A Multifeature Correspondence Algorithm Using Dynamic Programming

C. V. Jawahar and P. J. Narayanan
International Institute of Information Technology
Gachibowli, Hyderabad 500 019. INDIA
Email: {jawahar,pjn}@iiit.net

Abstract

Correspondence between pixels is an important problem in stereo vision. Several algorithms have been proposed to carry out this task in literature. Almost all of them employ only gray-values. We show here that addition of primary or secondary evidence maps can improve the correspondence computation. However any particular combination is not guaranteed to provide proper results in a general sitiuation. What one needs is a mechanism to select the evidences which are apropriate for a particular pair of images. We present an algorithm for stereo correspondence that can take advantage of different image features adaptively for matching. A match measure combining different individual measures computed from different features is used by our algorithm. The advantages of each feature can be combined in a single correspondence computation. We describe an unsupervised scheme to compute the relevance of each feature to a particular situation, given a set of possibly useful features. We present an implementation of the scheme using dynamic programming for pixel-to-pixel correspondence.

1 Introduction

Many computer vision algorithms need to identify the projection of common scene structure into similar image features in multiple images. These images could be taken by different cameras simultaneously as is the case in stereo vision, where the focus is on recovering the geometric structure of objects in the scene by matching up the common scene structure between images. Other structure from motion algorithms may move the camera and attempt to match the same scene points through the sequence of images generated. The problem of stereo correspondence is to identify for each pixel in the source ("left") image a matching pixel in the target ("right") image such that they both are images of the same physical point.

Different methods have been used to compute stereo correspondence. Some use information from a single pixel alone, while others use information from a small neighbourhood around the pixel. Yet others use features derived from the image for matching. Various image features are used in the literature, such as intensity, edge strength, corner strength, texture, etc. The choice depends essentially on the scene, as different features work well for different pairs of views. Area based matching algorithms are good for scenes with good texture, edge based algorithms are good when edges are present, etc. There have been some attempts to formulate the correspondence problem in a general statistical framework using the maximum likelihood estimates for the pixels [4] or by estimating the Bayesian priors from the intensity distribution [2]. These, in essence, compute a simple similarity measure between gray-values of pixels in both the images and find the optimal matches by imposing new constraints, using penalty terms for the occluded pixels.

We present a scheme in this paper to adaptively select the combination of features that work best for a specific pair of images. The selection starts with a superset of features that could be relevant for matching between the two images. These could include intensity along multiple spectrums such as different colour bands, edge strength, texture measures, etc. The matching measures computed from these diverse features are combined, with appropriate importances assigned to each in the form of a weight, to yield a single measure of similarity or dissimilarity between two candidate pairs of pixels. The weight of a particular feature encodes its relevance or importance in matching the pair of images. We also present a scheme for estimating the weights for the features used which converges fast on typical images.

The importance of integrating multiple feature measures for stereo correspondence has been recognized in the literature [4, 5, 6], but practical implementations involving multiple features are rare. We introduced the framework of generalised correlation to combine diverse types of features in a flexible manner [5]. It was quite successful in combining multiple features under a correlation framework. The importances of the individual feature measures were, however, hand computed with no flexibility to adapt to a pair of images automatically. We later devised a technique to estimate the importances of each feature based on the siutation under the correlation framework [6].

An adaptive, non-supervised scheme to estimate the relevances of the feature measures used depending on the results of the matching is also presented in this paper. We present the results from implementing our scheme using dynamic programming. The methodology differs consider-

ably from the existing dynamic programming formulations of stereo [7, 4, 3, 1] in the way in which it integrates match measures computed using heterogenous features.

The basic framework for combining multiple features and estimating their relevances adaptively is presented in Section 2. A procedure to compute the importance of the features to solve a particular problem is described in section 3. Results directed at demonstrating the effectiveness of the scheme are presented in Section 4. Concluding remarks and directions for future work are presented in Section 5.

2 The Approach

At the heart of any stereo correspondence scheme is a measure of similarity or dissimilarity between a pair of pixels, one belonging to the left image and the other belonging to the right image. For two set of pixels $\{x_1, \ldots, x_M\}$ belonging to the first (left or source) image and $\{y_1, \ldots, y_N\}$ belonging to the second (right or target) image, find the mapping $x_j \rightarrow y_k$ such that x_j and y_k are similar pixels, being images of the same scene point. The mapping need not be one-to-one or onto. Some points in both images may not have a corresponding one in the other due to occlusion. The similarity computation can be carried out based on a feature vector computed for each pixel. Correspondence is typically computed using a similarity measure $S(x_j, y_k)$ or a dissimilarity feature $D(x_j, y_k)$. The matching point for x_j is the pixel y_k that maximizes S or minimizes D over a search space. It is often possible to limit the search space for each pixel based on geometric or other constraints. For instance, the epipolar constraint limits the search space for each pixel in the left image to a line in the right image. The ordering constaint, the smoothness constraint, etc., can also be used to limit the search space in some situations.

In this paper, we construct feature vectors associated with each pixel by stacking measures derived from one or more features to estimate the similiarity or dissimilarity between pixels of the left and right images. The sum of absolute or squared difference of the feature vector components between the pixels can serve as a dissimilarity measure. It is possible to explicitly model occlusions and associate a cost for them in the objective function. In such cases, the optimization for matching is not done for individual pixels, but for a set of pixels, such as a scan line of the left image, matching with a similar set in the right image. We use the square of the magnitude of the difference vector as a simple dissimilarity measure for illustration in this paper though the scheme can be used with other similarity or dissimiliarty measures. We also assume parallel rectified views are being matched, limiting the search for each pixel j in the left image to the pixels of the same scan line in the right image.

2.1 Multifeature Matching Measure

To combine the effects of multiple features into a single dissimilarity measure, we form a feature vector at each pixel. The feature vector $\mathbf{X}_j$ for the jth pixel is formed by stacking measures from different feature images. Each component of the feature vector contains a measure relevant for matching derived from an image feature. Thus, the first component may be the intensity in the red band, the second in the green band, the third may be the edge strength, etc. A *feature relation matrix* encodes the relative importance of each feature in the matching process as in [5]. The combined dissimilarity measure between pixel j in the left image and pixel k in the right image can be given by

$$D(j,k) = [\mathbf{X}_j - \mathbf{Y}_k]^T \mathbf{M} [\mathbf{X}_j - \mathbf{Y}_k]$$

where $\mathbf{X}_j$ is the feature vector for pixel j in the left image and $\mathbf{Y}_k$ is the feature vector for pixel k in the right image. The feature relation matrix $\mathbf{M}$ encodes the relationships between different feature measures of the feature vector. The case when $\mathbf{M}$ is a diagonal matrix is of special interest. In that case, each entry $m_{ii} = w_i$ represents the weight of the feature i in the matching process and gives its relative importance. The correspondence computation can be tuned by varying these values. Since the contribution from a feature cannot be negative, $w_i \geq 0$. The above dissimilarity measure can then be written as

$$D(j,k) = \sum_i w_i (\mathbf{X}_j^i - \mathbf{Y}_k^i)^2 \tag{1}$$

where, $\mathbf{X}_j^i$ is the ith component of the feature vector for pixel j. The matching point for pixel j is the pixel k in the right image that minimizes the dissimilarity measure D, given by

$$\arg\min_k D(j,k) \tag{2}$$

where, the k varies over the set of possible target pixels. The search for each pixel j in the left image can be limited to the pixels of the same scan line, within a range of disparities $[d_m, d_M]$ corresponding to the minimum and maximum possible disparities, if known.

2.2 Dynamic Programming Formulation

Dynamic programming is an effective strategy to compute correspondences for pixels. Consider a cost matrix as in Figure 1 with the nodes representing the weight of matching a pixel in left image with a pixel in right image. The cost of matching pixel m in left image and pixel n in right image can be computed based on the costs of matching all pixels in the left of these two pixels (the shaded square in Figure 1 or the rectangular area on left-top of $c(m,n)$). If one assumes the ordering constraint, the optimal "path" computed to match the pixels in left and right images will result in the best set of matches for the pixels in left and right images.

Dynamic program based formulations match lines to lines. They can also use the matches found for previous

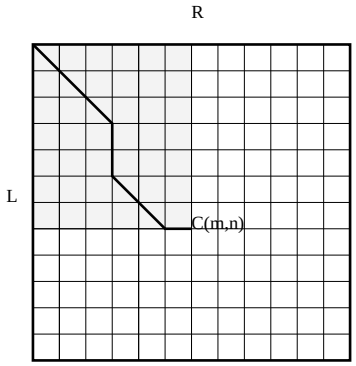

Figure 1. Matching two scan lines using a dynamic programming based formulation

pixels in the same scan line in the computation for the subsequent pixels [4, 3]. We extend this approach to find the matches using the multifeature dissimilarity measure given in Equation 1. We also model occlusions explicitly as done by Cox et al [4].

We use the following cost for matching a pixel j in the left image to a pixel k in the right image. Each pixel can be either occluded or matched. The dissimilarity measure in the case of occlusions is a constant. The modified dissimilarity measure can be given by

$$D'(j,k) = \begin{cases} C_o & \text{if there is an occlusion} \\ D(j,k) & \text{otherwise} \end{cases} \quad (3)$$

where C_o is the cost of occlusion. We should optimize the total cost of matching a scan line of the left image with the same scan line of the right image under the above formulation. The objective function for minimization is given by

$$J = \sum_{j \in S_j} D'_j \quad (4)$$

where j belongs to the set S_j of pixels in the left image and D'_j is the optimal matching cost for j over the scan line in the right image. The set S_j could be a scan line, a partitioning of the image based on any criterion, or the whole image itself. A possibly different set of weights will be computed for each feature over each partition S_j, as we will see later. We seek to find the individual matches for the pixels of the scan line that minimizes an aggregate measure represented by J.

3 Estimating Feature Relevances

The minimization of J has two parts. Minimization of J for each source pixel j over the target scan line and minimization of J over the weights w_i. The dynamic programming formulation achieves the first part as given in Equation 2, keeping weights fixed. We minimize over the all possible weights $W = \{w_i\}$ using the partial derivatives of J with respect to the weights of the features of the feature vector. We rewrite the objective function given in Equation 4 as given below.

$$J = \sum_i w_i^2 \sum_{j \in S_j} {}^iD'_j \quad (5)$$

The second summation aggregates the contribution of feature i in the matching process by summing its contribution ${}^iD'_j$ for each pixel j over a scan line. The form of the objective function given in Equation 5 enables us to identify and weight the contribution of each feature separately and provides analytical tractability to the optimization problem. The use of the same weights a second time in Equation 5 (it is already present in the expression for D given in Equation 1) enhances the impact of each feature and makes it possible for J to be optimized in two steps. The dynamic programming algorithm will optimize J with respect to the target pixel as mentioned above. The procedure to optimize it with respect to the weights w_i is given below.

An unconstrained minimization of J with respect to W is impossible, as $w_i = 0$ will be the minimum. We have already mentioned the constraint $w_i \geq 0$. Since the interest is only in finding the correspondences which will yield optimal value of J, we can impose the following constraint without any loss in generality.

$$\sum_{i=1}^{p} w_i = 1 \quad (6)$$

This restricts the weights to lie on a hyperplane in the positive orthant. This continous values of feature relevances allows a smooth varaion of importances and result in more stable iterative algorithm.

We use the following Lagrangian for the optimisation:

$$F(W,\lambda) = \sum_{i=1}^{p} w_i^2 \sum_j {}^iD'_j - \lambda(\sum_{i=1}^{p} w_i - 1)$$

Differentiating the Lagrangian with respect to w_m and equating to zero

$$\frac{\partial F}{\partial w_m} = 2w_m \sum_j {}^iD'_j - \lambda = 0$$

Solving for w_m and substituting it in Equation 6, we can get

$$\lambda = \frac{1}{\sum_{k=1}^{p} \frac{1}{2\sum_j {}^kD'_j}} \text{ and } w_m = \frac{1}{\sum_{k=1}^{p} \frac{\sum_j {}^mD'_j}{\sum_j {}^kD'_j}}. \quad (7)$$

Thus the weight w_m for each feature m can be updated, possibly for use in the next iteration, using Equation 7. Features with high costs of matching will be reduced in importance and vice versa, adaptively adjusting to the views based on the relative performance of each feature in the matching. The summation over j can be performed over each scan line, over the entire image, or over any other partitioning of the image. Accordingly, a set of weights will

be computed for each scan line, for the entire image, or for each partition, respectively. In our dynamic programming situation, the optimization of J is performed over each scan line independently. Hence, we use individual scan lines as our source partitiion S_j.

4 Implementation and Results

Since the ordering constraint is valid for epipolar corrected image pairs, dynamic programming [7, 4, 3] can be used for this task quite effectively, carrying forward the minimum matching cost and the matching point as the scan line in the left image is traversed. At each point, the cost of matching pixels j and k in left and right images, $C(j,k)$, is given by

$$\begin{aligned} C(j,k) &= \min\{C(j-1,k-1) + D(j,k), \\ &\quad C(j,k-1) + C_o, C(j-1,k) + C_o\} \end{aligned} \quad (8)$$

The C values are initialized to $C(i,0) = i * C_0, \forall i$ and $C(0,j) = j * C_0, \forall j$. A zeroth pixel matching with ith one implies an occlusion of i pixels. Once the optimal cost of matching the last pixel of the scan line is computed, the optimal path can be traced back by analyzing which term provided the minumum for each match in Equation 8. The first term corresponds to no occlusions, the second to left occlusion, and the last to right occlusion. The disparity for pixel i is $|i-j|$ if $C(i,j)$ is present on the optimal path.

Many constraints have been tried to improve the matching based on dynamic programming. We use the horizontal and vertical cohesivity constraints employed by Cox et al. [4]. Cohesivity constraints minimise the number of discontinuities in horizontal and vertical directions and provide sharp and crisp disparity maps. The constraints associated with the intensity edges to model occlusion given by Birchfield and Tomasi [3] could also be used.

Our method also keeps track of the costs of individual features along the optimal path. These are used to evaluate the relative importances of the features using Equation 7. The optimal path computed using the current set of weights using dynamic programming optimizes the D'_j components of the objective function. Estimation of the weights based on the costs of individual features optimizes J with respect to the feature weights. These steps can be repeated till the change in weights $\sum_i |w_i - w_i^{old}|$ is less than a threshold ϵ.

In the rest of this section, we provide examples to validate the usefulness of our algorithm for stereo correspondence. In these examples, the set S_j includes all pixels. Thus, whenever we estimated the feature relevances, we computed only one set of weights for the entire image. The images used were all 256×256. The ϵ used to estimate convergence for the total change in absolute weights between successive iterations was 0.0001.

Example 1 We study the effectiveness of using multiple features for correspondence on a pair of random dot stereograms in this example. The synthetic structure used is a wedding cake structure, popular in analyzing stereo algorithms, with three levels of disparities of 1, 2, and 5. The texture images used have 10% of the pixels assigned a random gray value in the range [0,255]. The texture image can have more than one such band, similar to RGB or other multispectral images. In that case we use each band as a different feature for matching. The disparity map computed with only one band had 1011 misclassified pixels, when compared with the true disparity map. We then made a colour image using three random image bands for texture. The disparity map is then computed using three bands, with equal importance given to each. The number of misclassified pixels reduced to 364 in this case. We estimated the relative importances of the three features using the procedure given in the previous section. All features were estimated to be equally relevant by our procedure. This was quite expected as each band essentially contained equal information.

The above example brings out the advantages of using multiple features for correspondence. The additional features need not be additional information such as those present in other spectral bands. The next example demonstrates how derived secondary features can be used effectively to improve the correspondence in presence of a natural texture. The pixel-to-pixel matching algorithms are very sensitive to the photometric variations and noise when using gray level values alone. Integrating derived features with the gray level values in the matching process can reduce the sensitivity to a large extent as the following example demonstrates.

Example 2 In this example, we consider a natural image texture, comprising of regions with strong and medium variations, shown in Figure 2(a). A wedding cake structure was imposed on this to generate the right image. Additionally, an additive zero-mean Gaussian noise with standard deviation $\sigma = 5$ was also introduced. The left and right images are shown in Figure 2(a) and Figure 2(b) and the true dispaity map is shown in Figure 2(h). The disparity map computed using the gray level alone, shown in Figure 2(e), is very noisy. The well known weakness of pixel-to-pixel matching schemes using a single feature in the presence of noise is demonstrated here. We subsequently integrated two derived features to the matching process using our framework. The edge strength – the magnitude of the edge vector obtained using simple Sobel operators in horizontal and vertical directions – was the first derived feature used. Texture number – a ternary number representation of the neighbourhood gray-values, whether they are less, more or equal compared to the present pixel – was the second [8]. The texture number encodes the local relationships of the pixel's gray level value with those of its neighbours. The texture unit number for pixel to a is defined as $T_a = 3^i E(a_i, a), i = 1 \ldots 8$ where $a_1, \ldots, a_8$ are the neighbours of a and $E(a_i, a)$ is 0, 1, or 2 according to the

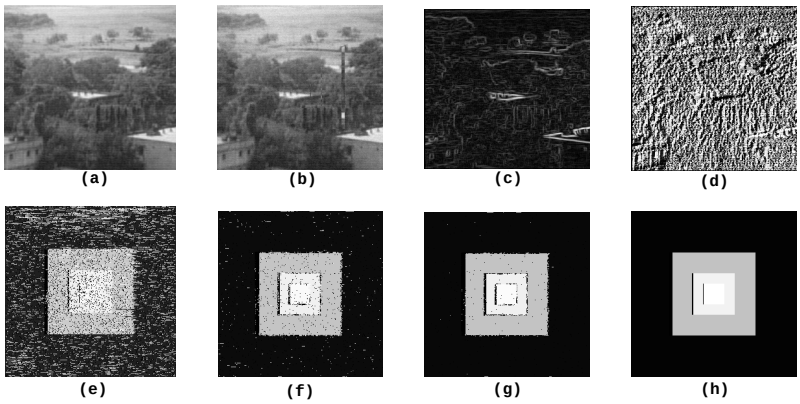

Figure 2. Correspondence computed with derived features (Refer to the Example 2 in the text)

gray-value of a_i is less, equal or more than that of a. The feature images corresponding to the two derived features are shown in Figure 2(c) and Figure 2(d) respectively. Correspondences were computed using combinations of these three features. Figure 2(f) shows the disparity map computed using the gray value and edge strength and Figure 2(g) shows the disparity image computed with all three. In each case, the features were weighted equally. The additional feature images reduced the mismatches considerably as can be seen from the disparity maps. Disparity map computed with gray-value alone had 21954 misclassified pixels. The combination of edge and gray-value reduced this to 3373 and the combination involving all three features reduced it further to 1614 pixels.

The above examples demonstrate the advantages of using heterogenous features to improve the correspondence accurracy. We now explore the effect of estimating their relative importances using the non-supervised procedure we presented. Emphasising some features above the others adaptively can improve the correspondence performance further, depending on the situation.

Example 3 We estimated the feature relevances using the procedure described in the previous section to the above example to compute the weights of the three features used. For this, the performance of each feature was independently computed while matching and the weights were adjusted using Equation 7 iteratively. The process converged in 28 iterations with a weight vector of $[0.11, 0.41, 0.48]^T$. Converegnce properties were excellent with change in weight going below 0.1 within 6 iterations and below 0.0001 within 28 iterations. The disparity map computed with the estimated weights is shown in Figure 3a. This further brought down the number of misclassified pixels to 1302.

Example 4 The estimated weight of each feature represents its relative importance in the matching process for the specific pair of images. In the presence of noise in an image, our method to estimate feature relevances automatically takes into account the noise content in each band or feature. Each feature can subsequently be emphasised or deemphasised. To demonstrate this, we consider a random-colour (three band) stereogram. Additive zero-mean Gaussian noise of $\sigma = 1, 5$ and 10 was added respectively to the first, second and third bands. The disparity map with equal weights to each, shown in Figure 3b, had 12363 misclassified pixels. The iterative feature relevance estimation procedure converged in 33 iterations and yielded a weight vector of $[0.77, 0.15, 0.08]^T$. Iterations stopped only when the change in weight was below 0.0001. The disparity map using the estimated weights, shown in Figure 3(c), had 266 misclassified pixels. The change in weight is plotted against the iteration number in Figure 3(d) to study the convergence properties of the iterative procedure. It can be seen from the graph that the convergence was fast and that the weights changed little after 5 or 6 iterations.

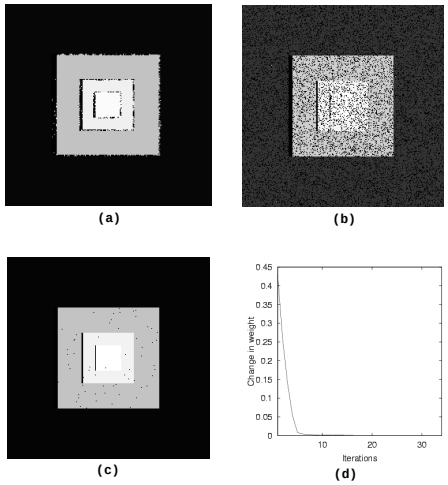

Figure 3. (a) Final disparity map computed for Example 2. (b) Disparity map with equal emphasis of all bands for a noisy random colour stereogram. (c) Disparity map for the same using optimal weights. (d) Convergence rate of the iterative algorithm.

Figure 4. Results on "bush" images (top) and the "meter" images (bottom)

Example 5 We now show the results of running our algorithm on a number of standard real images. The real images do not have the ground truth and hence the quantitative analysis presented above cannot be performed on them. The effectiveness and behaviour of our algorithm is better explained using the synthetic examples presented above. Figure 4 shows the results on the "bush" image pair and the "meter" image pair. The results are as good or better than the raw matching results of any pixel-to-pixel matching algorithms. Most algorithms improve their final results using tuned post-processing operations. Since the point of this paper is to demonsrate the scheme of feature integration in a pixel-to-pixel framework, we have not applied any post-processing to the results given by the program. The figure shows results of using one feature (gray level) and two features (gray level and the Sobel edge strength) at each pixel for the matching. The average weights vector estimated by our algorithm for the two features for the shrub image was $[0.786, 0.214]$ and for the meter image it was $[0.914, 0.086]$. The edge information is dense for the shrub image and was relevant to the match. It was not as reliable for the meter images as there were fewer edges in it.

5 Conclusions and Future Work

In this paper, we presented a stereo correspondence algorithm using dynamic programming that can integrate multiple types of features in a flexible manner. We also presented a non-supervised procedure to compute the relevance of each feature in a multifeature framework based on a pair of example images. The iterative estimation process can be tuned to a new situation in a few iterations. Our algorithm is most suitable to situations where a couple of representative pairs of images can be used for learning the relative importances of the features to be used for correspondence computation. These can be used subsequently for the computation on the actual images. One such situation is dynamic stereo, or stereo computed between corresponding frames of two video sequences of the same scene. Here the characteristics of the images relevant for stereo matching do not change much within the sequence. Thus, the first few frames can be used for computing the feature weights, which can be used for all subsequent frames.

References

[1] A. Bensrhair, P. Miche, and R. Debrie. Fast and automatic stereo vision matching algorithm based on dynamic programming method. *Pattern Recognition Letters*, 17:457–466, 1996.

[2] P. N. Belhumeur, "A Bayesian Approach to Binocular Stereopsis," *International Journal of Computer Vision*, vol. 10, pp 237–262, 1996.

[3] S. Birchfield and C. Tomasi. Depth discontinuities by pixel-to-pixel stereo. *International Journal of Computer Vision*, pages 269–293, 1999.

[4] I. J. Cox, S. L. Hingorani, S. B. Rao, and B. M. Maggs. A Maximum Likelihood Stereo Algorithm. *Computer Vision and Image Understanding*, 63(3), 1996.

[5] C. V. Jawahar and P. J. Narayanan. Generalised Correlation for Stereo Correspondence. In *ACCV 2000*, pages 631–636, 2000.

[6] C. V. Jawahar and P. J. Narayanan. Feature Integration and Selection for Pixel Correspondence. In *Indian Conference on Computer Vision, Graphics and Image Processing (ICVGIP)*, pages 330–337, 2000.

[7] Y. Ohta and T. Kanade. Stereo by Intra- and Interscanline Search Using Dynamic Programming. *IEEE Transactions on PAMI*, 7:139–154, 1985.

[8] L. Wang and D. C. He. Unsupervised textural classification of images using the texture spectrum. *Pattern Recognition*, 25(3):247 – 255, 1992.