b) ground truth



(c) our algorithm

Figure 9. Real imagery with dense ground truth

Functions $sgn_l(p-d)$, $sgn_r(p-d)$, $sgn_a(p-d)$, and $sgn_b(p-d)$ are defined similarly on the right image. Now define

$$E^t(d,p) = f(\sum_{i\in\{l,r,a,b\}} |sgn_i(p) - sgn_i(p-d)|),$$

where

$$f(x) = \begin{cases} 1 & \text{if x } \leq 2 \\ 0 & \text{otherwise} \end{cases}$$

Thus $E^t(q,d)$ measures how well signs of local variations match around p in the left image and $p-d$ in the right image. This is robust to all monotonic nonlinear changes. Notice that if the argument to function f is larger than 2, less than three of sgn_i functions match, so the use of $E^t(d,p)$ is unreliable and it is set to 0. This is expected in areas of low texture.

We use $E^t(d,p)$ at a stage separate from the stage where we use the interval $[E^s(d,p), E^r(d,p)]$. This second stage is for detecting textured regions effected by nonlinear errors which might have been missed, and it follows the stage we have described in Section 4. Only now $E^t(d,p)$ is used instead of the interval $[E^s(d,p), E^r(d,p)]$. This second stage for detecting textured regions is even simpler, since $E^t(d,p)$ takes on binary values. That is after $E^t(d,p)$ is computed, we find dense features in it directly without computing the match surface. The value of σ is also set to 5, as in the first stage.

5.2 Tsukuba Stereo Pair

Figure 9(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 Figure 9(b). The disparity map our algorithm computes is shown in Figure 9(c). The size of these images is 384 by 288, maximum disparity we search is 14. The running time is 1 sec, and 66% 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, which are textureless. Some parts of on the table are also not matched.

Out of the pixels that we match, 3.78% are found incorrectly, and 0.38% are off by not more than 1 pixel from the correct answer. The absolute average error is 0.06. This is the best performance out of the all the dense stereo algorithms compared in [24]. However direct comparison with the dense stereo algorithms is not fair because we only find a semi-dense disparity map.

There is a better way to compare our algorithm with the dense stereo algorithms. Some dense stereo algo-

% matched pixels	% ± 1 errors	% total errors
11	0.89	1.37
21	1.52	9.13
39	1.55	12.40
48	1.35	16.49
66	1.74	19.50
87	2.31	24.32
98	3.35	26.58

Figure 10. Percentage of matched pixels versus errors for the compact window algorithm

rithms compute a certainty map. A certainty map assigns higher numbers to pixels with higher confidence of their disparity estimate. There are many ways to compute a certainty map. One way is to assign high confidence to pixels with better matching scores. Another way is to assign high confidence to pixels if their estimated disparity has a narrow peak in the plot of matching cost versus disparity. In other words these are the pixels with a unique disparity giving better matching costs. These pixels typically lie in areas of high texture. We will threshold the certainty map and compare the percentage of matched pixels versus errors for some dense algorithms.

We have tried two methods of computing confidence maps for the dense stereo methods. First we tried to count only the disparity estimates from regions with high texture. This method can be used on a disparity map from any dense stereo algorithm, and we tried it on the graph cuts method in [4]. However that did not seem to give significant improvements in error counts.

Then we tried to threshold disparity map to leave only the pixels with the better matching score. We applied this thresholding to area correlation methods. We chose the compact window algorithm [28] because it performs reasonably well out of area correlation algorithms[2]. The results are summarized in figure 10. For the smaller percentage of pixels matched, the results do indeed improve, but at the same percentage of pixels matched as in our dense feature algorithm, which is 66%, our algorithm still performs significantly better.

5.3 Two Planes Stereo Pair

Figure 12(a) shows the left image of another stereo pair for which the dense ground truth is also known. This stereo pair is from Microsoft. Figures 12(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 29. The running time was 2 seconds, with 87% 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 four small regions which should be occluded are matched erroneously. Out of the pixels which our algorithm matches, 18.50% are matched incorrectly, 0.22% are off by not more than one pixel from the correct disparity.

5.4 Other real imagery with ground truth

Recently D. Scharstein and R. Szeliski have collected an impressive database of several stereo sequences for which they were able to compute the dense ground truth[3]. This database should become a long needed benchmark for testing stereo algorithms. The results our algorithm gives on this new data base are summarized in the table in figure 11. Image sizes are 431 by 381, and the largest disparity we search for is 21.

5.5 Birch Stereo Pair

Figure 13(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.

[2] See comparison of dense stereo methods at http://www.middlebury.edu/stereo/eval/results.html

[3] available at http://www.middlebury.edu/stereo/

name	$\pm$ 1 errors	% total errors	% matched	running time in seconds
Sawtooth	1.62	16.36	76	6
Venus	1.83	13.25	68	5
Bull	0.09	11.76	73	5
Poster	1.05	7.85	77	7
Barn1	0.62	7.83	83	6
Barn2	0.25	7.11	73	5

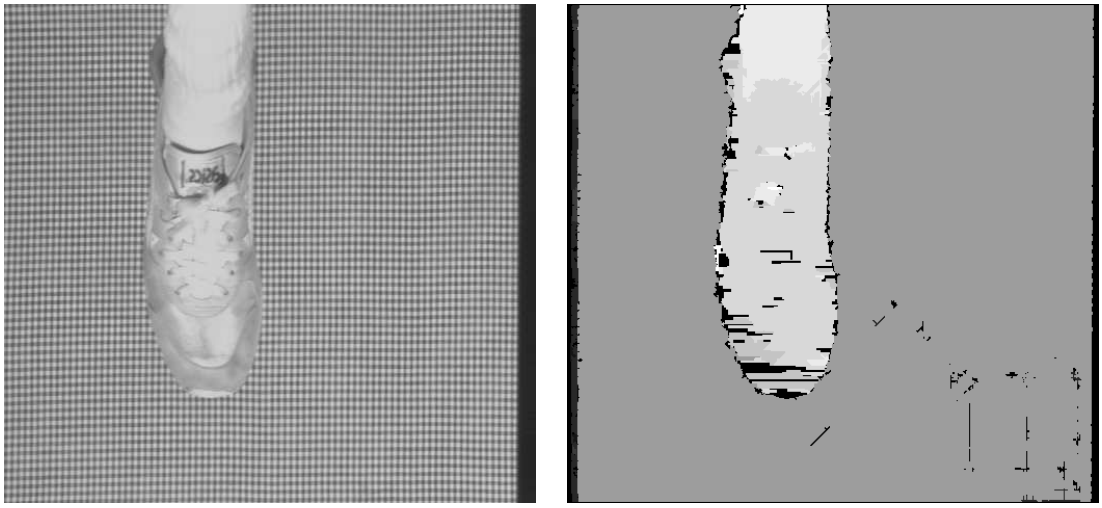
Figure 11. Performance of our algorithm on other data with ground truth.

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

Figure 12. Slanted planes stereo pair

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

Figure 13. Birch Sequence from SRI



(a) left image (b) our algorithm

Figure 14. Shoe sequence from CMU

5.6 Shoe Stereo Pair

Figure 14(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.

6 Discussion

We proposed a new approach to stereo correspondence. The basic idea that motivates our approach is to find regions in stereo pairs which are easy to match accurately, and we call such regions dense features. Based on the experiments we have conducted, we think that the dense feature approach is a useful direction for future research.

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, nonlinear errors, repeated texture, homogeneous image regions. It is robust with respect to parameters, they do not need to be tuned separately for each stereo pair. It is also very fast and efficient with memory.

The current implementation is quite simple, and there are many directions for future improvements. The biggest current limitation is that we cannot handle homogeneous sloped surfaces with slant other than horizontal. We plan to address this in our future work by 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 currently. 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, for example apply the methods in [11, 29].

Another direction for future research is a comprehensive study of different phenomena which effect the error between the corresponding pixels, and the insights such study might bear on the selection of parameters ϵ and σ.

Acknowledgments

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

References

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

[2] Stan Birchfield and Carlo 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. A variable window approach to early vision. *PAMI*, 20(12):1283–1295, December 1998.

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

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

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

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

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

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

[10] H. Ishikawa and D. Geiger. Occlusions, discontinuities, and epipolar lines in stereo. In *ECCV98*, pages I:232–248, 1998.

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

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

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

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

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

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

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

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

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

[20] M. Okutomi. A simple stereo algorithm to recover precise object boundaries and smooth surfaces. In *CVPR01*, pages xx–yy, 2001.

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

[22] 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.

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

[24] D. Scharstein, R. Szeliski, and R. Zabih. A taxonomy and evaluation of dense two-frame stereo methods. In *SMBV01*, pages xx–yy, 2001.

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

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

[27] H. Tao, H.S. Sawhney, and R. Kumar. A global matching framework for stereo computation. In *ICCV01*, pages I: 532–539, 2001.

[28] O. Veksler. Stereo matching by compact windows via minimum ratio cycle. In *ICCV01*, pages I: 540–547, 2001.

[29] S. Wang and J.M. Siskind. Image segmentation with minimum mean cut. In *ICCV01*, pages I: 517–524, 2001.

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