Reclustering Techniques Improve Early Vision Feature Maps

Alex Cozzi Florentin Wörgötter

Institut für Physiologie, Ruhr-Universität Bochum,

D-44780 Bochum, Germany

email: {cozzi,worgott}@neurop2.ruhr-uni-bochum.de

FAX: +234 709 4912

Reclustering Techniques Improve Early Vision Feature Maps

Reclustering Techniques Improve Early Vision Feature Maps

Abstract

We propose a recursive post-processing algorithm to improve feature-maps, like disparity- or motion-maps, computed by early vision modules. The statistical distribution of the features is computed from the original feature-map and from this the most likely candidate for a correct feature is determined for every pixel. This process is performed automatically by a clustering algorithm which determines the feature candidates as the cluster centers in the distribution. After determining the feature candidates a cost function is computed for every pixel and a candidate will only replace the original feature if the cost is reduced. In this way a new feature-map is generated which, in the next iteration, serves as the basis for the computation of the updated feature distribution. Iterations are stopped if the total cost reduction is less than a pre-defined threshold. In general, our technique is able to reduce two of the most common problems that affect feature-maps, the sparseness, i.e., the presence of areas where the algorithm is not able to give meaningful measurements, and the blur. In order to show the efficacy of our approach, we apply the reclustering algorithm to simple versions of stereo- and optical flow algorithms, which initially produce a lot of errors, showing results for synthetic and natural images. An advanced example of its application to more complex feature maps is also presented.

keywords: reclustering, feature maps improvement, segmentation, clustering, stereo, optical flow.

1 Introduction

A fundamental problem in early vision is that of reconstructing the three-dimensional structure of a scene from the luminance information available in one or more images.

Commonly this is achieved by computer vision systems in several stages. The first stage transforms the luminance representations into specific features—edges, estimates of disparity, motion, etc. . The output of this stage is called a *feature map* and it is roughly equivalent to Marr's "primal sketch" [17]. In the next stage(s) the feature maps—which are still incomplete and erroneous—are corrected by various means in order to arrive at a dense *improved feature map.* Mostly this is done by regularization techniques (see below). Marr's stages of inferring shape from the primal sketch to arrive at his "full $2\frac{1}{2}$-D sketch" [17] go beyond the improved feature map by finally recovering the 3D-structure of the visual scene as good as possible.

Stereo vision [2, 6, 9] and optical flow analysis [11, 15, 19, 1, 3] are instances of modules that have undergone considerable investigation by the vision research community.

A stereo module extracts depth information from a *stereo pair*, i.e., two images of a scene taken from different view points. The image cues used to derive such information are *disparities*, differences in the relative positions of the images of the same object in the stereo pair. In the ideal case, when the characteristics of the imaging system are known and a complete map of disparities can be constructed, it is possible to recover the three-dimensional structure of the parts of the scene visible in both images. In a similar way, it is possible to recover shape information analyzing the motion patterns in a sequence of images of a static scene taken from a moving camera. Still all these feature maps produced by visual modules are affected by errors, noise and areas where no information could be obtained by the images.

A general method to overcome the sparseness and the imprecision of feature maps computed by stereo and motion algorithms and to build an improved feature map is regularization [21]. Regularization can effectively eliminate noisy measurements, reduce the overall imprecision of the disparity map and generate a dense description of the scene by fitting several neighboring measurements with a smooth function. On the other hand, if the scene contained true discontinuities they will be also smoothed. The tradeoff between smoothing across false discontinuities while preserving real ones is controlled by the value of the regularization constant. Finding the best value of the regularization constant is critical to obtain good results with regularization techniques but is a difficult problem on its own. Recent work concentrates on devising explicit discontinuity-preserving regularization schemes [8, 23] or expressing the regularization problem in terms of the Markov random fields framework [14].

We propose an alternative general post-processing stage, that can improve many different kinds of feature maps. Our approach essentially consist of a recursive feature segmentation algorithm based on the assumption that the distribution of the features in the feature maps have some inner structure that can be identified by an appropriately chosen function. Under this assumptions we find that many feature maps can be significantly improved. We start with a simple example to explain the basic technique, then we show how to apply our so called *reclustering algorithm* to two common problems in early vision: disparity estimation and motion analysis. We discuss the problems involved in the processing of real images and how to deal with them. We present an example of a more complex reclustering scheme and we end with a discussion of the advantages and disadvantages of the new algorithm.

2 The Reclustering Algorithm

The structure of the reclustering algorithm is related to approaches which approximate erroneous features of the feature map by estimates taken from the statistical distribution of all features (see Sec. 4).

In the process of vision sets of objects in the three-dimensional space—objects, surfaces, illuminants, etc. —project into one or more images in the form of luminance values. This process can be formally described as function from the three-dimensional space $\mathcal{W}$ to the two-dimensional image space $\mathcal{I}$:

$$\mathcal{W} \xrightarrow{p} \mathcal{I} \tag{1}$$

where the projection function p is the perspective transform. A visual module is another function v from the image space $\mathcal{I}$ to the feature map space $\mathcal{F}$

$$\mathcal{W} \xrightarrow{p} \mathcal{I} \xrightarrow{v} \mathcal{F} \tag{2}$$

Now, regularities and smoothness in the space $\mathcal{W}$ lead to structured patterns in the structure of $\mathcal{F}$, since what the visual module function v tries to do is to recover information about $\mathcal{W}$ from the image space $\mathcal{I}$. This structure in $\mathcal{F}$ can be evidenced by an appropriately designed function r that project "related" points of $\mathcal{F}$ on the same point of a new space $\mathcal{C}$, the *clustering space*:

$$\mathcal{W} \xrightarrow{p} \mathcal{I} \xrightarrow{v} \mathcal{F} \xrightarrow{r} \mathcal{C} \tag{3}$$

Related areas of $\mathcal{F}$ can then be identified as the inverse images of the points of $\mathcal{C}$ with respect to r.

The effect of noise and errors is that related points in $\mathcal{F}$ do not project anymore into a single point of $\mathcal{C}$. Still, if r is accurately chosen, the noisy points of $\mathcal{C}$ spreads around the original noise-free point. From the cluster it is then possible to estimate the position of the ideal, noise-free point. The idea of the reclustering algorithm is to use the estimated positions of the noise-free points to to improve the feature maps in the space $\mathcal{F}$.

The improvement stage of the reclustering algorithm uses the inverse images of the estimated noise-free points of $\mathcal{C}$ as possible replacement candidates for the points of $\mathcal{F}$. A point in $\mathcal{F}$ is replaced if one of the candidates minimizes a cost function defined on the space $\mathcal{F}$.

The details of the technique are explained most easily with the help of a simple example based on a stereo module. Thus, the features that compose the feature map $\mathcal{F}$ represent disparity values that encode corresponding points between the images of a stereo pair. For the sake of simplicity, we assume that the epipolar lines coincide with the horizontal lines of the image pair. In this simple case corresponding points have the same horizontal coordinate in the two images and the problem of finding corresponding points is reduced to a 1-dimensional search. The structure of $\mathcal{F}$ is that of a two dimensional, $N \times M$ lattice of disparity values:

$$\mathcal{F} = \{(f, i, j) \in \mathbb{R} \times \{0, \ldots, N\} \times \{0, \ldots, M\}\} \tag{4}$$

We choose as projection function r a function that simply removes the position information from the space $\mathcal{F}$, i.e., $r(f, i, j) = f$. This means that elements of $\mathcal{F}$ with the same disparity value project onto the same point of $\mathcal{C}$, defined as:

$$\mathcal{C} = \{f \in \mathbb{R}\} \tag{5}$$

In Fig. 1 **c** we depict two scan-lines of a stereo pair, denoted as “left” and “right”. When the intensities values are matched, we identify three disparity regions in the left image: the first three pixels with a disparity of zero, the last two pixels with a disparity of two, and the central three pixels with a disparity of one. The central line in this picture shows the correct disparity map for the left scan-line, which should be the result of the algorithm if it works correctly.

As cost functional in this example we use the absolute difference between the luminance value of corresponding pixels (Eq. 6). In order to deal with fractional disparities we need to compute the luminance of the right scan-line with sub-pixel accuracy, to this end we introduce in the cost function a simple linear interpolation scheme (Eq. 7). The $\mathrm{I}_{\mathrm{left}}, \mathrm{I}_{\mathrm{right}}$ are the luminosity functions of the the two scan-lines respectively, and d_x is the estimated disparity at position x.

$$\mathrm{Cost}(x, d_x) = |\mathrm{I}_{\mathrm{left}}(x) - \mathrm{L}(x + d_x)| \tag{6}$$

$$\mathrm{L}(z) = (1 - \lfloor z \rfloor)\mathrm{I}_{\mathrm{right}}(\lfloor z \rfloor) + (\lfloor z \rfloor)\mathrm{I}_{\mathrm{right}}(\lceil z \rceil) \tag{7}$$

In Fig. 1 the first two steps of the reclustering algorithm are outlined and the final state is shown. In the beginning an external stereo algorithm [7] computes the disparities, depicted in the top panel ("computed disparity"). In the computed disparity map the sharp changes between different disparities values could not be recovered because of the windowing effects leading to smooth variations across disparity boundaries. The last line represent the costs for each one of the pixels of the disparity map: the absolute difference between the left scan-line values and the interpolated right scan-line values (Eqs. 6–7).

The first step of the reclustering algorithm is to project the features of the feature map into the space $\mathcal{C}$, the space of the distribution of the disparity values (Fig. 1, **a**, right) and to extract the peaks of the disparity distribution by a clustering algorithm. In the example we apply the K-means algorithm initialized with three centers uniformly distributed between the minimum disparity value (zero) and the maximum disparity value (two). The choice of the clustering algorithm should be dictated by the complexity of the problem and by efficiency considerations. For our experiments even the simple K-means algorithm give

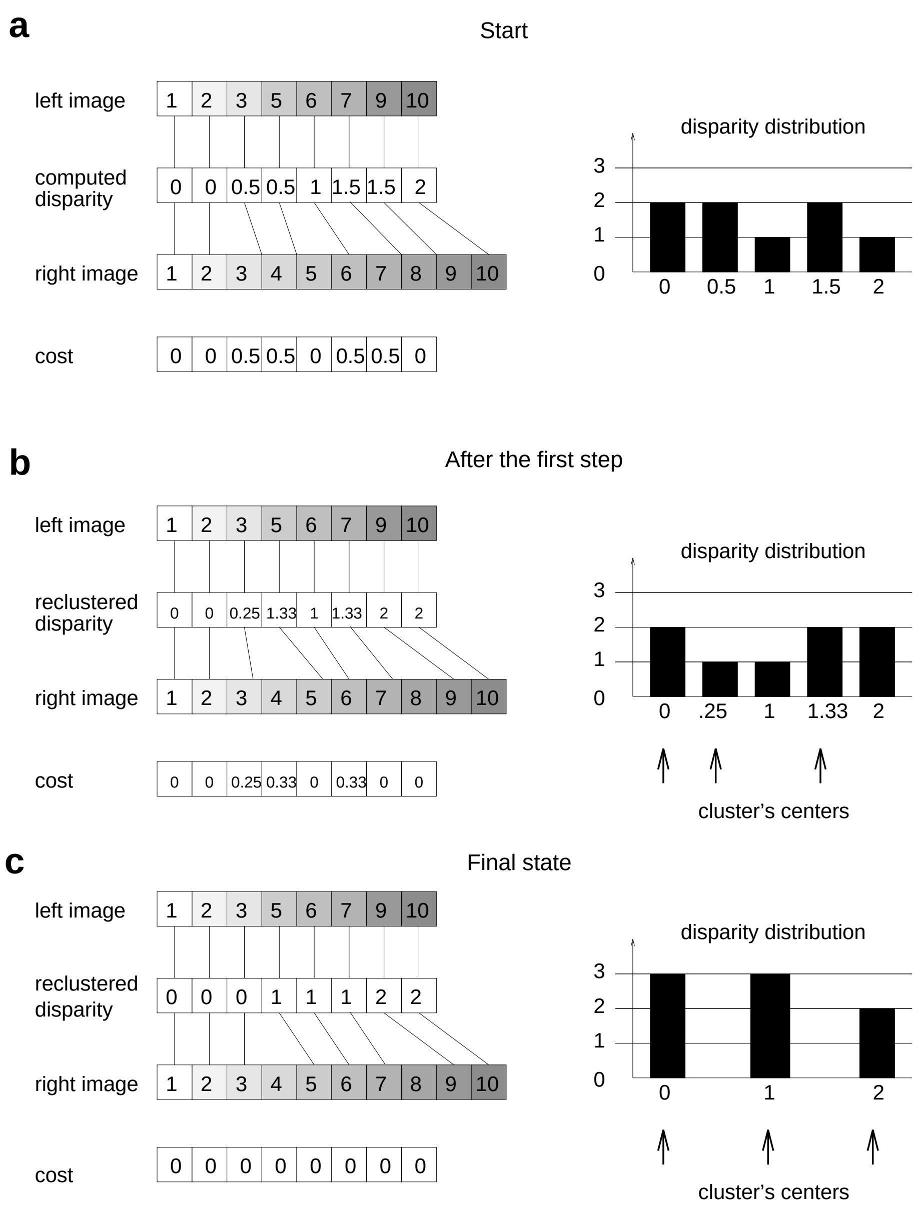


Figure 1: Reclustering example. In (**a**) is the initial state. After one iteration (**b**) of the K-means algorithm the centers reach the positions $(0.25, 1.\overline{3}, 2)$. Then the disparity map is improved usign the centers as possible alternative disparity values, with the results that four disparity values have been modified and their cost reduced. In (**c** is shown the final state where the algorithm converges.

satisfactory results.

The reclustering algorithm uses the position of the centers extracted by the K-means clustering as possible alternative disparity values to be substituted into the disparity map if in this way the cost is reduced.

It can be useful to apply the reclustering algorithm iteratively to the new disparity map, at each iteration reducing the cost associated with the disparity map and thus converging towards the optimum (Fig. 1, **c**). There are different ways to decide how many iterations are necessary. One possible stopping criterium is to iterate a fixed number of times, a second one is to iterate until the total cost is less than a fixed threshold, or until an iteration step decreases the sum of the costs by less than a fixed threshold.

3 Application to early vision

With an appropriate choice of the projection function, of the cost function and of the clustering algorithm the reclustering algorithm can be used to improve the feature maps generated by many early-vision algorithms. We presented a simple example of how to apply the reclustering algorithm to the problem of stereo vision. Using a two dimensional clustering space $\mathcal{C}$ and generalizing the cost function to deal with two dimensional shifts the algorithm can equally well be applied to the improvement of motion maps. In the following sections we describe two examples of the application of the reclustering algorithm to early-vision problems in artificial scenes before we turn to real images. In some of the following examples we use very simple early vision algorithms, which by themselves produce plenty of errors. The reason for this is that we do not intend to demonstrate any optimal early vision algorithm. Instead our goal is to show the quality of the improvement achieved by reclustering which can be judged much better when starting with poor initial results.

3.1 Disparity estimation

Fig. 3 shows an example of applying the reclustering algorithm to a stereo algorithm [7]. The input of the system is a noise-free computer-generated stereo pair (Fig. 2) The squares are covered by random-dot patterns in order to get output from the stereo algorithm. The images on the top of Fig. 3 show the original output of the phase-based stereo algorithm of Fleet et al. [7]. On the left side of the figure we plot the disparity distribution, on the right side the disparity map, rendered as a gray-scale image. Part **b** shows the result after the first iteration of the reclustering algorithm. The sharpening of the disparity distribution is noticeable in the plot on the left. The bottom row (**c**) shows the result after ten iterations of the reclustering algorithm: the error has been further reduced and the reclustering algorithm has succeeded in eliminating almost all the outliers.

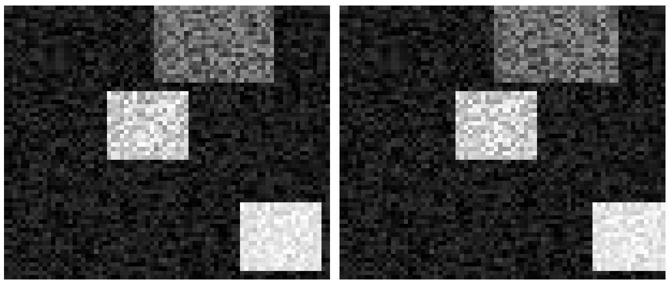


Figure 2: A computer-generated stereo pair

3.2 Optical flow computation

A second example is the application of reclustering to the recovering of the optical flow. Optical flow algorithms take as input a sequence of images and for each

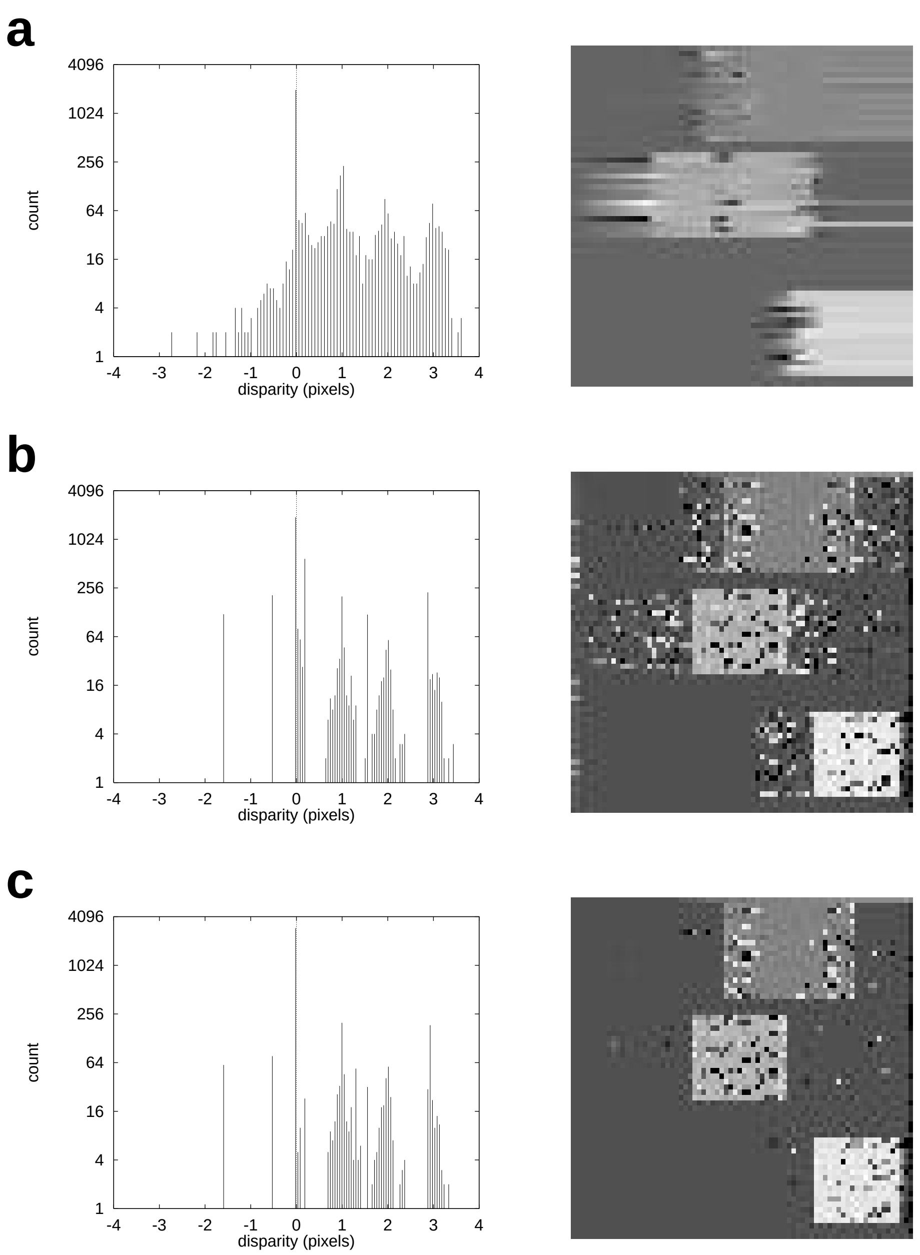


Figure 3: The reclustering technique applied to a stereo algorithm

pixel position produce an estimate of its motion, in the form of two-dimensional flow vectors [11, 15, 19, 1, 3].

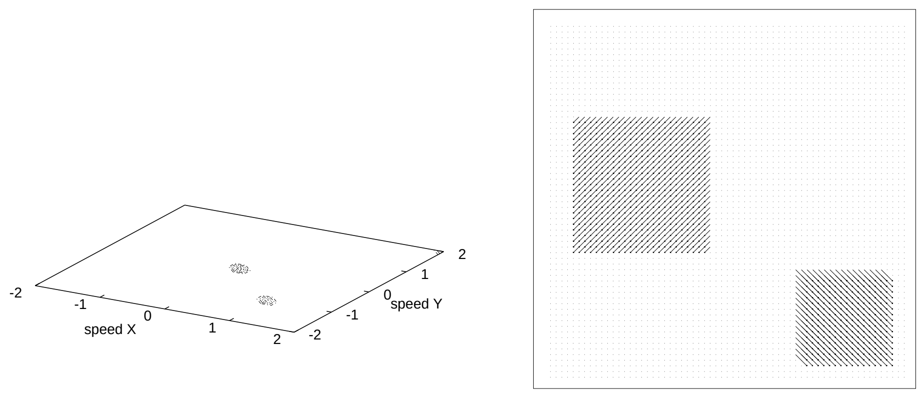
As before, a suitable cost function can be defined by the absolute difference in the luminance values of two corresponding pixels. The correspondence is determined by the motion vector: a pixel in the frame at time t in position (x, y) with associated motion vector (v_x, v_y) corresponds to the pixel in the frame at time $t+1$ in position $(x+v_x, y+v_y)$. Two-dimensional linear interpolation similar to Eq. 7 is used to compute the luminance at sub-pixel positions.

Fig. 4 shows the reclustering technique applied to the problem of motion estimation. The computer-generated test sequence contains three objects: two moving squares and a static background. In the top row the real flow-field used to generate the image sequence and its corresponding density plot is shown. The second row shows the optical flow computed by the algorithm of Lucas and Kanade [16] as implemented in [3], and the plot of the distribution of the flow vectors.

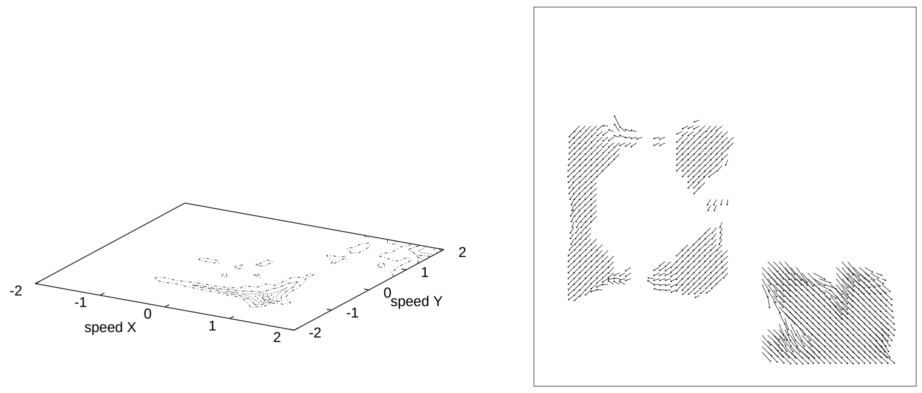
The errors are visible in the flow map and in the distribution of the flow vectors, where the centers of the peaks are surrounded by a platform of outliers. In addition the algorithm was unable to detect the static background, leading to empty regions in the motion map.

The third row contains the results of the application of the reclustering technique to the motion map: the reclustering algorithm was able to fill the holes and to eliminate most of the outliers. The residual outliers are generated in the regions where occlusions or dis-occlusions are taking place and the reclustering algorithm could not find any corresponding pixel.

a



b



c

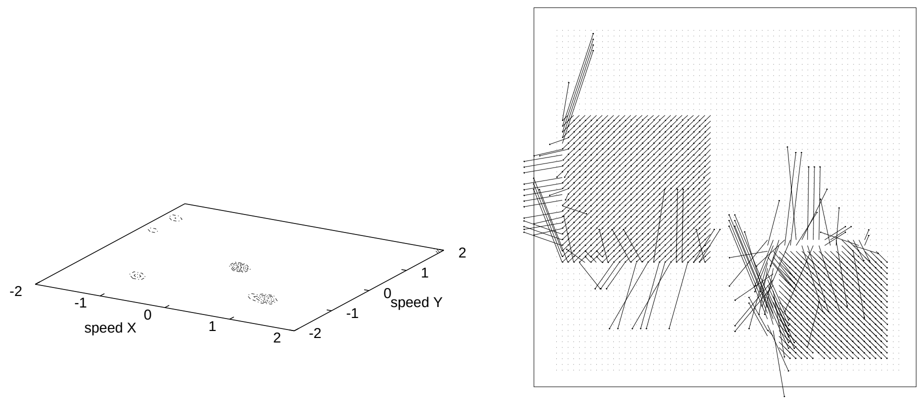


Figure 4: The reclustering technique applied to an optical flow algorithm. In a) is the real optical flow, used to generate the sequence, in b) the optical flow computed by the Lucas and Kanade algorithm, in c) the optical flow after reclustering.

3.3 Noise sensitivity

The cost function we used in the previous examples is very sensitive to noise, being based on the value of single pixels. It is clear that even a small amount of noise will disturb the minimizing procedure, leading the reclustering algorithm to destroy the structure of the feature map instead of improving it. The noise sensitivity of the cost function is a major problem in the treatment of real images since they always contain some amount of noise.

It is possible to overcome the problem by defining a new cost function (Eq. 8) as the sum of the cost defined in Eq. 6 summing over the whole neighborhood, using the same disparity value for every pixel in the neighborhood, that is, assuming that the whole neighborhood share the same disparity.

$$\mathrm{Cost}(x, y, d) = \sum_{i=-W}^{W} \sum_{j=-W}^{W} \mathrm{cost}(x+i, y+j, d) \tag{8}$$

In a simple experiment, we corrupted the original compute-generated stereo pair of Fig. 2 with additive Gaussian noise. With the original version of the reclustering algorithm even a noise level of 5% is enough to introduce gross errors in the improved disparity map. With a cost function defined on a window of 3×3 pixels size, the algorithm can tolerate this noise level quite well and with a window size of 5×5 pixels the tolerable noise level increases to 15% (Fig. 5).

3.4 Synthetic and natural scenes

Using a square window of 3×3 pixels led to satisfactory results for a wide range of natural stereo images. In Fig. 6 we show the results on the well know image of the trees from SRI.

In most of the cases the reclustering algorithm was able to sharpen the contours of the disparity image and improve its overall quality. When the scenes

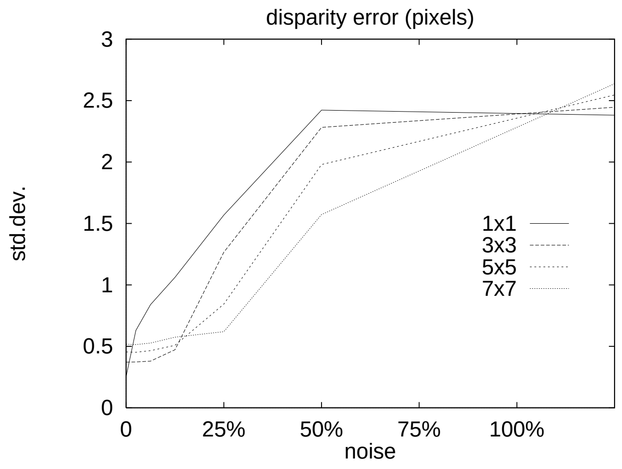


Figure 5: The effect of the window size on the noise sensitivity of the reclustering algorithm

contained tilted surfaces the reclustering approximated them with many fronto-parallel planes (but see below).

3.5 Limitations

If the raw feature map is not composed by a majority of areas with nearly-constant feature values, like flat surfaces in the stereo case, or rigid objects translating with constant velocity in the motion case, the simple-minded projection function used in the previous examples is not adequate and a more sophisticate approach is necessary. For example, the algorithm can be extended to deal with tilted or curved surfaces by a projection function from the disparity feature map to the space of the *disparity gradients.* Assuming the scene to be composed by nearly planar surfaces leads to the required distribution of disparity gradients with peaks surrounded by noise.

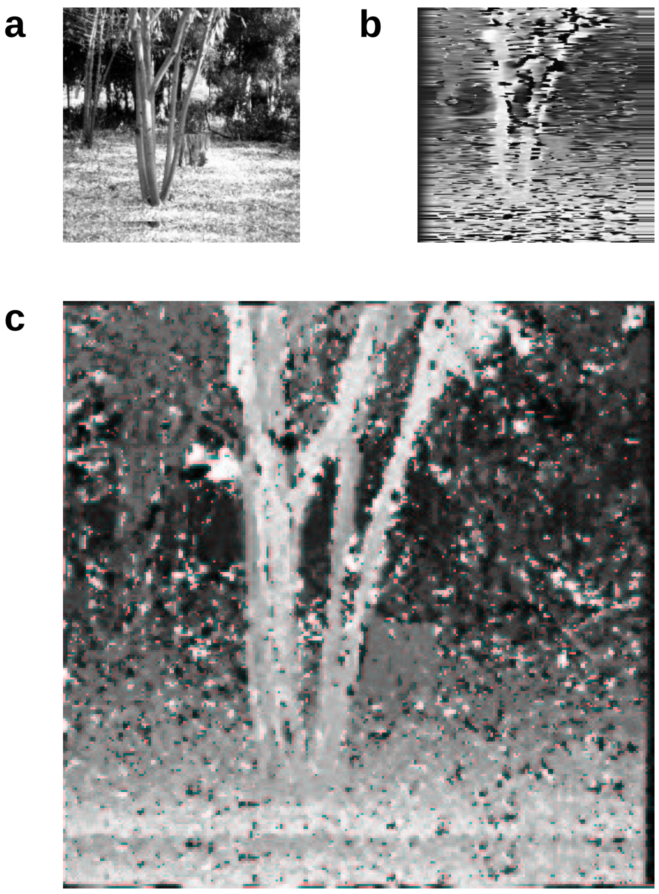


Figure 6: The reclustering applied to a natural scene. In a) is the left image of the stereo pair, b) is the depth map computed by the Fleet et al. algorithm. The Gabor filter had a modulation of 10 pixels and a bandwidth of 0.8 octaves. c) is the result after ten iterations of the reclustering algorithm, window 3×3 pixels, 10 clusters.

3.6 More advanced reclustering schemes

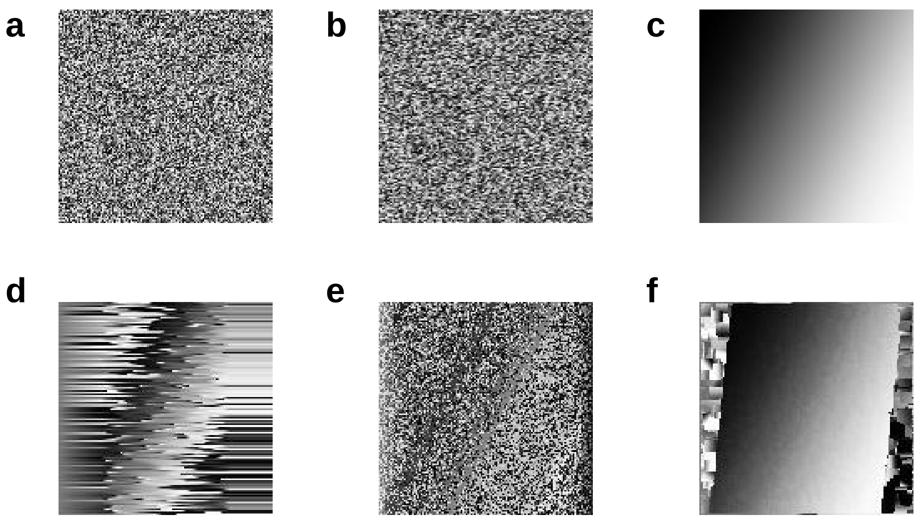
We implemented a more sophisticated version of the reclustering algorithm usign the disparity gradients as clustering space. In this case we use two clustering spaces, $\mathcal{C}_x$ is the set of the possible disparity gradients along the $X-$axis and $\mathcal{C}_y$ is the set of the possible disparity gradients along the $Y-$axis.

The cost function is defined by fitting a planar surface with the estimated gradient on the square neighborhood of each point in the disparity map and using the fitted disparities to compute the usual matching cost (Eq. 6).

The improvement step is implemented by searching independently for the best value of the X and Y gradient. If the cost of the disparities derived by the fitting planar patches lead to a cost reduction the map is changed accordingly.

Three examples of the application of the reclustering algorithm on the gradient are shown in Fig. 6, Fig. 7 and Fig. 8. For the tilted surface in Fig. 7 an excellent result is obtained and the stepwise approximation from the regular reclustering is massively improved. The same is true for the artificial museum scene (Fig. 8). Gradient reclustering works particularly well in these cases because the original images are noise-free.

A fairly advanced application of the reclustering algorithm is presented in Fig. 9. In this experiment a differential stereo algorithm [9, Chapter 16] recovers for each 9×9 pixel neighborhood the parameters of the 3D-plane that best approximates the local structure of the scene. The feature map in this case is the set of the extracted planar patches—a 3D-plane equation and its corresponding covariance matrix. The reclustering algorithm is used to join similar planes and assigning every pixel to the plane that best approximates it. A K-means clustering algorithm with 60 centers was used for the clustering stage in the example shown. As distance function the Mahalanobis distance was used and the center of each



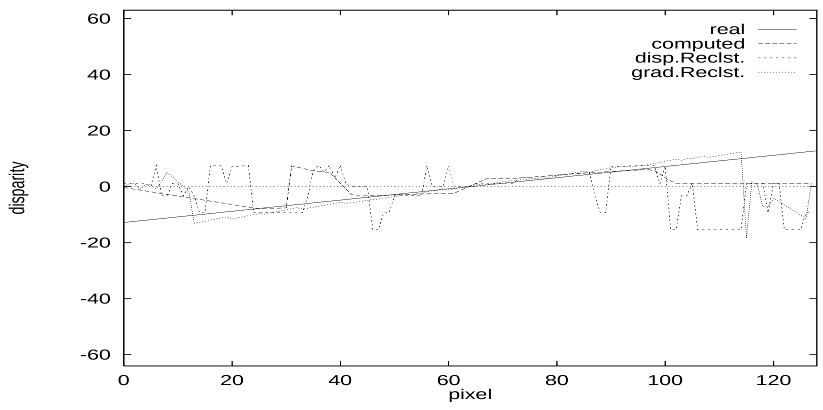


Figure 7: A computer-generated, noise-free stereo pair of a tilted plane covered by random-dot pattern is used to show the improvement of the reclustering algorithm applied to the gradient over the normal reclustering applied to the disparity. In a) and b) is the stereo pair, c) is the real depth map used to generate the scene, d) the depth map computed by the Fleet et al. algorithm. e) is the result after ten iterations of the reclustering algorithm applied on disparity values and in f) the result after ten iteration of the reclustering algorithm applied to the gradient values. The graph (below) shows the disparity values for the central line of the image for the different cases indicated (see labels). The reclustering on the gradient used a window of 5×5 pixels with five centers.

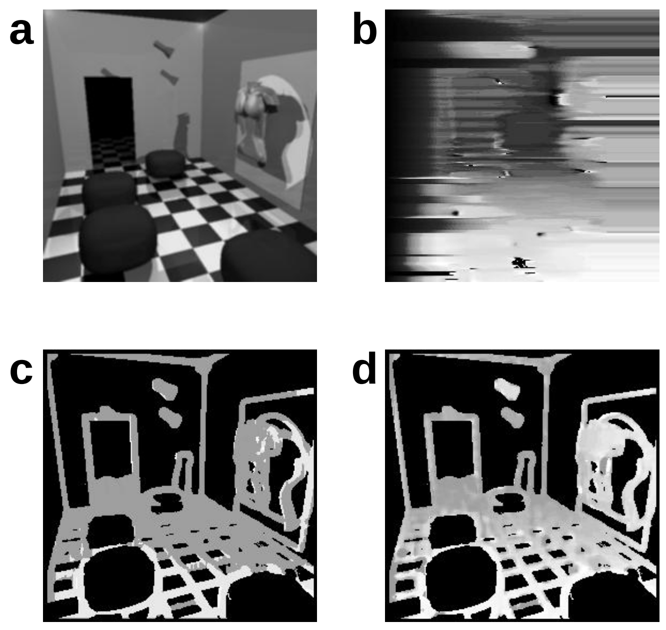


Figure 8: The reclustering on the gradient applied to a computer-generated scene. In a) is the left image of the stereo pair, b) is the depth map computed by the Fleet et el. algorithm. In c) is the result after ten iteration of the reclustering algorithm on the disparity and in d) the result of the reclustering algorithm on the gradient. For both c) and d) the window was 5×5 pixels and ten clusters were used.

cluster was computed by the Kalman batch update equation [18]. In the shown example only 7 out of the 60 centers used got assigned any point. Finally, a post-processing stage merges the centers whose Mahalanobis distance is less than 7.81 (the 95% quantile of the chi-square distribution with 3 degrees of freedom) leading to the shown three clusters. Similar results were obtained using 10, 20 and 40 centers. The error function is the one defined in Eq. 8 with a window size of 9×9 pixels.

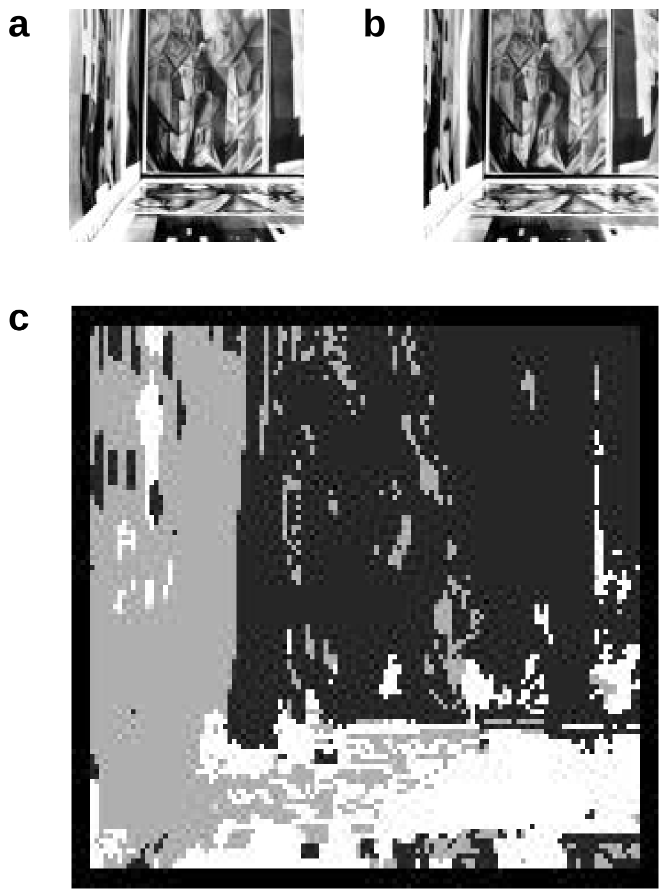


Figure 9: The reclustering algorithm used to segment a natural scene in planar patches. In a) is the left image of the stereo pair, b) is the right image of the stereo pair, in c) is the segmented image, that shows the three resulting clusters.

4 Discussion

A major advantage of the reclustering algorithm is its generality and wide applicability: we presented two examples of its application involving stereoscopy and motion analysis using straight forward reclustering, but it is possible to adapt it to a wide range of techniques of early vision with minor adaptations.

Another favorable characteristic is its efficiency. The computational cost of the algorithm is small and proportional to the number of clusters produced by the first step plus that of the second step which is proportional to the number of the clusters multiplied by the number of the points in the feature map. On the other hand, the generality of our approach limits its performance as compared to other more specialized techniques.

The strategy of the reclustering algorithm can be interpreted in another perspective. Consider for example the stereo case. The problem is to find the mapping from the pixels in the left image to the pixels in the right image. Since an exhaustive search across all possible mappings is infeasible we take the approach to compute an approximation (with the stereo algorithm) and then *use the original input data* (the luminance image) to check the approximation and to perform a limited search over the "interesting" points (the centers of the clusters) of the problem space. The algorithm is able to improve the feature map independently from the stereo algorithm for the very reason that it uses additional and independent information obtained from the original stereo pair.

The optimal case for the reclustering algorithm is that of minimizing a multidimensional cost function with multiple minima. In this case the dimension of the search space increases rapidly making any systematic minimization strategy infeasible, and the existence of multiple minima renders gradient-descent techniques ineffective. The reclustering approach is applicable if we are able to compute a first-order solution with a distribution that is close to that of the optimal solution—in the sense that their peaks coincide. In this situation the reclustering algorithm can use the distribution of the points in the $\mathcal{C}$ space to explore the feature space $\mathcal{F}$ for more advantageous solutions. In our examples this condition holds: the disparity maps are affected by the windowing effects, leading to blurred borders between regions with different disparities, but the global distribution of the disparity values and the positions of its peaks are basically unaffected.

The algorithm converges rapidly to a stable state. In Fig. 10 we show the graph of the mean cost decrease, i.e., the difference in the cost function before and after a reclustering step divided by the number of pixels that have been reclustered. The test image used is the tree in Fig. 6a. As can be seen the convergence is very fast, and after 4 to 5 iterations there are very few changes in the feature map.

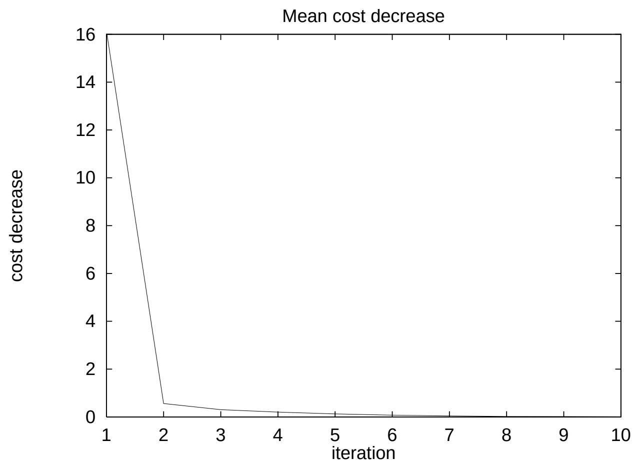


Figure 10: The total cost decrease divided by the number of pixel reclustered.

4.1 Similar approaches

The reclustering algorithm is equivalent to performing segmentation on the feature map and then applying an updating procedure to each pixel in a segment. There exists a vast amount of literature on segmentation techniques in computer vision (see for example [22, 10]), also the application of clustering algorithms to segmentation has been often used and reported in literature [5, 13]. The first step of our reclustering algorithm is actually an implementation of the *measurement space clustering* segmentation technique (see [10]).

Wang et al. [24] apply a segmentation technique to decompose a video sequence in different, independently moving layers. To this end, they first compute the optical flow for the whole image, and then use the K-means clustering algorithm to subdivide the optical flow map into those layers. Afterwards the optical flow field is computed again separately on each layer and recomposed leading to an improved quality of the total optical flow map. A similar but more sophisticated approach is used by Black and Jepson [4] to fit a parametric model of the optical flow on segmented images.

The updating procedure of our reclustering is based on the statistical properties of the feature map and as such it is also related to the large field of stochastic minimization techniques. The main difference is that the update procedure used by the reclustering algorithm is deterministic and uses only the peaks of the distribution.

An interesting connection can also be drawn with the robust statistics [12] and the outliers rejection techniques. In each cluster the value of the center depends only on the values of the features in the cluster. From the point of view of the computation of the center position the features belonging to the other clusters are treated as outliers and rejected, making the center position a simple kind of robust statistic.

References

[1] P. Anandan. A computational framework and an algorithm for the measurement of visual motion. *International Journal of Computer Vision*, 2:283—310, 1989.

[2] S. Barnard and M. Fischler. Computational stereo. *ACM Comput. Surveys*, 14(4):553–572, Dec. 1982.

[3] J.L Barron, D. Fleet, and S. Beauchemin. Performance of optical flow techniques. *International Journal of Computer Vision*, 12(1):43–77, Jan. 1994. The C source code is available on the FTP server ftp://csd.uwo.ca/pub/vision.

[4] M.J. Black and A.D. Jepson. Estimating optical-flow in segmented images using variable-order parametric models with local deformations. *PAMI*, 18(10):972–986, October 1996.

[5] G. Coleman and H.C. Andrews. Image segmentation and clustering. *PIEEE*, 67(5):773–785, May 1979.

[6] U.R. Dhond and J.K. Aggarwal. Structure from stereo: A review. *SMC*, 19(6):1489–1510, November 1989.

[7] D. Fleet, A. Jepson, and M. Jenkin. Phase-based disparity measurement. *Computer Vision, Graphic and Image Processing*, 53(2):198–210, March 1991.

[8] S. Ghosal and P. Vanek. A fast scalable algorithm for discontinuous optical-flow estimation. *PAMI*, 18(2):181–194, February 1996.

[9] R. M. Haralick and L. G. Shapiro. *Computer and Robot Vision*, volume 2. Addison Wesley, 1992.

[10] R.M. Haralick and L.G. Shapiro. Image segmentation techniques. *CVGIP*, 29(1):100–132, January 1985.

[11] B.K.P. Horn and B.G. Schunck. Determining optical flow. *AI*, 17:185–203, 1981.

[12] P. J. Huber. *Robust Statistics.* John Wiley & Sons, Inc., 1981.

[13] J.M. Jolion, P. Meer, and S. Bataouche. Robust clustering with applications in computer vision. *PAMI*, 13(8):791–802, August 1991.

[14] S.Z. Li. *Markov Random Field Modeling in Computer Vision.* Springer-Verlag, 1995. ISBN 0-387-70145-1.

[15] B.D. Lucas and T. Kanade. An iterative image registration technique with an application to stereo vision. In *DARPA81*, pages 121–130, 1981.

[16] B.D. Lucas and T. Kanade. Optical navigation by the method of differences. In *DARPA84*, pages 272–281, 1984.

[17] D. Marr. *Vision.* Freeman, New York, 1982.

[18] Peter S. Maybeck. *Stochastic Models, Estimation, and Control*, volume 1. Academic Press, New York, 1979.

[19] H.H. Nagel. On the estimation of optical flow: Relations between different approaches and some new results. *AI*, 33(3):299–324, November 1987.

[20] H.H. Nagel. Optical-flow estimation and the interaction between measurement errors at adjacent pixel positions. *IJCV*, 15(3):271–288, July 1995.

[21] T.A. Poggio, V. Torre, and C. Koch. Computational vision and regularization theory. *Nature*, 317:314–319, 1985.

[22] A. Rosenfeld and L.S. Davis. Image segmentation and image models. *PIEEE*, 67:764–772, May 1979.

[23] Richard Szeliski and James Coughlan. Spline-based image registration. Technical Report CRL 91/1, DEC Cambridge Research Laboratory, April 1994. available on-line.

[24] J. Y. Wang, E. H. Adelson, and U. Desai. Applying mid-level vision techniques for video data compression and manipulation. In *Proceedings of SPIE: Digital Video Compression on Personal Computers: Algorithms and Technologies*, San Jose, February 1994. SPIE.