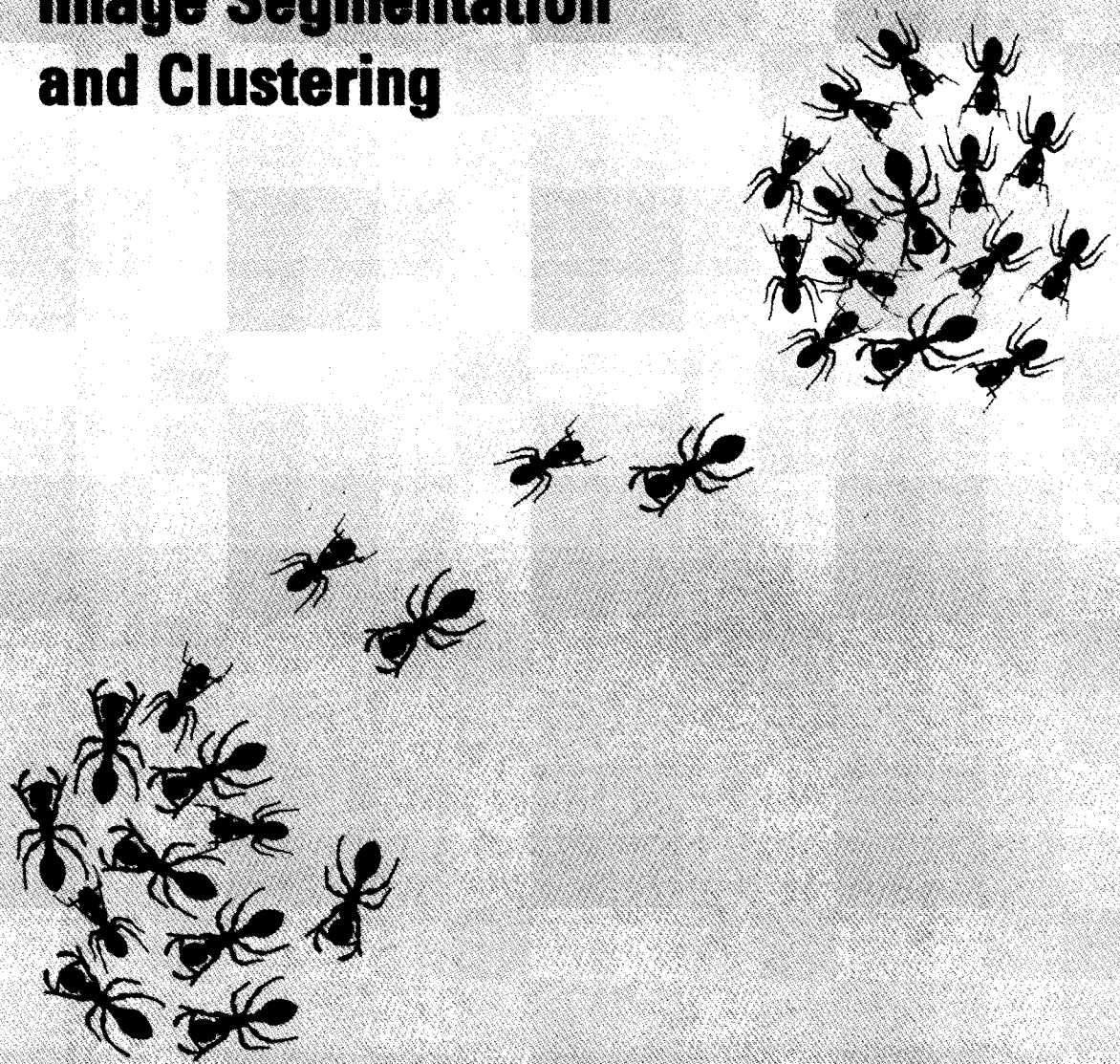
Motion Correspondence Image Segmentation and Clustering



Modeling and Optimization Aspects

Cor J. Veenman

Motion Correspondence, Image Segmentation, and Clustering

Modeling and Optimization Aspects

Proefschrift

ter verkrijging van de graad van doctor
aan de Technische Universiteit Delft,
op gezag van de Rector Magnificus prof.dr.ir. J.T. Fokkema,
voorzitter van het College voor Promoties,
in het openbaar te verdedigen op maandag 28 januari 2002 om 10.30 uur
door Cornelis Johannes VEENMAN
doctorandus in de informatica
geboren te Ankeveen.

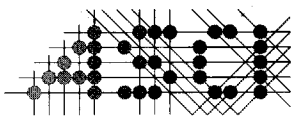
Dit proefschrift is goedgekeurd door de promotor:
Prof.dr.ir. E. Backer

Toegevoegd promotor:
Dr.ir. M.J.T. Reinders

Samenstelling promotiecommissie:

Rector Magnificus,	voorzitter
Prof.dr.ir. E. Backer,	Technische Univeristeit Delft, promotor
Dr.ir. M.J.T. Reinders,	Technische Universiteit Delft, toegevoegd promotor
Prof.dr. A.E. Eiben,	Vrije Universiteit Amsterdam
Prof.dr.ir. F.C.A. Groen,	Universiteit van Amsterdam
Prof. A.K. Jain,	Michigan State University, USA
Prof.dr. C. Roos,	Universiteit Leiden
Dr.ir. R.P.W. Duin,	Technische Universiteit Delft

Dit promotieonderzoek heeft gedeeltelijk plaatsgevonden in het STW project FASE (Facial Analysis and Synthesis of Expressions).



This work was carried out in the graduate school ASCI.
ASCI dissertations series number 74.

The cover image shows two ant species; the larger one is the Leptothorax acervorum (Fabricius), the smaller one is the Camponotus (Camponotus) japonicus Mayr.

ISBN 90-9015459-0

Printed by Universal Press, Veenendaal

Motion Correspondence, Image Segmentation, and Clustering

Modeling and Optimization Aspects

To my parents

Contents

Chapter 1

Introduction

This thesis contains a collection of papers concerning subfields of computer vision and machine learning.

First, we deal with the motion correspondence problem, which applies to various tasks in computer vision. By resolving motion correspondence, one can track small objects in a video sequence. After detection of the objects, the motion-based point correspondences help in identifying the objects in a sequence of video images. Another typical motion correspondence application concernes the recovery of the shape of a structured object where the tracked points represent features on the object.

Next, we discuss the image segmentation problem. The main goal of image segmentation is to reduce the number of image parts that has to be explained in terms of identity, shape, position, and dynamic behavior. To this end, the image is divided into regions that are as large as possible, but always cover a single object or a part thereof. Image segmentation applies among others to computer vision and image compression.

Finally, we address the clustering problem. Clustering is often applied when data are explored in order to retrieve information about the objects or phenomena from which the data are acquired. The goal is to find groups or clusters of objects that show strong resemblance based on the measured object data. In some applications, prototypes of the resulting clusters are determined that can serve as cluster representatives.

This chapter is an attempt to grasp the fundamental aspects that the three problems within this thesis have in common. All three problems can be viewed as pattern recognition problems, whether it concerns motion patterns, homogeneous image regions, or homogeneous data clusters in general. However, the resemblance is stronger than that. That is, for all three problems there are no examples given as to how the structure in the data is manifested. Because of the lack of this type of knowledge, these problems are considered unsupervised pattern recognition problems.

In Section 1.1, we first make a more clear distinction between supervised and unsupervised pattern recognition. Since we modeled all problems as optimization problems, we focus on modeling and model optimization issues for both types of pattern recognition problems

in Section 1.2. It appears that when modeling pattern recognition problems as optimization problems, one inevitably has to deal with conflicting optimization criteria. In Section 1.3 we therefore introduce essential terminology and solution methods from the field of *multi-criterion* optimization or *multi-objective* optimization. In Section 1.4, we elaborate on specific multi-criterion-oriented modeling issues of clustering as a typical unsupervised pattern recognition problem that facilitates the parameter setting as well as the interpretation of the results.

In Section 1.5, we show how the multi-criterion modeling aspects are manifested in the three pattern recognition problems that form the basis of this thesis. In particular, we indicate the conflicting criteria that arise in each of the problem domains. In addition, we show how the solution strategies adopted in each of the domains relates to common solution methods in multi-criterion optimization. It should be mentioned that, since this chapter is written in retrospect, in some cases this relationship is not strict.

Finally, in Section 1.6, we draw some conclusions about multi-criterion modeling and optimization of unsupervised pattern recognition problems. Specifically, we propose extensions for the three pattern recognition problems regarding the improvement of the models and the automatic estimation of their parameters.

1.1 Supervised and Unsupervised Pattern Recognition

For the sake of clarity, we start by giving the pattern recognition terminology as we use it. In pattern recognition, a set of data vectors is given that is obtained by a measuring and feature extraction process. The task is to discover structure in the data vectors by finding a *function* that maps the data vectors to continuous, discrete, or symbolic outputs, which we call *classes* [5], [15], [16], [29].

In pattern recognition, a major distinction is made with respect to the way the mapping function is determined. That is, the mapping function can be learned with or without the help of a supervisor, where the supervisor gives the corresponding classes for a set of data vectors. In case these mapping examples are given, the problem is called *supervised pattern recognition*, which includes regression and classification among others. In regression the outputs are usually continuous values, while in classification these are symbolic class labels. The goal of supervised pattern recognition is therefore to organize the knowledge of the supervisor in such a way that new data vectors can be properly recognized as classes known by the supervisor.

Without a supervisor, the problem is called *unsupervised pattern recognition*. The best-known unsupervised pattern recognition problem is the *clustering* problem, but also other problems can be classified as such, like the already mentioned *image segmentation* problem and the *motion correspondence* problem. Without the prior knowledge of the supervisor, one has to obtain the knowledge of the structure in the data otherwise. However, due to the lack of a supervisor the problem becomes almost unsolvable: at most the relative structure in the input data vectors can be found. Similar issues are involved as in supervised pattern recognition, like the extraction of features from measurements and the selection of discriminating

features. Other issues are more relevant in the unsupervised case, like the definition of proper distance measures between data vectors, and the selection of a proper scale or level of detail. Decisions about these issues cannot be made without some form of feedback. For that reason, supervision is imperative to a certain extent also for unsupervised pattern recognition.

1.2 Modeling and Optimization in Pattern Recognition

In this section, we unravel some modeling and optimization issues involved in pattern recognition problems. With respect to this, we only consider the given data set and no underlying distributions or possible re-sampling from an infinite data pool, i.e. we use one data set and that is all we have.

Both for supervised and unsupervised pattern recognition problems a function must be determined that maps data vectors to classes. In supervised pattern recognition we try to learn the function such that it optimally maps the input data vectors to the output classes given by the expert. Accordingly, we try to minimize the *error* made by the mapping function. In classification, the error is the number of misclassified data vectors, while in regression, the error is the average or total distance between the predicted class value and the class value given by the expert.

Clearly, for a given data set it is always possible to find a perfectly fitting mapping function, unless there are inconsistencies in the given examples. In real life, however, such results are not useful. For this reason, we no longer consider a perfect mapping function, but we rather aim for a *model* that suits the problem we want to solve.

For instance, for the classification problem, usually some misclassifications are preferred over an overfitted (overspecialized) model, because it is assumed that there are some measurement errors and supervision errors (outliers), and the number of data vectors can be too low to fit a significant model (undersampling). Overfitting is caused by a too strict minimization of the error and can be prevented by constraining the complexity of the model either during the design or by introducing a complexity minimization criterion in the model. As an example of the latter, when a classifier is based on a mixture of Gaussians model, the number of Gaussians can be minimized or the size of the Gaussians can be maximized. Accordingly, an appropriate classification model contains an error minimization criterion to achieve *specialization* and a complexity minimization criterion to achieve *generalization*.

When the model is defined in such a way, a trade-off between both criteria must be found when learning the model with the given examples. In order to validate the learned model the input data vectors are usually divided up into a training set and a test set. Then, the model is optimized with the training data and validated with the test data. However, in this way it is not possible to learn a proper trade-off, since this procedure only gives a performance measure (test error) afterwards. The trade-off can be found by iteratively optimizing the model such that the test error is minimal. However, minimizing the error in the test set by weighing specialization and generalization against each other in the training set leads to another optimization problem. This procedure would require an extra validation set to come up with a reliable performance measure. We note that in classification the term generalization

is also used to qualify a 'good' trade-off between both criteria leading to a small test error, that is, a low number of misclassified data vectors in the test set. Clearly, establishing the right trade-off between specialization and generalization is a very delicate matter. In effect this trade-off should be generalized over all possible (unavailable) data sets instead of one single small data set, which leads to another difficulty, known as the bias-variance dilemma. We do not consider this type of error minimization and refer to standard textbooks for additional information on this issue [5], [29].

In the unsupervised case, a suitable model also includes conflicting generalization and specialization criteria. If we for instance consider the clustering problem, the generalization criteria generally aim at making the clusters larger and the cluster borders smoother, while specialization criteria make clusters smaller and their borders irregular. However, maximization of a specialization criterion leads to as many clusters as data vectors and maximization of a generalization criterion leads to one cluster containing all data vectors. In general, there is no way to automatically find the right trade-off between both criteria, because no examples are given, i.e. the supervision is lacking. Where in classification eventually overfitting leads to sensitivity to outliers and to undersampling, in clustering overspecialization leads to no significant structure at all.

In the remainder we focus on unsupervised pattern recognition issues. Especially, we consider the conflicts between generalization and specialization criteria that are the most problematic in unsupervised pattern recognition problems. To this end, we first overview some relevant aspects and methods for solving multi-criterion optimization problems.

1.3 Multi-Criterion Optimization

In multi-criterion optimization problems, multiple minimization and maximization criteria are defined over the same variables. Since these criteria are generally dependent, an improvement in one criterion may lead to a deterioration decrease in another. Accordingly, in multi-criterion optimization problems, it is usually impossible to deliver objectively good answers since there is a large set of mathematically equivalent optimal solutions called the Pareto optimal set or Pareto front. The Pareto optimal set consists of those solutions (vectors in criterion space) of which no criterion can be improved without degrading another one, see Fig. 1.1. In the figure there are two criteria $F_1(x)$ and $F_2(x)$ defined over variable x having a circular solution space or *feasible region*, where both $F_1(x)$ and $F_2(x)$ are minimization criteria. When minimized independently, $F_1(x)$ and $F_2(x)$ result in $F_1^*(x)$ and $F_2^*(x)$, respectively. The curved thick line in the figure represents the Pareto set of this problem, i.e. the solutions on this curve cannot be improved with respect to one criterion without decreasing the other. Since it is not possible to decide on the optimal solution from this set mathematically, a decision maker is needed to select an element from the Pareto set as a preferred solution.

There is a number of ways to solve multi-criterion optimization problems, and hence, to select the desired solution from the Pareto set. Almost all multi-criterion solutions methods convert the multi-criterion optimization problem into a single-criterion optimization problem,

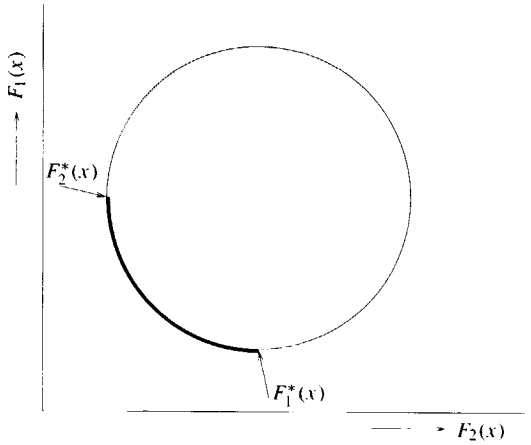


Figure 1.1: Illustration of the Pareto set in a bicriterion optimization problem with a circular feasible region. The points $F_1^*(x)$ and $F_2^*(x)$ are respectively obtained by optimizing $F_1(x)$ and $F_2(x)$ individually. The thick curved line connecting both points represents the Pareto set or Pareto front.

such that one solution from the Pareto set is found. For the optimization of the corresponding single-criterion optimization problem any of the known solution methods can be applied [2], [7], [8], [11], [18], [24], [31] besides specially designed single-criterion optimization algorithms. Additionally, with some of these methods it is possible to determine (or estimate) the Pareto set or parts thereof by varying the parameters of the method.

Multi-criterion optimization solution methods can be classified based on the role that the decision maker plays [13]. In the least involved cases, the decision maker may only accept or reject the solution (no-preference methods). Otherwise, the decision maker can articulate his preference beforehand (a priori methods), select the preferred solution from the computed Pareto set (a posteriori methods), or he can progressively explore the Pareto set (interactive methods).

From the methods that involve the decision maker, the a posteriori methods are computationally the most costly, since the whole Pareto set has to be found. The a priori methods, on the other hand, are the most efficient. With these methods the desired solution can be found directly, since a priori knowledge is used to estimate the parameters of the corresponding single-criterion problem. Clearly, these methods require that the parameters of the single-criterion solution method correspond to meaningful aspects of the problem domain.

Here, we shortly describe some typical methods to solve multi-criterion optimization problems. For a more complete and elaborate description we refer to [20], [21], [28].

Weighted Sum Method

The most widely used method for converting a multi-criterion optimization problem into a single-criterion optimization problem is weighting the multiple criteria into a single criterion. That is, all criteria are multiplied with a proper weighting coefficient and summed up into one criterion function. This method is almost automatically used for facing a multi-criterion optimization problem, though without mentioning the underlying conflicting criteria or alternative solution methods are not mentioned.

This method has a number of drawbacks. Since there is no relation between the weights and the obtained solution, it is difficult to determine the right weighting coefficients. For a proper use, the method should be applied repeatedly in order to explore the Pareto set (as in interactive and a posteriori methods), see Fig. 1.2(a). It can, however, be difficult to sample the Pareto set at regular distances by varying the weighting coefficients. A more serious problem is that parts of the Pareto set *cannot* be found when its elements lie on a non-convex shaped surface, see Fig. 1.2(b).

ε-Constraint Method

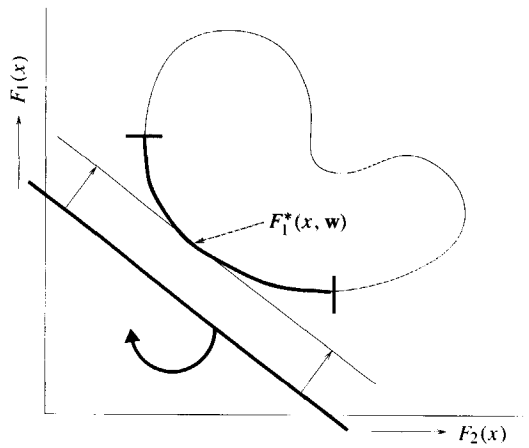
Another basic method that can be used to find optimal solutions is the ε-constraint method [9]. According to this method all criteria except one are converted into hard constraints. These constraints give an upper limit in case the original criterion has to be *minimized* or a lower limit if the original criterion has to be *maximized*. Like the weighting method, the ε-constraint method can be used to determine the Pareto front. That is, by varying the values of the constraints and by solving the single-criterion optimization problem for every constraint setting samples can be taken on the front at regular distances, see Fig. 1.3. Though this method can be computationally intensive, the Pareto front can indeed be found, whether it is convex or not.

Hierarchical or Ranking Method

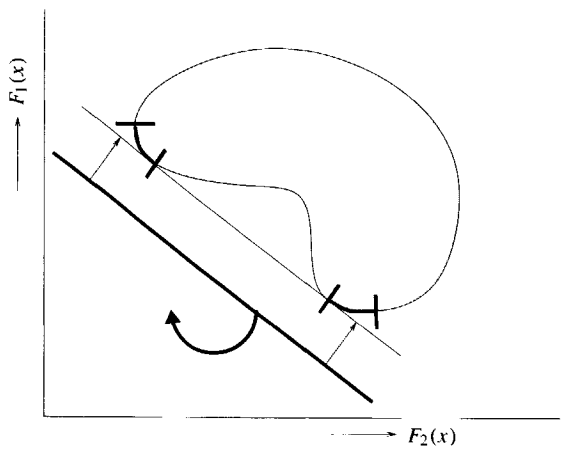
In case the criteria can clearly be ordered in terms of their importance then the hierarchical or ranking method may be appropriate. The criteria are optimized in order of importance, while per criterion optimization step a constraint is added to fix the range of the optimal criterion value of the previous step. That is, the previous criterion may only worsen to a certain extent, see Fig. 1.4. This method is clearly an a priori method, though the decision maker can tune the constraint range ratio parameter δ.

Value or Utility Functions

Sometimes it is possible to define a function on the multiple criteria that shows how the decision maker would value each criterion vector. Accordingly, with such a *value function* or *utility function* method the multi-criterion optimization problem converts into a true single-criterion optimization problem. It is, however, generally very difficult to define a proper



(a)



(b)

Figure 1.2: In (a) the weighting sum method is applied to a bicriterion problem with a convex Pareto front. In (b) the same method is applied to a problem with a non-convex Pareto front. Accordingly, parts of the Pareto front cannot be found by tuning the weight vector. In the figures, the curved thick lines represent the Pareto front or parts thereof (in case the Pareto set is non-convex). The straight thick lines represent the setting of the weights, where the slope of the lines is $atan(w_1/w_2)$ and w_1 and w_2 are the weighting coefficients of $F_1(x)$ and $F_2(x)$ respectively.

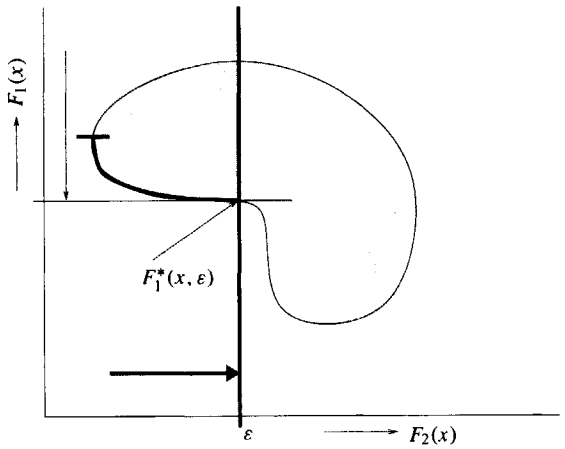


Figure 1.3: Illustration of the estimation of a non-convex Pareto front with the ε-constraint method. The point $F_1^*(x)$ on the Pareto front is the result of minimizing $F_1(x)$ subject to $F_2(x) \leq \varepsilon$. In the figure, the curved thick line represents the part of the Pareto front that is found when ε is step-wise incremented from zero to the position of the straight thick line.

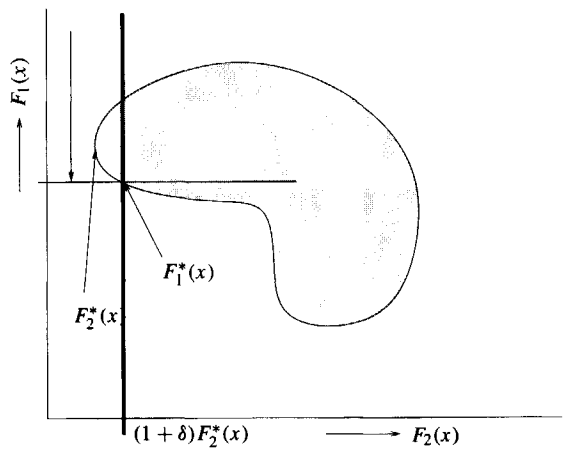


Figure 1.4: Illustration of the optimization of a bicriterion optimization problem using the ranking method. After optimization of $F_2(x)$, resulting in $F_2^*(x)$, the next criterion $F_1(x)$ is optimized subject to $F_2(x) \leq (1+\delta)F_2^*(x)$, which results in $F_1^*(x)$.

value function, though it is assumed that implicitly the decision maker makes use of such a function.

1.4 Clustering as a Multi-Criterion Optimization Problem

In order to elaborate on multi-criterion optimization aspects of unsupervised pattern recognition problems, we focus on clustering as exemplary problem in this section. We already mentioned that for clustering some form of supervision is needed, among others to define proper distance measures between the data vectors and to define a proper level of detail. Hence, clustering is usually a stage of an interactive process called *exploratory data analysis*.

Exploratory data analysis contains all (interdependent) stages, from finding proper measurements, to extracting proper features, and to selecting a proper clustering. The clustering part consists of designing the cluster model and optimization algorithm, estimating the parameters of the method, and delivering a corresponding data partitioning, where the model parameters are related to a trade-off between the generalization and specialization criteria. Sometimes the *data analyst* is able to give a good estimate of these cluster method parameters. For instance, the number of clusters is quite often assumed to be known. If so, the results of one run are assumed to be suitable. Otherwise the analyst evaluates the resulting partitioning with specific validation functions or statistical tests in the so-called cluster validation phase. If the analyst disqualifies the partitioning, the cluster parameters are modified and the data vectors are re-clustered. A major difficulty with this approach is that re-parameterisation is far from trivial. That is, the means to check the cluster validity are in general totally different from the cluster algorithm, so the validity test results have no direct relation to the parameters of the cluster algorithm. Another way to check the validity is to run the cluster algorithm over a complete range of parameter settings and to validate the range of results with a validation function, like the Davies Bouldin Index [4]. With such a method the analyst attempts to discover trends in the validation function curve in order to find proper parameter settings and the corresponding cluster results.

If we consider exploratory data analysis, the correspondence to multi-criterion optimization becomes apparent, since the clustering stage inevitably results in at least a bicriterion optimization problem, being a generalization and a specialization criterion. Therefore, also in exploratory data analysis the analyst is involved in finding a proper trade-off between the conflicting criteria. We list a number of issues that should be considered in the design of a clustering model and the corresponding optimization algorithm. Especially the interpretation of the optimization results and the estimation of the model parameters are taken into account.

First, however, we propose a generalization of the popular K-means model that will be used to serve as an example in explaining these issues. In the normal K-means model the number of clusters is assumed to be known. Further, the specialization criterion (F_1) is defined as the minimization of the squared distance between all data vectors and their corresponding cluster means [19]. In the generalized K-means model, we do not fix the number of clusters, but we *minimize* it instead. The minimization of the number of clusters serves as a generalization criterion (F_2).

Criteria

If the analyst is to give feedback in the clustering stage, the criteria must refer to meaningful aspects of the problem domain, i.e. they must be *qualitatively significant*. For instance in the generalized K-means model, the squared distance and certainly the number of clusters can be meaningful in the problem domain.

Parameters

In addition to the criteria, the model parameters should be meaningful in the problem domain, that is, they should be *quantitatively significant*. Therefore, certain multi-criterion solution methods are preferred over others. For instance, in case the model parameters are weighting coefficients, it is difficult for the analyst to select proper values. Otherwise, for instance if the ε-constraint method is used the (constraint) parameters can be meaningful if the corresponding criteria are qualitatively significant. In the just proposed generalized K-means model, the quantitative significance of squared distance is questionable but the number of clusters can clearly be quantitatively meaningful.

Inspection

Finally, if the number of criteria, the size of the Pareto set, and the optimization algorithm efficiency allow for it, then it can be preferable for the supervisor to inspect the Pareto set. Then, from these solutions he can select a proper one. Both the shape of the Pareto front and the corresponding solutions can be analysed.

In case the number of clusters is not known, the normal K-means model is typically optimized for various values of K (number of clusters). Then, the solutions are validated with specific validation functions [1], [4], [6], [12]. This is equivalent to solving the generalized K-means model using the ε-constraint method, where the ε constraint is imposed on the number of clusters, i.e. $K \leq K_{max}$. Since the minimization of the squared distance criterion always results in as many clusters as possible, the solutions will consist of exactly K_{max} clusters. The only difference is that with the normal K-means method the trends are analyzed in the shape of specific validation functions instead of the shape of the Pareto front.

1.5 Applications

In this section we elaborate on the optimization criteria involved in the unsupervised pattern recognition problems that we deal with in the remainder of this thesis. Further we attempt to relate the implemented model and optimization method to the known solution methods for multi-criterion optimization problems. Below, we first consider the motion correspondence problem. Then, in the following section we describe the modeling and optimization issues of the comparable image segmentation and clustering problem.

1.5.1 Motion Correspondence

Motion correspondence has to be established when points are detected in a video sequence while there is no *reliable* or *significant* appearance information about the corresponding image features [3], [23], [25], [26]. Then, only positional information is used to link the points in time. The main causes for unreliable and insignificant color information are changing light conditions, poor recording conditions, and restricted recording devices or media. Additional problems are temporarily occlusion, false detections, missing detections, and points that enter or leave the video scene. The tracking of image features with or without appearance information is an important task in computer vision, for instance to track small objects, to perform gesture analysis or to recover object structure from feature-point motion, e.g. [27].

Denoted as an unsupervised pattern recognition problem, the motion correspondence problem has the following characteristics: the number of feature points is unknown, the point identity is unknown, the number of time instances during which a point is present is unknown, and an unknown number of false detections is present.

We start with a restricted definition of the motion correspondence problem, where the number of feature points is constant. Defined in this way the motion correspondence problem can be modeled as a bicriterion optimization problem. That is, the overall smoothness of the point tracks must be *maximized* (F_1) and the number of outliers, that is, false and noisy measurements, must be minimized. The second criterion is implemented by minimizing the maximum deviation from local track smoothness (F_2). Additionally, we impose a uniqueness constraint that states that point measurements are assigned to at most one point track and that a point track contains at most one point measurement per time instance. The first criterion can be considered a specialization criterion and the second a generalization criterion. That is, when tracks are only optimized for smoothness and all *local* deviations from smoothness are allowed, the tracks are specifically fit to the given data set. On the other hand, when the maximum local deviation from smoothness is minimized, many measurements will be labeled false. Consequently, the motion tracks rather contain easily fitting measurements than the outlying measurements.

In general an approach where the analyst has to select the optimal solution for the resulting bicriterion optimization problem interactively is undesirable. Especially because we aim to track high numbers of points in dense scenes and long sequences, it is practically impossible.

In the proposed model, we assume that we have a good estimate of the upper limit of the maximum deviation from local smoothness. In effect, we applied the ε-constraint method without mentioning it in the corresponding chapters. We used a fixed constraint value that represents the maximum deviation from smoothness ϕ_{max}. Accordingly, the model is optimized for smoothness (F_1) subject to $F_2 \leq \phi_{max}$. At this stage, because of the uniqueness constraint, additional measurements can be classified as outliers or false in case the number of measurements was higher than the number of point tracks.

In Chapter 2, we optimize the track smoothness (F_1) only between two time instances, which we call a greedy solution. This optimization problem can be solved efficiently using linear optimization techniques. Besides, we applied a similar model and optimization scheme

to the problem of encoding audio and speech signals with sinusoids, though that subject is not covered in this thesis [17].

In Chapter 3, we enhanced the optimization scheme by extending the temporal optimization scope of the track smoothness criterion. We solved this problem that is known to be intractable with a heuristic search algorithm. The algorithm uses a best first heuristic per recursion level and uses additional constraints to prune the search tree.

In Chapter 4, we attack the general motion correspondence problem, that is, we allow the number of points to vary in time. When only the previous two criteria F_1 and F_2 are optimized, a solution with many very short point tracks is optimal. Accordingly, we add a third criterion F_3 that aims at the *minimization* of the number of tracks. Moreover, to prevent false measurements to be considered (or linked) as short tracks, we also add a number of continuity constraints. These constraints demand that in valid tracks a limited number of consecutive measurements may be missing and that a certain minimal number of consecutive measurements must be present. In the optimization algorithm for the model, the added criterion F_3 has priority over the other two. The resulting hierarchical optimization scheme is specially designed and does not resemble the hierarchical or ranking method that has been described in Section 1.3.

1.5.2 Image Segmentation and Clustering

In Chapter 5, we address the image segmentation problem. In this problem, the task is to find an unknown number of image regions or segments that are each as homogeneous as possible. Further, the union of *adjacent* segments should be heterogeneous [10], [14], [22]. The described work is an extension of previously published results [30]. The image segmentation problem is very similar to the clustering problem that is dealt with in Chapter 6, [5], [15], [29]. The main difference is that in the image segmentation problem, spatial connectivity has to be imposed on the segments. Therefore, the problem is also called spatial clustering. Both problems are clearly unsupervised pattern recognition problems, since the ground truth (symbolic class labels) is lacking. The conflicting criteria that have to be optimized are homogeneity of individual clusters (segments) and heterogeneity of the union of clusters. In Fig. 1.5, we illustrate this conflict. In the figure, there are two groups of three smaller clusters. This data set can be considered as either two clusters (the dashed circles) or six clusters (the solid circles), where the two clusters are more general and the six clusters are more specialized.

The unsupervised pattern recognition problem aspects of the clustering and image segmentation problem are the following: the required densities in data or feature space are unknown, the number of clusters is unknown, the shapes of the clusters is unknown, and an unknown number of outliers is present among the data vectors.

Since the image segmentation problem is similar to the clustering problem, the models that we propose for both problems are similar too. To solve the clustering problem a proper trade-off must be found between *maximization* of cluster homogeneity and *maximization* of the heterogeneity of the union of two clusters. The homogeneity maximization is expressed as a minimization of the total squared distance from all data vectors to the corresponding

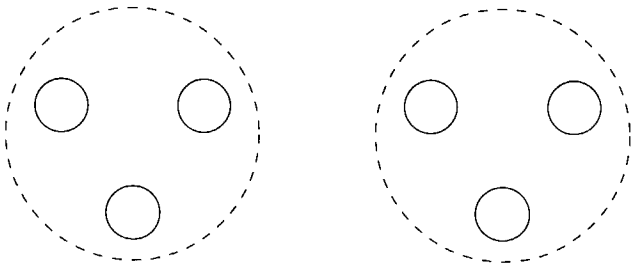


Figure 1.5: A set of two or six clusters showing the conflict between cluster generalization and cluster specialization. The solid circles are more specialized clusters, while the dashed circles represent more general clusters.

cluster mean F_1 (sum-of-squared-error criterion), see also Eq. 6.1. The optimization of the heterogeneity of the union of two clusters is expressed as a maximization the minimal joint variance of any two clusters F_2, see of Eq. 6.4.

In order to solve this bicriterion optimization problem, we actually used the ε-constraint method. We minimized the sum-of-squared-error criterion F_1, while we constrain the minimal joint variance criterion, that is, $F_2 \geq \sigma^2_{max}$. For the image segmentation problem in Chapter 5, we assume that we can estimate a proper value of the joint variance constraint. This assumption is based on the fact that the minimization of the sum-of-squared-error criterion subject to the minimal joint variance constraint generally results in clusters with a variance below the established constraint value σ^2_{max}, see proof in Section 5.2. Accordingly, the minimal joint variance constraint value can be estimated from the image, based among others on the image noise.

In Chapter 6, we deal with the cluster problem. Since the number of data vectors is small compared to those in the image segmentation problem (10^4 data vectors versus 10^6 pixels), estimating the Pareto set becomes feasible in the clustering problem. Implicitly, the ε-constraint method has been applied to explore the model parameter space, resulting in an estimation of the Pareto front. It appears that from the shape of the Pareto front suitable solutions can indeed be recognized. Accordingly, studying the Pareto front can be used as a cluster tendency or validation method. See for instance the computed feasible region and corresponding Pareto front for the *Iris* data set in Fig 1.6. In the figure, the sum-of-squared-error criterion F_1 is displayed as a function of the minimum joint variance F_2. The Pareto front consists of those solutions from the feasible region that have a minimal squared error while their joint variance is maximal; the black dots in the figure represent the, in this case, discrete Pareto front. Since the Pareto front is non-convex, in retrospect it can be concluded that the popular weighted sum method would not have been suitable to find the Pareto front for the optimization of this multi-criterion model. Additionally, in Chapter 6, we show that heuristics can be defined that automatically find a suitable trade-off for the cluster problem from an analysis of the Pareto front. In effect, the use of these heuristics result in a value function method for this bicriterion problem.

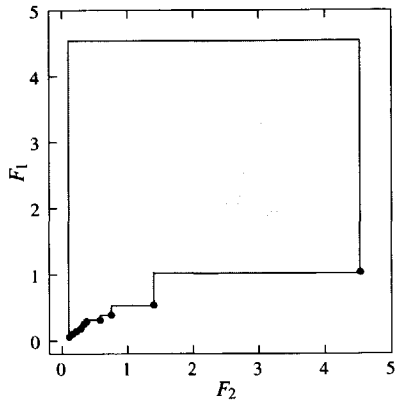


Figure 1.6: Illustration of the feasible region of the *Iris* data set estimated with the ε-constraint method. The conflicting criteria are the sum-of-squared-error criterion F_1 and the minimum joint variance criterion F_2. In this case the Pareto front (solid dots) is discrete, because it concerns a discrete optimization problem. This figure is an adapted version of Fig. 6.11 in Chapter 6.

Both for the image segmentation problem and the clustering problem the corresponding single-criterion optimization algorithm has been specifically designed. The algorithm contains non-deterministic elements in order to escape from local optima, since this optimization problem is also known to be intractable.

1.6 Discussion

This study of modeling and optimization aspects of a number of unsupervised pattern recognition problems makes clear that by nature these problems give rise to multiple conflicting criteria.

First, we want to stress that the popular, almost automatic transformation of a multi-criterion optimization problem into a single criterion optimization problem using weighting coefficients has a number of drawbacks. The most severe drawback is that the Pareto front, the set of mathematical equivalent optimal solutions, cannot be determined in case the front is non-convex. Consequently, with the weighted coefficients method some optimal solutions are ruled out a priori.

Second, as we showed with the clustering model proposed in this thesis, the shape of the Pareto front can give valuable information about a proper trade-off between the conflicting optimization criteria. Moreover, it appears that the Pareto front of this model is non-convex, which we showed by estimating it with the less conventional ε-constraint method. Accordingly, it could indeed not be found with the weighted sum method.

Third, any available information should be exploited in order to articulate the analyst's

preference for certain solutions in a multi-criterion model. These preferences can for instance be expressed in the form of additional hard constraints, as those we implemented for the motion correspondence problem. In that way, we were able to solve the corresponding complex combinatorial multi-criterion optimization problem with an a priori method.

In view of the optimization issues raised in this chapter, we foresee a number of concrete improvements that can be made to solve the problems that we dealt with in this thesis. First, the ε-constraint method that is implicitly used for the motion correspondence problem could be extended so the Pareto front of this problem can be inspected. Accordingly, it would be possible to find a proper value of the corresponding constraint value (ϕ_{max}). This would be very interesting, since the determination of a suitable ϕ_{max} value is far from trivial.

Second, for the clustering problem the shape of the Pareto front indeed reveals a suitable trade-off between generalization and specialization criteria. The clustering problem is, however, certainly not yet solved. For instance, in case the variance of the clusters differs the current model has shortcomings. Other generalization criteria may be considered or added to improve the model.

Finally, we want to emphasize that especially in case of incommensurable criteria it is important to make the criteria explicit. Instead of turning the problem into a single criterion problem, one should to be aware of the conflicting criteria that are inherent to the problem.

Bibliography

[1] J.C. Bezdek and N.R. Pal. Some new indexes of cluster validity. *IEEE Transactions on Systems, Man, and Cybernetics — Part B*, 28(3):301–315, 1998.

[2] D. Corne, M. Dorigo, and F. Glover. *New Ideas in Optimization*. McGraw-Hill, Cambridge, 1999.

[3] I.J. Cox. A review of statistical data association techniques for motion correspondence. *International Journal of Computer Vision*, 10(1):53–66, 1993.

[4] D.L. Davies and D.W. Bouldin. A cluster separation measure. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 1(2):224–227, April 1979.

[5] R.O. Duda, P.E. Hart, and D.G. Stork. *Pattern Classification*. John Wiley and Sons, Inc., New York, 2001.

[6] J. C. Dunn. Well separated clusters and optimal fuzzy partitions. *Journal of Cybernetics*, 4:95–104, 1974.

[7] F. Glover and M. Laguna. *Tabu Search*. Kluwer Academic Pusblishers, Boston, 1997.

[8] D.E. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley, 1989.

[9] Y.Y. Haims, D.A. Wismer, and L.S. Lasdon. On a bicriterion formulation of the problems of integrated system identification and system optimization. *IEEE Transactions on Systems, Man, and Cybernetics*, 1(3):296–297, July 1971.

[10] R.M. Haralick and L.G. Shapiro. Survey: Image segmentation techniques. *Computer Vision, Graphics, and Image Processing*, 29:100–132, 1985.

[11] D.M. Himmelblau. *Applied Nonlinear Programming*. McGraw-Hill, New York, 1972.

[12] L.J. Hubert and P. Arabie. Comparing partitions. *Journal of Classification*, 2:193–218, 1985.

[13] C.L. Hwang and A.S.M. Masud. *Multiple Objective Decision Making – Methods and Applications: A State-of-the-Art Survey*. Springer-Verlag, Berlin, Heidelberg, 1982.

[14] A.K. Jain. *Fundamentals of Digital Image Processing*. Prentice-Hall Inc., New Jersey, 1989.

[15] A.K. Jain and R.C. Dubes. *Algorithms for Clustering Data*. Prentice-Hall Inc., New Jersey, 1988.

[16] A.K. Jain, R.P.W. Duin, and J. Mao. Statistical pattern recogntion: A review. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(1):4–34, January 2000.

[17] J. Jensen, R. Heusdens, and C.J. Veenman. Optimal time-differential encoding of sinusoidal model parameters. In *Proceedings of the 22nd Symposium on Information Theory in the BENELUX*, pages 165–172, The Netherlands, May 2000.

[18] S. Kirkpatrick, C. Gelatt, and P. Vecchi. Optimization by simulated annealing. *Science*, 220:671–679, 1983.

[19] J. MacQueen. Some methods for classification and analysis of multivariate observations. In L.M. Le Cam and J. Neyman, editors, *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability*, volume 1, pages 281–297, 1967.

[20] K.M. Miettinen. *Nonlinear Multiobjective Optimization*. Kluwer Academic Publishers, Dordrecht, 1999.

[21] A. Osyczka. *Multicriterion Optimization in Engineering*. Ellis Horwood Ltd., West Sussex, 1984.

[22] N.R. Pal and S.K. Pal. A review on image segmentation techniques. *Pattern Recogntion*, 26(9):1277–1294, 1993.

[23] A.B. Poore. Multidimensional assignments and multitarget tracking. In *Partitioning Data Sets; DIMACS Workshop*, pages 169–196, New Brunswick, USA, 1995. American Mathematical Society.

[24] W.H. Press, S.A. Teukolsky, W.T. Vetterling, and B.P. Flannery. *Numerical Recipes in C; The Art of Scientific Computing*. Cambridge University Press, second edition edition, 1988.

[25] K. Rangarajan and M. Sha. Establishing motion correspondence. *CVGIP: Image Understanding*, 24(6):56–73, July 1991.

[26] I.K. Sethi and R. Jain. Finding trajectories of feature points in a monocular image sequence. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 9(1):56–73, January 1987.

[27] M. Shah and R. Jain. *Motion-Based Recognition*. Kluwer Academic Publishers, Norwell, Massachusetss, USA, 1997.

[28] R.E. Steuer. *Multiple Criteria Optimization: Theory, Computation, and Application.* John Wiley and Sons, Inc., 1986.

[29] S. Theodoridis and K. Koutroumbas. *Pattern Recognition.* Academic Press, London, 1999.

[30] C.J. Veenman, M.J.T. Reinders, and E. Backer. Competitive segmentation: A struggle for image space. In *Proceedings of the Sixth International Conference on Parallel Problem Solving from Nature*, pages 517–526, Paris, France, September 2000.

[31] D.A. Wismer and R. Chattergy. *Introduction to nonlinear optimization: a problem solving approach.* Elsevier, North-Holland, 1978.

Chapter 2

Resolving Motion Correspondence for Densely Moving Points

This chapter has been published as 'Resolving Motion Correspondence for Densely Moving Points', by C.J. Veenman, M.J.T. Reinders, and E. Backer, in *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 23, no. 1, pp. 54-72, Januari 2001.

Abstract This paper studies the motion correspondence problem for which a diversity of qualitative and statistical solutions exist. We concentrate on qualitative modeling, especially for situations where assignment conflicts arise, either because multiple features compete for one detected point or because multiple detected points fit a single feature point. We leave out the possibility of point track initiation and termination, because that principally conflicts with allowing for temporary point occlusion. We introduce individual, combined, and global motion models and fit existing qualitative solutions in this framework. Additionally, we present a new efficient tracking algorithm that satisfies these — possibly constrained — models in a greedy matching sense, including an effective way to handle detection errors and occlusion. The performance evaluation shows that the proposed algorithm outperforms existing greedy matching algorithms. Finally, we describe an extension to the tracker that enables automatic initialization of the point tracks. Several experiments show that the extended algorithm is efficient, hardly sensitive to its few parameters, and qualitatively better than other algorithms, including the presumed optimal statistical multiple hypothesis tracker.

Keywords: Motion correspondence, feature point tracking, target tracking, algorithms.

2.1 Introduction

Motion correspondence has a number of applications in computer vision, ranging from motion analysis, object tracking and surveillance to optical flow and structure from motion [11], [24], [25], [26]. Motion correspondence must be solved when features are to be tracked that appear identical or that are retrieved with a simple feature detection scheme which loses essential information about their appearance. Hence, the motion correspondence problem deals with finding corresponding points from one frame to the next in the absence of significant appearance identification (see Fig. 2.1(a)). The goal is to determine a path or track of the moving feature points from entry to exit from the scene, or from the start to the end of the sequence. During presence in the scene, a point may be temporarily occluded by some object. Additionally, a point may be missed and other points may be falsely detected because of a failing detection scheme, as in Fig. 2.1(b) and Fig. 2.1(c)[1].

A candidate solution to the correspondence problem is a set of tracks that describes the motion of each point from scene entry to exit. We adopt a uniqueness constraint, stating that one detected point uniquely matches one feature point. When 2-D projections from a 3-D scene are analyzed, this is not trivial, because one feature point may obscure another. If we further assume that all M points are detected in all n frames, the number of possible track sets is $(M!)^{n-1}$. Among these solutions, there is a unique track set that describes the true motion of the M points. In order to identify the true motion track set, we need prior knowledge about the point motion, because otherwise all track sets are equally plausible. This knowledge can range from general physical properties like inertia and rigidity to explicit knowledge about the observed objects, like for instance the possible movements of a robot arm in the case that

[1] In the remainder of this paper we display the measurements from different time instances in one box and use # labels to indicate the time the point was detected.

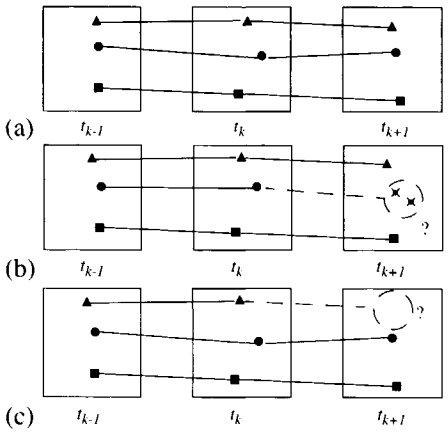


Figure 2.1: Three moving points are measured at three time instances. The lines represent the point correspondences in time. In (a) all points are measured at every time instance. In (b) there is an extra or false measurement at t_{k+1}, and in (c) there is a missing measurement at t_{k+1}

points on a robot arm are to be tracked. Clearly, generic motion correspondence algorithms cannot incorporate scene information. Moreover, they do not differentiate between the points in the scene, i.e. all points are considered to have similar motion characteristics.

When many similar points are moving through a scene, ambiguities may arise, because a detected point may well fit correctly to the motion model of multiple features points. Additional ambiguities are caused by multiple detected points that fit correctly to the model of a single feature. These correspondence ambiguities can be resolved if combined motion characteristics are modeled, like for instance least average deviation from all individual motion models. Besides resolving these ambiguities we also have to incorporate *track continuation* in order to cope with point occlusion and missing detections. Other events that we may need to model are *track initiation* and *track termination*, so that features can enter and leave the scene, respectively.

The available motion knowledge is usually accumulated in an appropriate model. Then, a specific strategy is needed to find the optimal solution among the huge amount of candidate solutions defined by the model. When the nearest-neighbor motion criterion is used, (see also Section 2.3) and there are neither point occlusions nor detection errors, track set optimality only depends on the point distances between any two consecutive frames. It is then legitimate to restrict the scope of the correspondence decision to one frame ahead, which we call a *greedy matching* solution to the correspondence problem. In other cases in which velocity state information is involved, correspondence decisions for one frame influence the optimal correspondence for the next frames and the problem becomes increasingly more complex. In such cases only a *global matching* over all frames can give the optimal result. In this paper, we consider the more difficult cases, i.e. dense and fast moving points, which makes the use

of velocity state information essential. Because there are no efficient algorithms to find the optimal track set by global matching, only approximation techniques apply. Several statistical [2] and qualitative approximation techniques have been developed both in the field of target tracking and computer vision.

Statistical Methods

The two best known statistical approaches are the Joint Probabilistic Data-Association Filter (JPDAF) [9] and the Multiple Hypothesis Tracker (MHT) [20]. The JPDAF matches a fixed number of features in a greedy way and is especially suitable for situations with clutter. It does not necessarily select point measurements as exact feature point locations, but, given the measurements and a number of corresponding probability density functions, it estimates these positions. The MHT attempts to match a variable number of feature points globally, while allowing for missing and false detections. Quite a few attempts have been made to restrain the consequent combinatorial explosion, such as [3], [4], [5], [6], [15], [16]. More recently, the equivalent sliding window algorithms have been developed, which match points using a limited temporal scope. Then, these solve a multidimensional assignment problem, which is again NP-hard, but real-time approximations using Lagrangian relaxation techniques are available [7], [8], [17], [18], [23].

A number of reasons make the statistical approaches less suitable as a solution to the motion correspondence problem. First, the assumptions that the points move independently and, more strongly, that the measurements are distributed normally around their predicted position may not hold. Second, since statistical techniques model all events as probabilities, these techniques typically have quite a number of parameters, such as the Kalman filter parameters, and a priori probabilities for false measurements, and missed detections. In general, it is certainly not trivial to determine optimal settings for these parameters. In the experiments section we show that the best known statistical method (MHT) is indeed quite sensitive to its parameter setting. Moreover, the a priori knowledge used in the statistical models is not differentiated between the different points. As a consequence, the initialization may be severely hampered if the initial point speeds are widely divergent, because the state of the motion models only gradually adapts to the measurements. Finally, the statistical methods that optimize over several frames are despite their approximations computationally demanding, since the complexity grows exponentially with the number of points.

Heuristic Methods

Alternatively, a number of attempts has been made to solve the motion correspondence problem with deterministic algorithms [1], [12], [14], [19], [22]. These algorithms are usually conceptually simpler and have less parameters. Instead of probability density functions, *qualitative motion heuristics* are used to constrain possible tracks and to identify the optimal track set. By converting qualitative descriptions like smoothness of motion and rigidity into quantitative measures, a distance from the optimal motion can be expressed (where a zero distance makes a correspondence optimal). The most commonly known algorithm is

the conceptually simple greedy exchange algorithm [22], which iteratively optimizes a local smoothness of motion criterion averaged over all points in a sequence of frames. The advantage of such deterministic algorithms is that it is quite easy to incorporate additional constraints, like (adaptive) maximum speed, and a maximum deviation from smooth motion, while this a priori knowledge can restrain the computational cost and improve the qualitative performance, e.g. [1], [10].

The main contributions of this paper are the presentation of a *1) qualitative motion modeling framework* for the motion correspondence problem. We introduce the notion of individual motion models, combined motion models, and a global motion model, and we differentiate between strategies to satisfy these models. Further, we propose a *2) new efficient algorithm* that brings together the motion models, an optimal strategy, and an effective way to handle detection errors and occlusion. Finally, we present an extensive *3) comparative performance evaluation* of a number of different qualitative methods.

The outline of the paper is as follows. We start by giving a formulation of the motion correspondence problem in the next section. Then in Section 2.4, we present our qualitative motion model and show how the existing deterministic motion correspondence algorithms can be fit into it. Additionally, we present a new algorithm that effectively resolves motion correspondence using the presented model in Section 2.5. In Section 2.6, we compare the qualitative performance, the efficiency, and the parameter sensitivity of the described algorithms. Further, we show how the proposed algorithm can be extended with self-initialization and evaluate it with synthetic data experiments in Section 2.7. We broaden this evaluation in Section 2.8, with real-data experiments. We finish the paper with a discussion on possible extensions and some conclusions.

2.2 Problem Statement

In this section we describe the motion correspondence problem as treated in this paper. In motion correspondence, the goal is tracking points that are moving in a 2-D space that is essentially a projection of a 3-D world. The positions of the points are measured at regular times, resulting in a number of point locations for a sequence of frames. For the moment, we assume that we have initial motion information of all points, which is given by point correspondences between the first two frames. From Section 2.7 onwards this restriction is lifted. Since the measured points are projections, points may become occluded and thus missing. Moreover, the point detection may be imperfect, resulting in missing and false point measurements. Because long occlusion on the one hand and scene entrance and exit on the other hand are conflicting requirements, we leave out the possibility of track initiation and track termination, so the number of features to be tracked is constant. Applications using this problem definition range from object tracking in general, like animal tracking to perform behavior analysis, particle tracking, and cloud system tracking, to feature tracking for motion analysis. In the remainder of this paper, we abbreviate the moving points to 'points' and their measured 2-D projections to 'measurements'.

More formally: There are M points, p_i, moving around in a 3-D world. Given is a

sequence of n time instances for which at each time instance t_k there is a set X^k of m_k measurements $\mathbf{x}_j^k$, with $1 \leq j \leq m_k$ and $1 \leq k \leq n$, of points p_i. The measurements $\mathbf{x}_j^k$ are vectors representing 2-D coordinates in a 2-D space, with dimensions S_w (width) and S_h (height). The number of measurements, m_k, at t_k, can be either smaller (occlusion) or larger (false measurements) than M. At t_1, the M points ($M \leq m_1$) are identified among the m_1 measurements. Moreover, the corresponding M measurements at t_2 are given. The task is to return a set of M tracks that represent the (projected) motion of the M points through the 2-D space from t_1 to t_n using the movements between t_1 and t_2 as initial motion characteristics. A track T_i, with $1 \leq i \leq M$, is an ordered n-tuple of corresponding measurements: $\langle \mathbf{x}_{j_1}^1, \mathbf{x}_{j_2}^2, ..., \mathbf{x}_{j_n}^n \rangle$, with $1 \leq j_k \leq m_k$. It is assumed that points do not enter or leave the scene and that the movement can be modeled independently. A track that has been formed up to t_k is called a track head and is denoted as T_i^k, where $1 \leq i \leq M$.

2.3 Qualitative Motion Modeling

The assumption underlying the qualitative model that we advocate is that points move smoothly from time instance to time instance. That is, not only the individual points move smoothly, but the total set of points moves smoothly as well, both between time instances and over the whole sequence. Hereto, we define a qualitative model in which these qualitative statements are explicitly represented by a composition of motion models that we have called the *global motion model*, the *combined motion model* and the *individual motion model*. The individual motion model represents the motion of individual points. To embed the motion smoothness constraints, we can make use of well-known general physical properties like rigidity and inertia. Without loss of generality, we only consider first-order motions, and thus leave out acceleration-state information. Consequently, the motion vector of a feature point can be estimated from only two consecutive measurements. On the basis of the motion vector and the adopted individual motion model, the position of the point at the next time instance can be predicted. The measurement that is closest to this prediction can then be selected as corresponding measurement. In reality, however, the points do not move exactly according to their predictions, because of shortcomings of the adopted individual motion model. These are among others caused by the limited order of the motion model, the fact that measurements are 2-D projections of 3-D movements, and by noise in the system.

To express the misfit between a measurement and the predicted position, the candidate motion vector between the candidate measurement and the last measurement in the track is calculated. Using the inertia argument the cost representing the misfit is expressed in terms of the candidate motion and the previous true motion vector. These costs can be used to select the appropriate candidate measurement to make a correspondence. When points are moving far apart from each other or when they move reasonably according to their models, their measurements can easily be assigned to the corresponding feature point. With densely moving point sets, however, assignment conflicts can easily occur. That is, one measurement fits correctly multiple individual motion models or multiple measurements correctly fit one motion model. To resolve these ambiguities, the motion smoothness constraint is also im-

posed on the complete set of points. To this end, we introduce the *combined motion model*, which expresses the deviation from this motion constraint. As an example, we could enforce that the average deviation from the individual motion models is minimal.

Even with the use of the combined motion model it is not always possible to decide on point correspondences. For that reason the motion smoothness constraint is additionally extended over the whole sequence in the *global motion model*.

In the remainder of this section, we present some individual motion models, combined motion models and a global motion model and we give quantitative expressions for each of them. To simplify the notation of the criteria that lead to the point tracks T_i, we introduce the assignment matrix $A_k = \left[a_{ij}^k\right]$, where the entries a_{ij}^k have the following meaning: $a_{ij}^k = 1$ if and only if measurement $\mathbf{x}_j^{k+1}$ is assigned to track head T_i^k and otherwise zero. Because some measurements are false and others are missing, there can be some measurements that are not assigned to a track head (all zeros in a column in A^k), and some track heads that have no measurement assigned to them (all zeros in a row in A^k). Or, more formally:

$$\sum_{i=1}^{M} a_{ij}^k \leq 1, \quad 1 \leq j \leq m_{k+1}; \quad \sum_{j=1}^{m_{k+1}} a_{ij}^k \leq 1, \quad 1 \leq i \leq M; \quad a_{ij}^k \in \{0, 1\} \tag{2.1}$$

We use two alternative notations for a correspondence between a measurement and a track head. First, we define α_j^k as:

$$\alpha_j^k = i \Leftrightarrow a_{ij}^k = 1 \tag{2.2}$$

Second, we use ordered pairs (i, j) to indicate that measurement $\mathbf{x}_j^{k+1}$ has been assigned to track head T_i^k. Z^k then contains all assignment pairs from t_k to t_{k+1} according to:

$$Z^k = \{(i, j) \mid a_{ij}^k = 1\} \tag{2.3}$$

Tracks T_i can now be derived from A, which is the concatenation of the assignment matrices A^k. We introduce a deviation matrix $D^k = \left[c_{ij}^k\right]$ to denote all individual assignment costs c_{ij}^k between track heads T_i^k and measurements $\mathbf{x}_j^{k+1}$.

The assignment matrix identifies all correspondences from frame to frame, while the deviation matrix quantifies the deviation from the individual motion track per correspondence. The matrices A^k and D^k both have M rows and m_{k+1} columns. The rows represent the M track heads, T_i^k, and the columns represent the m_{k+1} measurements, $\mathbf{x}_j^{k+1}$ that have been detected at t_{k+1}.

Individual Motion Models

We now formulate three individual motion models, together with an expression to compute a *deviation* from the optimal track. The first model uses only one previous measurement to

predict the new position. We have indicated the dependence of only one previous measurement by the order of the individual model: $O_{im} = 1$. The other two individual models depend on two measurements and consequently have order $O_{im} = 2$. The following motion criteria coefficients c_{ij}^k are all defined from track head T_i^k to a measurement $\mathbf{x}_j^{k+1}$.

im1 The *nearest-neighbor* model does not incorporate velocity information. It only states that a point moves as little as possible from t_k to t_{k+1}.

$$c_{ij}^k = \| \mathbf{x}_j^{k+1} - \mathbf{x}_i^k \|, \qquad \text{where } 0 \le c_{ij}^k \le \sqrt{S_w^2 + S_h^2}. \tag{2.4}$$

im2 The *smooth motion* model as introduced by Sethi and Jain [22] assumes that the velocity magnitude and direction change gradually. The smooth motion is formulated quantitatively in the following criterion:

$$\begin{aligned} c_{ij}^k \;=\; & 0.1 \left[1 - \frac{\left(\mathbf{x}_i^k - \mathbf{x}_{\alpha_i^k}^{k-1}\right) \cdot \left(\mathbf{x}_j^{k+1} - \mathbf{x}_i^k\right)}{\left\|\mathbf{x}_i^k - \mathbf{x}_{\alpha_i^k}^{k-1}\right\| \left\|\mathbf{x}_j^{k+1} - \mathbf{x}_i^k\right\|} \right] + \\ & 0.9 \left[1 - 2 \frac{\sqrt{\left\|\mathbf{x}_i^k - \mathbf{x}_{\alpha_i^k}^{k-1}\right\| \left\|\mathbf{x}_j^{k+1} - \mathbf{x}_i^k\right\|}}{\left\|\mathbf{x}_i^k - \mathbf{x}_{\alpha_i^k}^{k-1}\right\| + \left\|\mathbf{x}_j^{k+1} - \mathbf{x}_i^k\right\|} \right], \end{aligned} \tag{2.5}$$

where $0 \le c_{ij}^k \le 1$.

im3 The *proximal uniformity* model by Rangarajan and Shah [19] assumes little motion in addition to constant speed. The deviation is quantified in the following criterion:

$$\begin{aligned} c_{ij}^k \;=\; & \frac{\left\| \left(\mathbf{x}_i^k - \mathbf{x}_{\alpha_i^k}^{k-1}\right) - \left(\mathbf{x}_j^{k+1} - \mathbf{x}_i^k\right) \right\|}{\sum_{p=1}^{M} \sum_{q=1}^{m_{k+1}} \left\| \left(\mathbf{x}_p^k - \mathbf{x}_{\alpha_p^k}^{k-1}\right) - \left(\mathbf{x}_q^{k+1} - \mathbf{x}_p^k\right) \right\|} + \\ & \frac{\left\| \mathbf{x}_j^{k+1} - \mathbf{x}_i^k \right\|}{\sum_{p=1}^{M} \sum_{q=1}^{m_{k+1}} \left\| \mathbf{x}_q^{k+1} - \mathbf{x}_p^k \right\|}, \end{aligned} \tag{2.6}$$

where $0 \le c_{ij}^k \le 1$.

Combined Motion Models

Combined motion models serve to resolve correspondence conflicts between two successive frames in case of dense moving point sets, making the individual model errors dependent on each other. Next, we give two combined model criterions C^k as a function of A^k and D^k, that are defined at t_k over all established track heads.

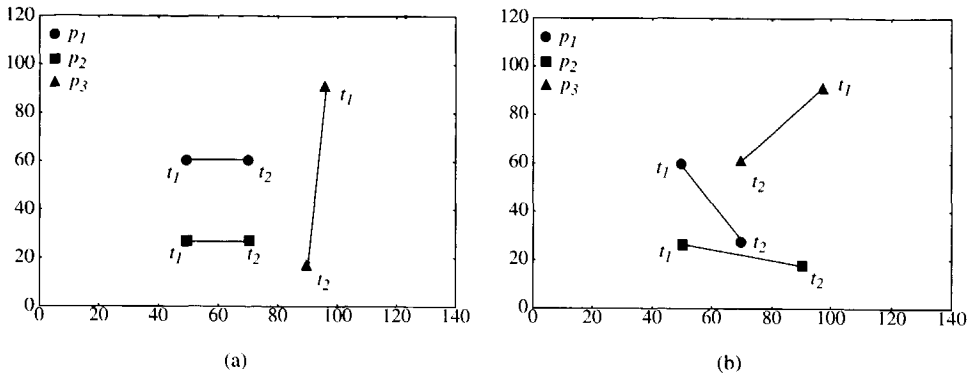


Figure 2.2: Three moving points that are matched with **im1** and **cm1** using either $z = 1$ (a) or $z = 2$ (b). As a consequence larger deviations are penalized more in (b).

cm1 The *average deviation* model. This is a typical combined model which is usually realistic. It accounts for the average deviation from the optimal track according to the individual model [21], [22], [26]. Quantitatively, we use the generalized mean, which has a z parameter to differentiate between emphasis on large and small deviations from the optimal individual track (see Fig. 2.2).

$$C^k(A^k, D^k) = \left[\frac{1}{M} \sum_{i=1}^{M} \sum_{j=1}^{m_{k+1}} a_{ij}^k \left(c_{ij}^k\right)^z \right]^{\frac{1}{z}} \tag{2.7}$$

cm2 The *average deviation conditioned by competition and alternatives* model is derived from [1], [19]. In this combined model measurements are assigned to that track head that gives low deviation from the optimal track, while both the other tracks are less attractive for this measurement and the other measurements are less attractive for this track.

$$C^k(A^k, D^k) = \frac{1}{M} \sum_{i=1}^{M} \sum_{j=1}^{m_{k+1}} \left(a_{ij}^k c_{ij}^k - w_1 R_a(i) - w_2 R_c(j)\right), \tag{2.8}$$

where:

$$R_a(i) = \frac{1}{m_{k+1} - 1} \sum_{q=1}^{m_{k+1}} \left(1 - a_{iq}^k\right) c_{iq}^k; \qquad R_c(i) = \frac{1}{M - 1} \sum_{p=1}^{M} \left(1 - a_{pj}^k\right) c_{pj}^k \tag{2.9}$$

$R_a(i)$ represents the average cost of alternatives for T_i^k and $R_c(j)$ the average cost for competitors of $\mathbf{x}_j^{k+1}$.

Global Motion Model

To find the optimal track set (over all frames) according to a certain combined model, we need to compute the accumulated global motion deviation $S(D)$ as in the following expression:

$$S(D) = \min_{A \in U} \sum_{k=O_{im}}^{n-1} C^k(A^k, D^k), \tag{2.10}$$

where U is the set of matrices A that satisfy Eq.2.1.

That is, the overall minimum of averaged combined criterions defines the optimal track set. Because finding this minimum is computationally expensive, a *greedy matching* is considered in this paper. This means that instead of finding correspondences over all frames, we establish optimal correspondences between two successive frames, given the state of the individual motion models and the combined model up to that moment. After these sub-optimal correspondences have been established the states of the individual models are adjusted and the next frame is considered. In other words, Eq.2.10 is approximated by minimizing $C^k(A^k, D^k)$ separately, i.e.:

$$\hat{S}(D) = \sum_{k=O_{im}}^{n-1} C_{min}^k(D^k), \qquad \text{where } C_{min}^k(D^k) = \min_{A^k \in U^k} C^k(A^k, D^k) \tag{2.11}$$

This approximation approach reduces the complexity of the problem considerably, although at the cost of greedy, possibly less plausible, correspondence decisions (see for example Fig. 2.3). In the remainder, we leave out the D and D^k parameter for S, $\hat{S}$, and C_{min}^k respectively.

Model Constraints

The motion models we have described so far allow for any point speed and for any deviation from smoothness. The models only state that those assignments are preferred that have little deviation from the individual model. There are, however, situations in which there is more knowledge available about the point motions, like the minimum speed (d_{min}) and the maximum speed (d_{max}) [1], [12], [14], [21], maximum violation of smoothness (ϕ_{max}) [1], [14], [21], and spatial or temporal adaptive speed and smoothness violation constraints [10]. When imposed on the individual motion models, these constraints enable the recognition of impossible assignments, which can be both qualitatively and computationally beneficial. These constraints can for instance be implemented by setting the individual criterion to a very high value when some constraint is violated. The strategy that satisfies the models (see next subsection) can exploit these constraints more adequately by leaving out of consideration those correspondences that violate the motion constraints.

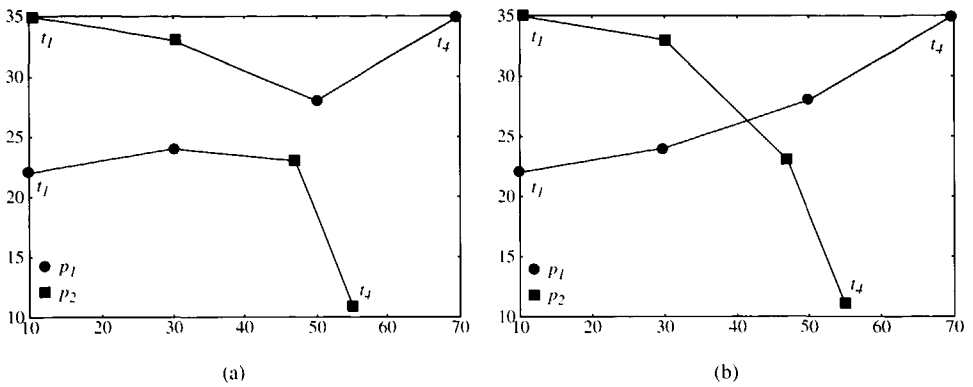


Figure 2.3: Two moving points at four time instances. When the smooth motion model (**im2**) with the average deviation model (**cm1**) are assumed, (b) gives a two times lower deviation from the optimal path than (a). However, (a) is decided for when greedy matching is used.

Strategy

To find the optimal track set, we compute the global motion deviation. However, we are not interested in the actual value of S, but in the assignment matrix A that results in the minimal global motion deviation. In the next section, we first show how existing algorithms approximate the minimization of C^k and consequently deliver a sub-optimal solution A^k. In Section 2.5, we present an optimal as well as efficient algorithm to find that A^k that minimizes C^k.

2.4 Algorithms

Having modeled the feature point motion and having described quantitative expressions that can be used to identify the optimal track set, we now review a number of existing algorithms and fit them in our motion framework using our concept of individual and combined motion models. Further, we describe the strategy they use to find the optimal correspondences. Because all algorithms perform greedy matching, their task is to find C^k_{min}.

S&S Algorithm (im2/cm1/z=1)

The first algorithm we looked at was originally developed by Sethi and Jain [22]. The original algorithm assumes a fixed number of feature points to be tracked and does not allow for occlusion and detection errors. Here, we describe the adjusted algorithm by Salari and Sethi [21] that partially fixes these shortcomings. The algorithm adopts a smooth motion model for individual motion (**im2**). The combined motion model is an average deviation model

(**cm1/z=1**). To find an optimum of the global motion, the algorithm iteratively exchanges measurements between tracks to minimize the criterion on average.

Initially, the tracks are led through the nearest neighboring measurements in the sequence. In this stage conflicts are 'resolved' on a first-come, first-served basis. That is, at t_{k+1} measurements are assigned to the closest track parts T_i^k that have been formed up to t_k to which no point was assigned yet. Consequently, the initialization procedure is a greedy **im1/cm1** approximation.

Then, each iteration step modifies at most two assignment pairs somewhere in the sequence, by exchanging the second entry of the pair. The algorithm considers all possible exchanges within the d_{max} range of two track heads in the whole sequence and the exchange that gives the highest gain by decreasing the average criterion deviation is executed. The iteration phase stops when gain can no longer be obtained. The exchange gain between assignment pairs (i, p) and (j, q) (see Eq.2.3) is defined in the following way:

$$g_{ij}^k = c_{ip}^k + c_{jq}^k - (c_{iq}^k + c_{jp}^k) \tag{2.12}$$

To achieve even better tracking results, the algorithm first optimizes correspondences over all frames in the forward direction and then (after this iteration phase stops) it optimizes correspondences in the backward direction. Only when the optimization process has not changed anything in either direction, the algorithm stops. This bi-directional optimization process can indeed increase the tracking quality, but, unfortunately, this process is not guaranteed to converge, especially with densely moving points [19].

In contrast with what we said before, this algorithm seems to optimize over the whole sequence. However, when we look carefully at the optimization process within one iteration phase, we see that this is only partially true. As long as the tracks are wrong at the start, exchanges in the remainder of the track will mostly be useless. This is due to the fact that the tracks were initialized using another criterion than the one that is considered in the iteration phase. Consequently, the optimization is only effective at the initial measurements of the tracks. This problem is most severe when the sequence is long and when the difference between the initialization criterion (nearest neighbor) and the optimization criterion (smooth motion) is large, i.e. with high speeds and high densities. We tested this statement by feeding to the S&S algorithm the example shown in Fig.2.3. If we do not optimize in both directions, the S&S algorithm indeed makes greedy correspondences as in Fig.2.3(a), which supports the statement that S&S is a greedy matching algorithm.

The Salari and Sethi version of this so-called greedy exchange algorithm additionally proposes a way to resolve track continuation, initiation and termination. They introduce a number of *phantom points* to the set of measurements in each frame. These phantom points serve as replacements of missing measurements, while satisfying local constraints. By imposing the maximum allowed local smoothness criterion and a maximum speed, missed measurements are recognized and filled in with phantom points. Moreover, the constraints also allow the detection of false measurements. Effectively, false measurements are replaced by phantom points if the introduction of a phantom point results in a lower criterion value.

This approach generally works fine except that missing measurements (represented by a phantom point) always have the maximum criterion and displacement. For instance, if point p_i has not been measured at t_k, the algorithm can easily associate a measurement of p_i at t_{k+1} to another point which is within the criterion range ϕ_{max}. It is important to remark that the phantom points only enforce that the local movement constraints are satisfied, but when a phantom point is put in a track, the track is in fact divided into two tracks. In other words, this maximum criterion approach solves the correspondence problem *up to* the maximum criterion. Choosing a low maximum criterion leads to many undecided track parts and a higher maximum criterion leads to possibly wrong correspondences. This is where the track initiation/termination and occlusion events become conflicting requirements, as already mentioned in Section 2.2.

R&S Algorithm (im3/cm2)

A different approach to the correspondence problem is chosen by Rangarajan and Shah [19]. They have a different combined motion model and do not use an iterative optimization procedure. The R&S algorithm assumes a fixed number of feature points and it allows for temporary occlusion or missing point detections, but not for false detections. It uses the proximal uniformity model (**im3**) as individual motion model and **cm2** as the combined motion model. This algorithm does not constrain the individual point motion, i.e. it does not have a d_{max} or ϕ_{max} parameter.

To find the minimum of the combined model (Eq.2.8), the authors use a greedy non-iterative algorithm. In each step of the algorithm, that particular point $\mathbf{x}_j^{k+1}$ is assigned to track head T_i^k that has a low deviation from the optimal motion (low individual deviation) while on average all alternative track heads have a larger deviation with respect to $\mathbf{x}_j^{k+1}$ *and* on average all other measurements have a worse criterion with respect to T_i^k.

We continue the description of the algorithm in terms that fit the proposed motion framework as established in Section 2.3. The algorithm selects that assignment pair (i, j) that maximizes $R'_a(i) + R'_c(j)$ among all minimal track head extensions, where $R'_a(i)$ and $R'_c(j)$ are derived from Eq.2.9 according to:

$$R'_a(i) = \frac{1}{m_{k+1} - 1} \sum_{q=1, q \neq j}^{m_{k+1}} c_{iq}^k; \qquad R'_c(j) = \frac{1}{M - 1} \sum_{p=1, p \neq i}^{M} c_{pj}^k \tag{2.13}$$

Then, an optimal assignment pair $g(X_t, X_m)$ is repeatedly selected in the following way:

$$g(X_t, X_m) = \big((i, j) \mid i = \arg\max_{p \in X_t} \big(R'_a(p) + R'_c(j)\big), j = \arg\min_{q \in X_m} c_{pq}^k\big), \tag{2.14}$$

where X_t is the set of track head indices that have not yet been assigned a measurement, and X_m is the set of measurement indices that have not yet been assigned to a track head. After an assignment has been found, the track head and measurement are removed from the respective

index sets X_t and X_m. The algorithm accumulates the assignment costs, and eventually stops when X_t is empty. The criterion computation can be summarized in the recurrence relation as follows:

$$C'(X_t, X_m) = \begin{cases} 0, & \text{if} X_t = \emptyset \\ \left(c_{ij}^k + C'(X_t - \{i\}, X_m - \{j\}) \mid (i, j) = g(X_t, X_m)\right), & \text{otherwise} \end{cases} \quad (2.15)$$

The matching assignment pairs are collected similarly:

$$Z'(X_t, X_m) = \begin{cases} \emptyset, & \text{if} X_t = \emptyset \\ \left((i, j) \cup Z'(X_t - \{i\}, X_m - \{j\}) \mid (i, j) = g(X_t, X_m)\right), & \text{otherwise} \end{cases} \quad (2.16)$$

Consequently, this strategy results in the following approximation of C_{min}^k:

$$\hat{C}_{min}^k = C'(\{i \mid 1 \leq i \leq M\}, \{j \mid 1 \leq j \leq m_{k+1}\}) \quad (2.17)$$

and the set of assignment pairs as defined in Eq.2.3:

$$Z^k = Z'(\{i \mid 1 \leq i \leq M\}, \{j \mid 1 \leq j \leq m_{k+1}\}) \quad (2.18)$$

Additionally, the algorithm differentiates between two cases: 1) all measurements are present and 2) some measurements are missing, by occlusion or otherwise. In the first case, the algorithm works as described above. Otherwise, because there is a lack of measurements at t_{k+1}, the problem is not which measurement should be assigned to which track head, but which track head should be assigned to which measurement. Then, the assignment strategy is similar to the above. When all track head assignments T_i^k to measurements $\mathbf{x}_j^k$ are found, it is clear for which tracks a measurement is missing. The R&S algorithm directly fills in these points with extrapolated points. The disadvantage of this track continuation scheme becomes apparent when the point occlusion lasts for a number of frames. Direct extrapolation results in a straight extension of the last recognized motion vector, which, in the long term, can deviate much from the true motion track so that recovering becomes increasingly difficult (see the experiments in Section 2.6.2).

C&V Algorithm (im2/cm2)

The third and last scheme we describe has been developed by Chetverikov and Verestóy [1]. Their method allows for track initiation, track termination, and occlusion only during two time instances. C&V assume the smooth motion model (**im2**) and **cm2** as combined motion model. The algorithm extends track heads T_i^k by first collecting all candidate measurements

$\mathbf{x}_j^{k+1}$ in the circle with radius d_{max} around $\mathbf{x}_i^k$ whose criterion does not exceed ϕ_{max}. The candidate measurements are considered in optimal criterion order with respect to the track head. Then, for each measurement all competing track heads are collected. The candidate measurement will be rejected if it is the best alternative for any of the competing track heads. When there are no candidates left, the track head will not be connected. Remaining unconnected track parts, caused by occlusion or otherwise, are handled in a post-processing step, which we leave out of the discussion.

This scheme does not maximize the cost of the alternatives (i.e. $w_1 = 0$ in Eq.2.8) and track heads are only considered as competitors if they are within the d_{max} as well as ϕ_{max} range. Moreover, their cost is not averaged as in Eq.2.8: *any* competitor that fits a measurement *best*, prevents that the measurement is assigned to T_i^k.

The basics of this algorithm can be summarized as follows[2]. Let $X_a(i)$ be the set of alternative track head extensions for track head T_i^k as defined below:

$$X_a(i) = \left\{ j \in X_m \mid i \in X_c(j), \forall p \in X_c(j) \left(j = \arg\min_{q \in X_m} c_{pq}^k \Longrightarrow p = i \right) \right\}, \tag{2.19}$$

where each measurement $\mathbf{x}_j^{k+1}$ has a set of competing track heads $X_c(j)$ according to:

$$X_c(j) = \left\{ i \in X_t \mid \left\| \mathbf{x}_j^{k+1} - \mathbf{x}_i^k \right\| < d_{max}, c_{ij}^k < \phi_{max} \right\} \tag{2.20}$$

The algorithm selects a measurement from $X_a(i)$ for a track head from X_t according to:

$$g(X_t, X_m) = \left((i, j) \mid i \in X_t, j = \arg\min_{q \in X_a(i)} c_{iq}^k \right) \tag{2.21}$$

Substituted into Eq.2.15-2.18, this leads to the minimal combined criterion approximation and the corresponding set of assignment pairs Z^k.

The advantage of this scheme is that the d_{max} parameter is exploited very efficiently. With low point densities, there is usually just one candidate point and there are no competing track heads for that point. However, higher point densities or large d_{max} values can reveal the inadequacy of the strategy to find the minimal combined motion model deviation. Because the deviation is not averaged over competitors and alternatives, greedy assignment decisions are the result.

2.5 Optimal Algorithm to Minimize C^k

In the previous section we saw that known algorithms adopt a sub-optimal search strategy to minimize C^k. In this section, we propose an algorithm that finds the minimum of the

[2]We describe the algorithm in our own terms assuming a fixed number of points and verification depth = 2, see [1] for details.

combined motion models efficiently. To this end, we use the Hungarian algorithm, which efficiently finds the solution of the classical assignment problem [13]. Danchick and Newman [6] first used this algorithm in a similar context; to find hypotheses for the Multiple Hypothesis Tracker. In general, the algorithm minimizes the following expression:

$$C = \sum_{i=1}^{m} \sum_{j=1}^{m} a_{ij} w_{ij} \tag{2.22}$$

subject to:

$$\sum_{i=1}^{m} a_{ij} = 1, \quad 1 \leq j \leq m; \quad \sum_{j=1}^{m} a_{ij} = 1, \quad 1 \leq i \leq m; \quad a_{ij} \in \{0, 1\}$$

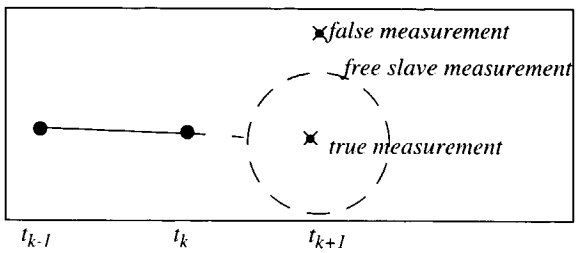
It typically finds the minimal cost assignment, which can be represented in a weighted bipartite graph consisting of two sets of vertices, X and Y. The m vertices from X are connected to all m vertices of Y with weighted edges w_{ij}. The algorithm then assigns every vertex from X to a separate vertex in Y in such a way that the overall cost is minimized.

In order to be able to apply the Hungarian algorithm and to handle detection errors and occlusion, we prepare the measurement data such that the problem becomes squared. We propose to handle the false detection problem by introducing *false tracks* as proposed earlier in [26]. False tracks do not have to adhere to any motion criterion, so that measurements that do not fit the motion model of any true track will be moved to these false tracks. By associating a maximum cost deviation (ϕ_{max}) with assignments to false tracks, we even recognize false measurements if other measurements are missing.

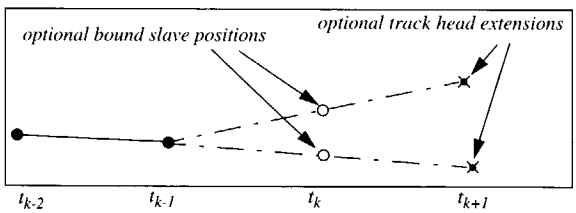
We propose to implement track continuation by introducing the concept of *slave measurements* (Fig. 2.4(a)), similar to the interpolation scheme in [26]. Slave measurements have two states: free and bound. A *free* slave is not willing to be assigned to a track. Consequently, it has a maximum deviation cost from the optimal motion track. Free slave measurements serve similar goals as the phantom points in [21]. A slave measurement is *bound* when it has been assigned to a track, despite its high deviation. Bound slaves imitate the movements of their neighboring measurements. One calculates their position by interpolating the positions of the preceding and succeeding measurements in the track established so far (Fig. 2.4(b)). The interpolated positions enable a more accurate calculation of the motion criterion. In this way, we retain as much motion information as possible and we are therefore able to find plausible correspondences. Additionally, we assign a high cost ($> \phi_{max}$) to correspondences that have d_{max} exceeded. This ensures that in such cases, a slave measurement is preferred over a measurement that does not fit the model constraints.

Greedy Optimal Assignment (GOA) Tracker: Formal Description

To properly handle missing and false measurements, we extend the assignment matrices A^k. That is, we want to be able to assign false measurements to false tracks and slave measurements to true track heads that have no measurement at t_{k+1}. Since all measurements can be



(a)



(b)

Figure 2.4: (a) shows a true measurement, a false measurement and a free-slave measurement at t_{k+1}. The slave measurement is on the border of the dotted circle. (b) shows possible bound-slave measurement positions related to possible track-head extensions.

false and the measurements of all track heads can be missing, we add m_{k+1} rows to allow for m_{k+1} false tracks, and we add M columns to allow for M slave measurements, resulting in the definition of the square matrix A_λ^k (resembling the dummy rows and columns in the validation matrix as proposed in [9]).

The size of the individual criterion matrix is adjusted similarly. The entries in the m_{k+1} extra rows and in the M extra columns all equal the maximum cost resulting in cost matrix D_λ^k.

Having defined these square matrices, we can solve the linear assignment problem for one frame after the other, assuming that the correspondences between the first two frames are given (in case $O_{im} > 1$) so that the initial velocity vector can be computed.

In order to calculate the motion criterion, the individual motion models with $O_{im} = 2$ need the vector $(\mathbf{x}_i^k - \mathbf{x}_{\alpha_i^k}^{k-1})$ and all need $(\mathbf{x}_j^{k+1} - \mathbf{x}_i^k)$. If either of $\mathbf{x}_{\alpha_i^k}^{k-1}$ or $\mathbf{x}_i^k$ is a slave measurement, we estimate these vectors by scanning back in T_i^k to collect two true measurements in the nearest past being $\mathbf{x}_{\alpha_i^{k\to p}}^p$ and $\mathbf{x}_{\alpha_i^{k\to q}}^q$ respectively, with $1 \le p < q \le k$ and $\alpha_i^{k\to q}$ means $k-q$ times recursive application of α_i^k. Consequently, the vector estimates are defined as follows:

$$\mathbf{x}_i^k - \mathbf{x}_{\alpha_i^k}^{k-1} = \frac{\mathbf{x}_{\alpha_i^{k\to q}}^q - \mathbf{x}_{\alpha_i^{k\to p}}^p}{q-p}; \qquad \mathbf{x}_j^{k+1} - \mathbf{x}_i^k = \frac{\mathbf{x}_i^{k+1} - \mathbf{x}_{\alpha_i^{k\to q}}^q}{k+1-q} \tag{2.23}$$

Having obtained these velocity vector estimates, we can now compute the individual motion criteria c_{ij}^k. We transform the criterion matrix to a bipartite graph and prune all edges with weights that exceed ϕ_{max}. Then, to satisfy the combined motion model, we adjust the edge weights w_{ij} as defined below.

cm1 average deviation model:

$$w_{ij} = \left(c_{ij}^k\right)^z \tag{2.24}$$

cm2 average deviation conditioned by competition and alternatives, using Eq. 2.13;

$$w_{ij} = c_{ij}^k - w_1 R_a'(i) - w_2 R_c'(j) \tag{2.25}$$

As mentioned before, the actual value of the minimized C^k is not important. Therefore in **cm1**, the $1/z$ power can be ignored, because the $1/z$ power function is monotonic increasing.

Algorithm

1. Starting with $k = O_{im}$ compute all costs c_{ij}^k in the cost matrix D_λ^k as follows:
   (a) true tracks to true measurements, i.e. $1 \le i \le M$, $1 \le i \le m_{k+1}$:
   If the maximum speed (d_{max}) constraint is violated then $c_{ij}^k = \phi_{max} + \epsilon$.
   Otherwise c_{ij}^k is calculated according to the individual motion model.
   (b) all other entries: $c_{ij}^k = \phi_{max}$
2. Construct a bipartite graph based on the criterion matrix D_λ^k.
3. Prune all edges that have weights exceeding ϕ_{max}.
4. Adjust the edge weights according to the combined motion model in Eq.2.24 and 2.25.
5. Apply the Hungarian algorithm to this graph, which results in the minimal cost assignment. The resulting edges (assignment pairs) correspond to an output A_λ^k, from which the first M rows and m_{k+1} columns represent the assignment matrix A^k.
6. Increase k; if $k < n$ go to 1, otherwise, done.

2.6 Performance Evaluation

To evaluate the performance of the different algorithms, we compared them qualitatively and quantitatively. In Section 2.6.1, we start by looking at their correspondence quality by using a specially constructed example that (also) tests the algorithm's track continuation capabilities. Then, in Section 2.6.2, we explore the sensitivity of the algorithms to some problem parameters like the point density and the total number of points, and algorithm parameters like d_{max}. In all experiments in this section, the correspondences between the first two frames are known and passed on to all algorithms (even to those that are capable of self-initialization to avoid that one of the methods is favored).

2.6.1 Constructed Example

The carefully constructed example shows two crossing feature points with a missing measurement at t_4 for the first and at t_5 for the second point (see Fig. 2.5(a)). The difficulty of this data set is that in two consecutive frames a measurement is missing, but for different points. With all algorithms we used the smooth motion model (**im2**). For algorithms that have a ϕ_{max} parameter; we varied its value from 0.05 to 1 (lower values do not allow the initial motion of p_2). Further, we fixed the d_{max} value to 20.

S&S Results

The S&S algorithm either leads to wrong correspondences or to disconnected track parts. We used two different settings of ϕ_{max} to show the shortcomings of S&S. First, with a high ϕ_{max} ($0.1 \leq \phi_{max} \leq 1$), the algorithm makes wrong correspondences (Fig. 2.5(b)). When assigning measurements to track heads T_i^k, the algorithm prefers track heads that have a true measurement at t_k over track heads that have a phantom point at t_k. Of course the motion criterion for that true measurement assignment may not exceed the maximum criterion. On the other hand, if ϕ_{max} is lower (e.g. 0.05), the algorithm separates four track parts, while correspondences between the track parts have to be found afterwards (see Fig. 2.5(c)).

R&S Results

The R&S algorithm, which has no parameters, chooses the right correspondence when one measurement lacks at t_4. Then, it estimates the missing measurement by extrapolation and continues with the next frame. With point extrapolation for one frame only, the deviation is limited. In the next frame (t_5) the situation is similar to the previous frame. The algorithm connects the single present measurement to the right track head and extrapolates the missing measurement. The processing of the last 3 frames is straightforward (see Fig. 2.5(d)).

C&V Results

At t_4, C&V assigns the single measurement to the right track head (T_2^3). Then at t_5 only one track head (T_2^4) remains to which the measurement can be assigned. If it did not fit because

the distance was too great, this measurement could start a new track. Since it is not too far away, the only point at t_5 is also assigned to T_2. The two track parts that belong to p_1 are not connected in the post-processing step (Fig. 2.5(e)).

GOA Tracker Results

When the algorithm proposed in this paper is applied to this data set with the smooth motion and average deviation model, all correspondences are made correctly. Moreover, the algorithm interpolates the missing measurements better than R&S and, hence, forms the most plausible tracks (see Fig. 2.5(f)).

2.6.2 Performance with Generated Data

In this section we describe the tests we carried out to evaluate various aspects of the described algorithms. To this end we used a data set generator that is able to create data sets with uncorrelated random point tracks of various densities and speeds. Among the described algorithms only the R&S algorithm does not exploit the d_{max} parameter to improve the quality and efficiency. For the experiments, we added the d_{max} parameter to R&S (now called R&S*)[3] similar to the GOA tracker. Then, we tuned all algorithms to find the optimal d_{max} setting for each of them and used that setting in all experiments. For C&V, R&S*, and the GOA tracker the true maximum is optimal and for S&S a very high value, $d_{max} = 50$, is optimal. In Section 2.6.2, we consider the sensitivity of the algorithms for the d_{max} parameter setting. We did not test the ϕ_{max} sensitivity, because it constrains the motion similarly. Other experiments evaluate the performance for increasingly difficult data sets, an increasing number of missing point detections, and the efficiency of the algorithms.

For the generation of the uncorrelated tracks, we used the data set generator called Point Set Motion Generator (PSMG) according to [27] (see example in Fig. 2.6). Because this data generator model allows feature points to enter and leave the 2-D scene, which we do not consider in this paper, we modified the model to prevent this by replacing invalid tracks until all tracks were valid. The PSMG has the following parameters (defaults in brackets):

1. Number of feature point tracks ($M = 50$)
2. Number of frames per point track ($n = 8$).
3. Size ($S_w = S_h$) of the square space ($S = 100$).
4. Uniform distributions for both dimensions of initial point positions between 0 and S.
5. Normal distribution for the magnitude of the initial point velocity vector: $v_i^0 = N(\mu_{v_0} = 5, \sigma_{v_0} = 0.5)$
6. Uniform distribution for the angle of the initial velocity vector, between 0 and 2π.

[3] If d_{max} is very high, then R&S* behaves like the original R&S, i.e. unconstrained speed.

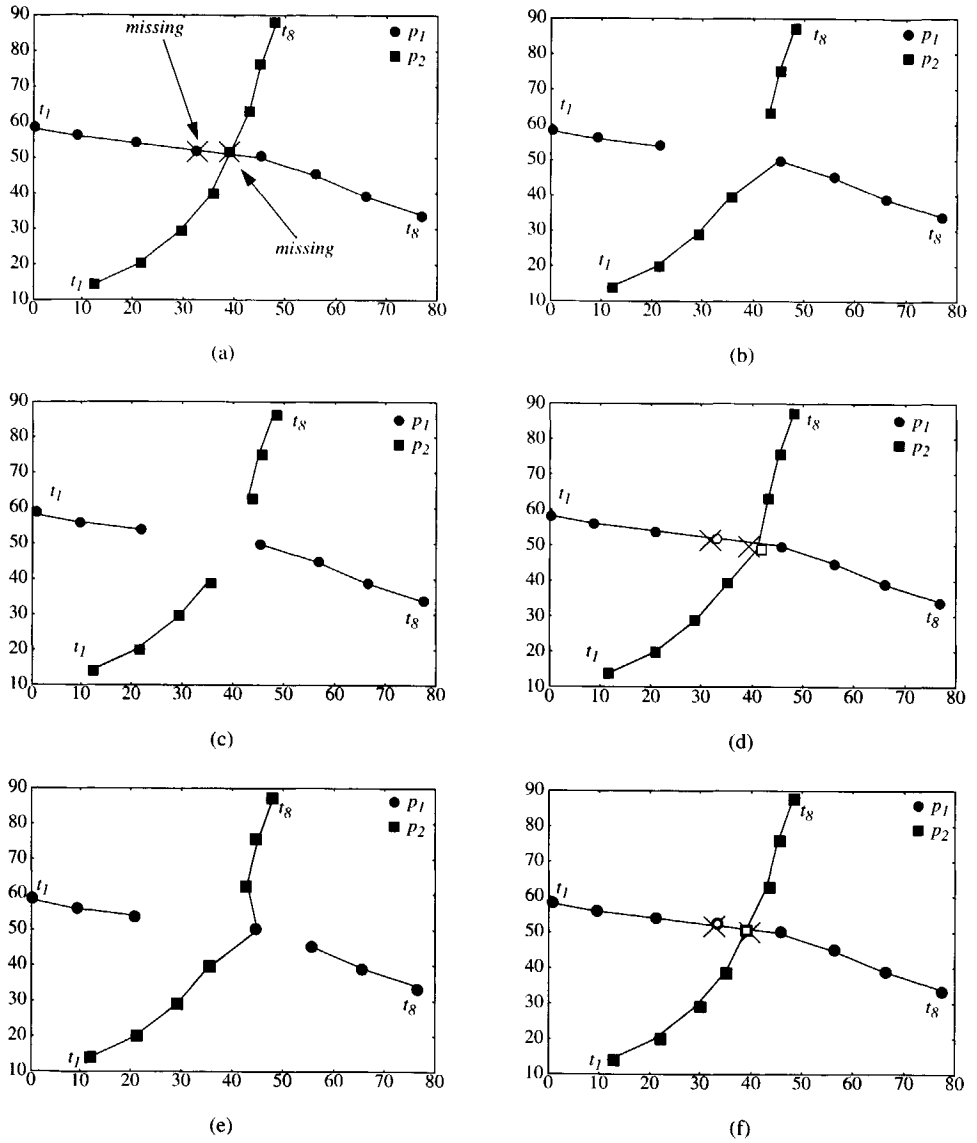


Figure 2.5: (a) Two input measurements at 8 time instances. At t_4 a measurement for point p_1 is missing and at t_5 a measurement of point p_2 is missing. The figures show the results of (b) S&S using $0.1 \leq \phi_{max} \leq 1$, (c) S&S using $\phi_{max} = 0.05$, (d) R&S, (e) C&V, and (f) the GOA tracker respectively. In the figures the estimated points are shown as nonfilled boxes and crosses indicate the true positions of the missing points.

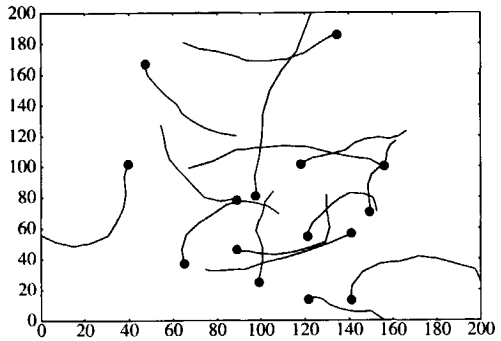


Figure 2.6: Example PSMG data set with 15 points during 8 time steps.

7. Normal distribution for the *update* of the velocity vector magnitude v_i^k, from t_k to t_{k+1}: $v_i^{k+1} = N(v_i^k, \sigma_{v_u} = 0.2)$
8. Normal distribution for the *update* of the velocity vector angle β_i^k from t_k to t_{k+1}: $\beta_i^{k+1} = N(\beta_i^k, \sigma_{\beta_u} = 0.2)$
9. Probability of occlusion ($p_o = 0$, i.e. no occlusion)

A number of different measures have been proposed to quantify the quality of performance, like the *distortion measure* [19] and the *link-based error* and *track-based error* [27]. We use the track-based error as in [27], which is defined as follows:

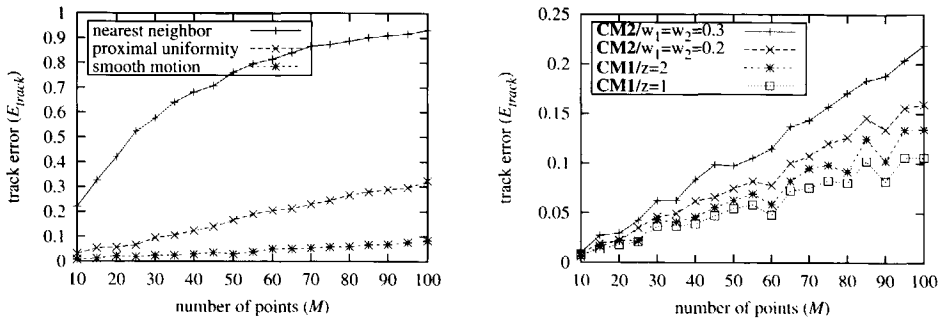
$$E_{track} = 1 - \frac{T_{correct}}{T_{total}}, \tag{2.26}$$

where T_{total} is the total number of true tracks and $T_{correct}$ is the number of *completely* correct tracks.

Some remarks about the experiments. First, in all cases the shown results are an average of 100 runs. We did not incorporate significance levels because the minimal possible track error depends on the actual presented data, and hence, on the appropriateness of the individual motion model. Nevertheless, the ranking and relative quality for each experiment were the same as those illustrated in the figures. Second, in this section we ran the S&S algorithm only with a forward optimization loop, because otherwise the algorithm would not converge (see also Section 2.4).

Tuning Individual and Combined Models

To find an optimal combination of individual and combined motion models, we assume that the individual models and combined models are independent. In order to find the best individual model for the PSMG generated data, we ran experiments with the individual models



(a) (b)

Figure 2.7: (a) Track error of the GOA tracker with the average deviation model (**cm1**), in combination with the nearest neighbor, smooth motion or proximal uniformity model. (b) Track error of the GOA tracker with the smooth motion model in combination with **cm1** and **cm2**.

im1, **im2** and **im3**, together with the combined model **cm1**/$z = 1$ implemented in the GOA tracker. In Fig. 2.7(a), we show the results of this experiment. Clearly, the model **im2** fits this generated data set best.

In order to identify the best combined model for this data set, we ran tests with **cm1**/$z = 1$, **cm1**/$z = 2$, **cm2**/$w_1 = w_2 = 0.3$ and **cm2**/$w_1 = w_2 = 0.2$ as shown in Fig. 2.7(b). We chose w_1 equal to w_2, because we want to express that the lack of alternatives is equally important as the absence of competing track heads. **cm2** with even lower w_1 and w_2 values becomes better until it finally equals **cm1**/$z = 1$ when $w_1 = w_2 = 0$. From these tests we conclude that the smooth motion model ($\phi_{max} = 0.2$) with average deviation model ($z = 1$) is the best combined modeling for PSMG data. Hence, we used these models in the remaining experiments, if possible. That is, only the GOA tracker allows for *combined model* settings and can be adjusted in that sense.

Variable Density Performance

To show how the algorithms perform with an increasing number of conflicts, we applied them to several data sets with an increasing point density. To this end, we generated the data in a fixed sized 2-D space and varied the number of point tracks. In Fig. 2.8(a), we display the results of all algorithms. The figure clearly shows that the GOA tracker performs best.

Variable Velocity Performance

Another experiment to test the tracking performance of the algorithms is varying the mean velocity and keeping the number of points constant. In order to obtain reasonable speed variances with all mean velocities, we scaled both σ_{v_0} and σ_{v_u} with the mean values according

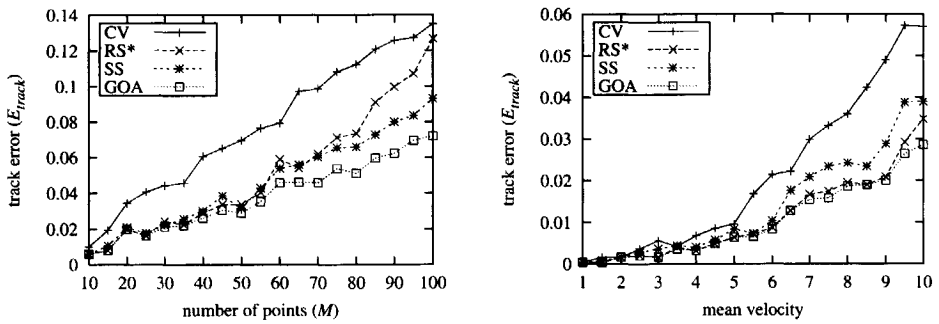


Figure 2.8: (a) Results of the algorithms applied to increasingly dense point sets. (b) Track error as a function of the mean velocity.

to $\sigma_{v_0} = 0.1\mu_{v_0}$ and $\sigma_{v_u} = 0.04\mu_{v_0}$. In addition, we enlarged the space in which the point tracks are generated to $S = 200$, to prevent that mainly diagonal tracks are allowed. The ranking of the algorithms is similar to the variable density experiment and again the GOA tracker performs better than all other schemes (see Fig. 2.8(b)).

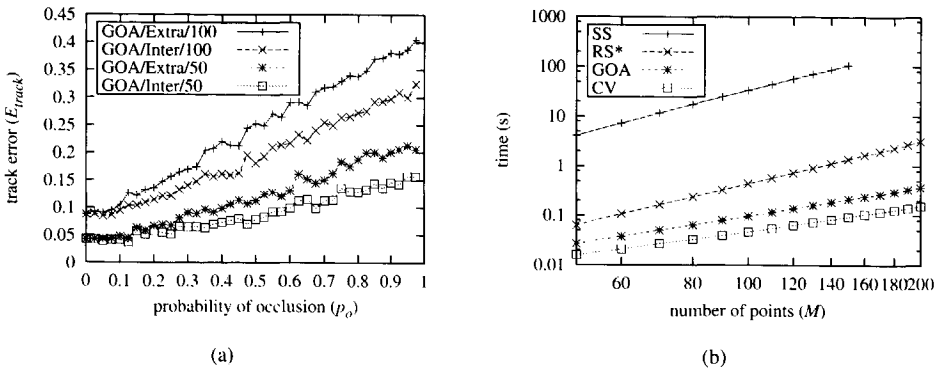
Track Continuation Performance

In this experiment, we compared the track continuation performance of the R&S extrapolation scheme and the slave measurements interpolation, as proposed in this paper. We left out the other two algorithms because S&S does not really handle track continuation and C&V only allows very limited occlusion. In order to properly compare the track extrapolation and the slave measurements interpolation, we implemented them both in the GOA tracker. We tested the track continuation performance in a variable occlusion experiment, with $0 \leq p_o \leq 1$. In Fig. 2.9(a), we display the track error results of the GOA tracker with both track continuation schemes with either 50 or 100 points.

As illustrated in this figure, the slave measurements approach proposed in this paper clearly achieves better track continuation results than the track extrapolation scheme as proposed by Rangarajan and Shah [19]. The difference between the approaches is larger with a higher probability of occlusion (p_o), because then occlusion during a number of consecutive frames will occur more often, in which case the difference between interpolation and extrapolation becomes apparent.

Variable Volume Performance

This test is directed towards the measurement of the computational efficiency of the different algorithms. Hereto, we keep the point density constant while increasing the number of point



(a) (b)

Figure 2.9: (a) Track error of the GOA tracker with either slave interpolation (Inter) or the R&S extrapolation scheme (Extra) in a variable occlusion experiment with 50 or 100 points. (b) Illustration of the efficiency of the algorithms in a variable volume experiment.

tracks (and thus enlarging the size of the 2-D space proportionally). Consequently, the correspondence problem remains equally difficult, but the problem size grows. In Fig. 2.9(b), we show the results with logarithmically scaled axes. The figure shows that, with optimal d_{max}, C&V is the fastest. Further, the computation time of the algorithms is widely divergent but all algorithms have polynomial complexity. We list the polynomial orders in the summary of the experiments in Section 2.6.3.

Sensitivity to d_{max} Parameter Setting

As mentioned, up to this point all algorithms used the tuned and optimal settings of the d_{max} parameter. In this sensitivity experiment, we show the importance of the a priori knowledge about a reasonable value for this parameter. To this end, we varied the d_{max} parameter from the known true value up to a high upper limit, $d_{max} = 50$ (lower values than the true maximum speed are clearly not sensible). Fig. 2.10(a) clearly shows that both S&S and R&S* are most sensitive to variations in this parameter. Remarkably, S&S performs better when d_{max} is set far too high. We expect that the ill initialization, together with the exchange optimization causes this effect because every point exchange must obey the d_{max} constraint. Both C&V and the GOA tracker are hardly sensitive to d_{max} variations (which implies that they do not take advantage of it either). Computationally, especially the C&V algorithm is hampered by an incorrect or ignorant d_{max} value as Fig. 2.10(b) illustrates. Consequently, the GOA tracker is the fastest when d_{max} is over 5 times the true maximum speed.

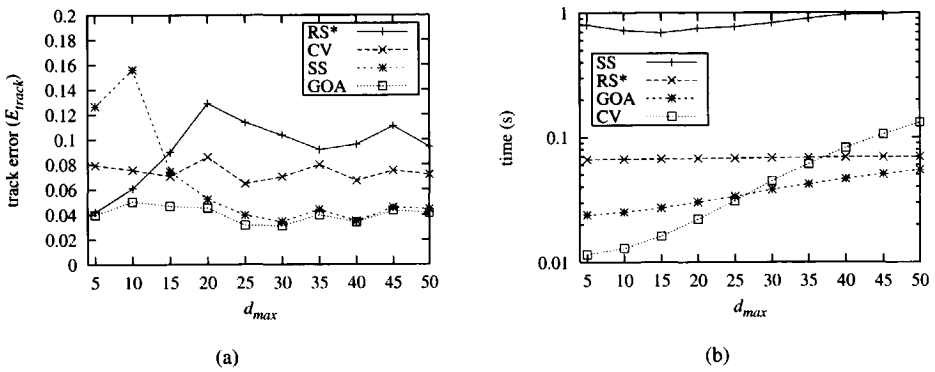


Figure 2.10: Illustrates the sensitivity of the algorithms to d_{max} variations. (a) shows the track error performance and (b) shows the computational performance.

2.6.3 Summary of Experiments

In conclusion, for tracking a fixed number of points the GOA tracker is qualitatively the best algorithm among those we presented, according to its track continuation handling in the first test and its performance in all PSMG experiments. Moreover, it is hardly sensitive to the d_{max} parameter setting. S&S performs only slightly worse when we used the optimal d_{max} setting ($d_{max} = 50$), but it is an order of magnitude slower than the GOA tracker. Moreover S&S did not perform well on the specially constructed example, nor does it give interpolated positions of the missed points. The version of R&S*, with an added d_{max} parameter and modified individual model, is efficient and qualitatively good as long as it has an accurate estimate of d_{max}. The sensitivity experiment shows that R&S* performs worst of all algorithms if this value is not known (or not used as in the original R&S implementation). With (near) optimal maximum velocity setting, C&V is the fastest. If this optimal value is not known (which is usually the case), then the efficiency of C&V degrades rapidly. We should also note that, in our experiments, S&S performed consistently better than C&V, which does not agree with the results reported in [27]. This is probably because in [27] a different d_{max} setting is used for S&S, to which this algorithm is quite sensitive, as we showed in Section 2.6.2. This implies that S&S cannot exploit the d_{max} parameter effectively to handle missing and spurious measurements. Finally, the variable occlusion experiment clearly showed that the slave measurements implement track continuation better than the point extrapolation scheme [19]. In Table 2.1, we summarize the PSMG experiments. The last column shows the polynomial order of complexity of the algorithms as derived from the variable volume experiment.

Table 2.1: Summary of the PSMG experiments.

Algorithm	variable density ($M = 100$) E_{track}	variable velocity ($\mu_{v_0} = 10$) E_{track}	variable volume ($M = 100$) time (s)	polynomial order a in $O(M^a)$
GOA	0.07	0.029	0.097	1.9
S&S	0.09	0.039	34	3.0
C&V	0.14	0.057	0.047	1.6
R&S	0.13	0.035	0.44	2.8

2.7 Algorithm Extension with Self-Initialization

In the problem statement in Section 2.2, the correspondences between the first two frames were assumed to be known. In this section, we generalize the problem by lifting this restriction and elaborate on how self-initialization is incorporated in the GOA tracker.

Two algorithms we discussed have an *integrated* way of automatically initializing the point tracks. That is, both S&S and C&V only use the measurement positions for the initialization. R&S, on the other hand, uses additional information, i.e. the optical flow field, which is computed between the first two frames. We advocate the integrated approach, because it is more generally applicable and it allows for optimizing the initial correspondences using a number of frames, as we proposed in the *global motion model* in Section 2.3. Here, we propose to extend the GOA tracker with features of the S&S algorithm. After that, we demonstrate the appropriateness of this extension and again analyze the parameter sensitivity of the algorithms that support self-initialization.

2.7.1 Up-Down Greedy Optimal Assignment Tracker (GOA/up-down)

The S&S algorithm has a number of shortcomings, of which the computational performance has been shown to be the most apparent. Also, as mentioned, we deliberately left out the bi-directional optimization, which quite often does not converge. However, for self-initialization the bi-directional optimization is essential.

We propose to modify the GOA tracker in the spirit of [21] and [22] by initializing the correspondences between the first two frames using the described optimal algorithm to minimize C^k with **iml/cml**. After these correspondences are made, we continue the optimization of the remaining frames (*up*) in the normal way and additionally optimize the same frames backwards (*down*). Further forward and backward optimization proved to be useless, because the optimization process already converged. The reason for this fast convergence is that both

the initial correspondences and the optimization scheme have been improved considerably compared to S&S.

2.7.2 Self-Initialization Experiments

To test the performance of the algorithms that are capable of self-initializing the tracks, together with the just described extended GOA/up-down tracker, we did another variable density experiment, and a sensitivity experiment using the PSMG track generator. The individual models need not be tuned again because the parameter settings of the PSMG are the same as in Section 2.6.2. This time, we left out S&S because of serious convergence problems with their bi-directional optimization scheme, which is essential for self-initialization. R&S does not implement self-initialization using only point measurements, so it cannot be applied within these experiments.

Although we did not discuss statistical motion correspondence techniques in detail in this paper, we included the multiple hypothesis tracker (MHT) as described and implemented by Cox and Hingorani [3] in this experiment in order to see how it relates to non-statistical greedy matching algorithms. We should note that this MHT implementation is not the most efficient (for improvements see e.g. [15]), though qualitatively equivalent to the state of the art of the statistical motion correspondence algorithms.

Variable Density Experiment

For this experiment we tuned the algorithms optimally for the given data sets. That is, both C&V and GOA/up-down use the true d_{max}. The (eight) parameters of the MHT (like the Kalman filter and Mahalanobis distances), were tuned with a genetic algorithm, for which we used the track error as fitness function.

Actually the only difference with the variable density experiment in Section 2.6.2 is that here the initial correspondences are not given. Fig. 2.11(a) shows the performance of the algorithms. Clearly, GOA/up-down performs best and, remarkably, almost as good as when the initial correspondences were given. The performance of the MHT is similar to the GOA tracker until it seriously degrades, when the number of points exceeds 50, see Fig. 2.11(a). This can be explained from the fact that the parameters for the MHT were trained for (only) 50 points. We did not include more points, because the training was already very time consuming ($>$ 2 days on a Silicon Graphics Onyx II). It is, however, striking to see that the GOA tracker also performs consistently better than the MHT even with less than 50 points, although the latter optimizes over several frames. Among others this may be caused by the effective self-initialization scheme of the GOA tracker. The up-down scheme can be said to optimize the initial correspondences over the whole sequence when optimizing up. Then the remaining correspondences are established in the down phase.

Sensitivity Experiment

When the correspondences for the initial frames are not given, we expect the algorithms to be more sensitive to the d_{max} setting. Namely, when the initial velocity is unconstrained,

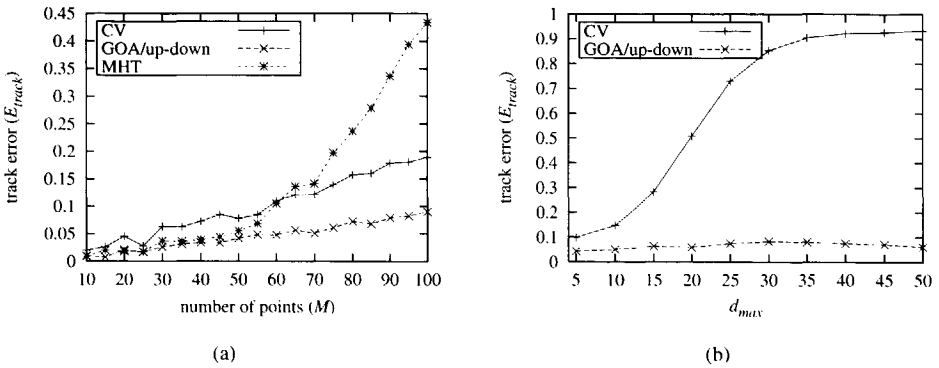


Figure 2.11: (a) shows the track error as a function of the number of points in a variable density experiment with self-initialization, and (b) shows the track error as a function of the d_{max} setting.

the greedy matching algorithms can easily make implausible initial choices from which they cannot recover. To study this behavior, we did another sensitivity experiment for C&V and the GOA tracker and additionally an experiment to test the sensitivity of the MHT. We studied the MHT separately, because is has different parameters (and no d_{max}). First, Fig. 2.11(b) indeed shows that for C&V a good estimate of d_{max} is essential. GOA/up-down, however, hardly suffers from a lack of a priori knowledge concerning d_{max}, which is partially because the *global cost* was optimized for the initial frames. Moreover, the up/down optimization scheme can no longer be considered purely greedy, because correspondences are reconsidered in the backward direction. Since both algorithms were computationally influenced similarly as when the initial correspondences were given, we did not include the figure here. We have to mention, however, that the computation time of C&V increased even 10 times faster (11 sec. when $d_{max} = 50$), because in this experiment the number of alternatives becomes much higher in the first frame. As a consequence, the GOA tracker was already the fastest when d_{max} was set over 3 times the true maximum speed.

In order to fairly test the sensitivity of the MHT and to show the results for all parameters in the same figure, we tested the performance in the range from 1/5 of the optimal setting to 10 times the optimal setting of all essential parameters (10 runs per setting). Consequently, the results in Fig. 2.12(a) can easily be compared in relation to Fig. 2.11(a), in which 5 (d_{max}) is also optimal. The figure clearly shows that there is a small parameter range in which the performance is (sub)optimal. Especially increasing or decreasing the Mahalanobis distance or the initial state variance parameter with 1/5th results in a performance penalty of roughly a factor two. Also the computation time increases dramatically if the parameters are not properly set, as Fig. 2.12(b) shows. We plotted the names of the essential parameters in the figures, but refer to [3], [20] for a complete description.

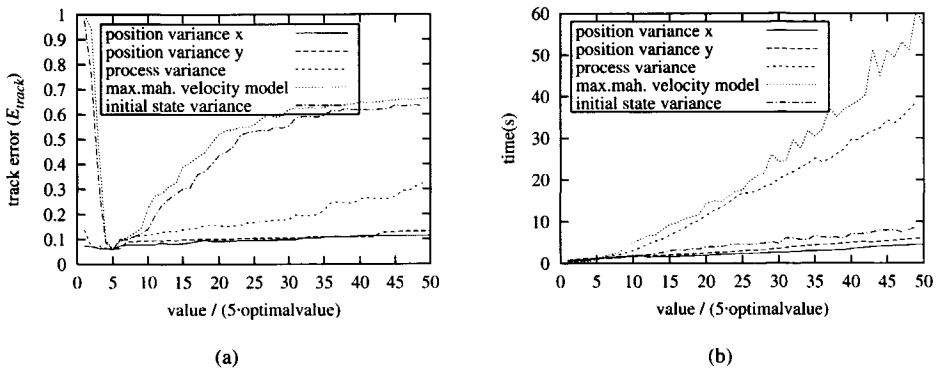


Figure 2.12: Parameter sensitivity experiment for the MHT. (a) shows the track error as function of parameter variations and (b) shows the computation time.

2.8 Real Data Experiment: Tracking Seeds on a Rotating Dish

Our final experiment is based on real image data. In this experiment we put 80 black seeds on a white dish and rotated the dish with a more or less constant angular velocity, which implies the use of the smooth motion model (**im2/$\phi_{max} = 0.1$**). The scene was recorded with a 25 Hz progressive scan camera using 4 ms shutter speed, resulting in a 10 image video sequence with very little motion blur[4]. The segmentation of the images was consequently rather straightforward, i.e. in all 10 images all 80 seeds were detected and there were no false measurements. There was a large difference in speed of the seeds ranging from 1 pixel/s in the center to 42 pixel/s at the outer dish positions. Like in Section 2.7, we tested only those algorithms that have self-initialization capability and again we included the MHT. Clearly, in contrast with Section 2.7, in this experiment the point motion is strongly dependent. Since all algorithms are hampered equally, this experiment actually tests the general applicability of the algorithms[5]. To be able to run the MHT properly we tuned its main parameters by applying a genetic algorithm (for which the ground truth was established by manual inspection.) For this experiment we also added the S&S algorithm, because this time it converged consistently, that is, with different d_{max} settings.

Fig. 2.13 shows the resulting tracks overlaid on the first image of the sequence. Only the GOA/ up-down tracker was able to find all the true seed tracks, while the d_{max} setting did not influence the results. Even GOA/up (not down!) was able to track the 80 seeds correctly over all 10 frames, regardless of the d_{max} value. Not surprisingly, the C&V algorithm that already proved to be sensitive to d_{max} suffers severely from the divergent seed speeds. The S&S

[4]The rotating dish sequence is available at http://www-ict.its.tudelft.nl/tracking/datasets/sequences/rotdish80.tgz.

[5]One could, of course, argue that for this data set a rotational individual model or polar coordinates for the measurement positions would fit better.

algorithm, which is also sensitive to d_{max}, again makes less errors when d_{max} is relaxed. In general the behavior of S&S turned out to depend greatly on the d_{max} and ϕ_{max} settings. Although the MHT was extensively tuned and it optimizes over several frames simultaneously, it still made a few errors. Besides, the MHT is substantially slower than the other algorithms.

2.9 Discussion

Throughout this paper, we introduced a framework for motion modeling, and we presented the greedy optimal assignment (GOA) tracker that we extended with self-initialization. In this section we discuss some potential other extensions and improvements.

Although the tracking of a variable number of points conflicts with occlusion handling, it is certainly a feature that should be considered as an extension to the GOA tracker. Among the described algorithms we have seen two ways to approach this conflict of requirements, either by actually not implementing track continuation (S&S) or by only allowing occlusion during a very limited number of frames (C&V). First, the GOA tracker can support track initiation and termination by replacing the slave measurements with the phantom points as in S&S. Alternatively, the GOA scheme can be incorporated in the C&V algorithm. The idea is that at each time instance, the GOA scheme is applied first to find corresponding measurements for all point tracks that have been established so far. Then, the original C&V scheme links the remaining measurements if possible. As a result the tracking features of C&V still apply and its performance increases[6].

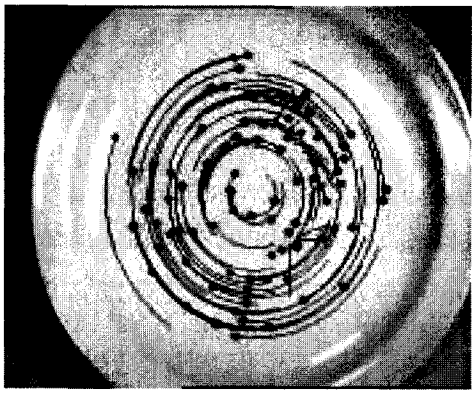
Further, to deal more effectively with the underlying physical motion, the order of the individual motion models could be increased, e.g. by modeling point acceleration. Clearly, extending the scope of the individual models implies difficulties for the model initialization and the track continuation capabilities.

Finally, the scope over which the global matching is approximated can be extended. In this paper, we approximated $S(D)$ in a greedy sense, i.e. we only minimized the combined model over two successive frames. We already illustrated in Fig. 2.3 that extending the scope for this minimization would yield more plausible tracking results. However, extending the scope implies that we need to cope with an increasingly complex problem, to which the efficient Hungarian algorithm as such cannot be applied anymore.

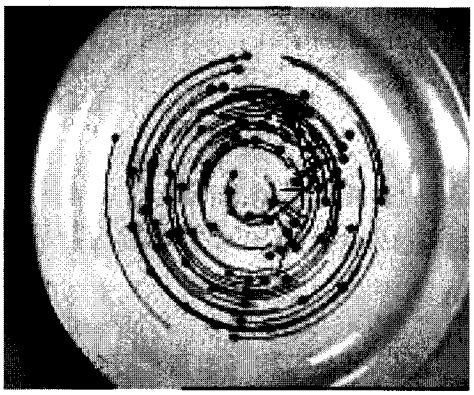
2.10 Conclusion

In this paper we showed an adequate way to model the motion correspondence problem of tracking a fixed number of feature points in a non-statistical way. By fitting existing algorithms in this motion framework, we showed which approximations these algorithms make. An approximation that all described algorithms have in common is that they greedily

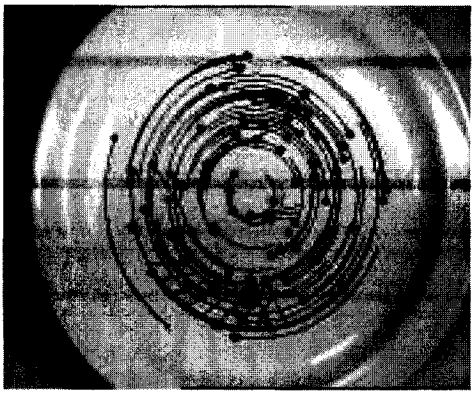
[6]We have already implemented this idea, but did not include it in the experiments for the sake of clarity. With a fixed number of points, its performance could indeed be rated in between that of GOA/up-down and that of the original C&V algorithm.



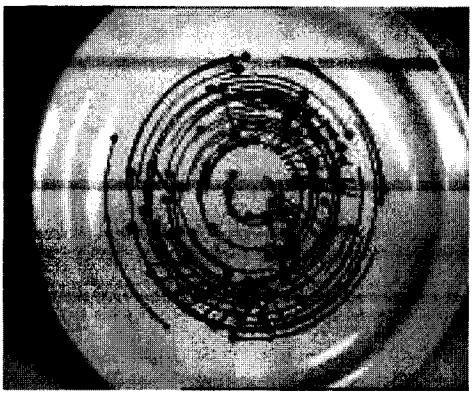
(a) S&S: d_{max} = 42 p/s: 25 errors, 7.4 sec.



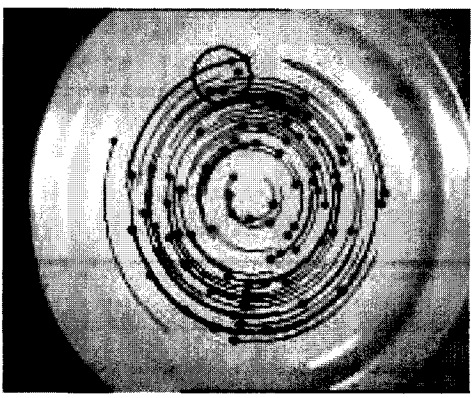
(b) S&S: d_{max} = 50 p/s: 18 errors, 13 sec.



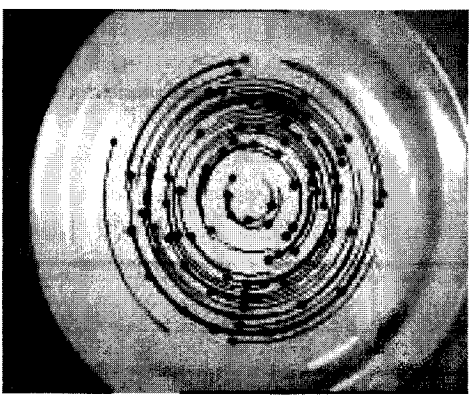
(c) C&V: d_{max} = 42 p/s: 18 errors, 44 msec.



(d) C&V: d_{max} = 50 p/s: 31 errors, 58 msec.



(e) MHT: 3 errors, 86 sec.



(f) GOA/up-down: 0 errors, 90(160) msec.

Figure 2.13: Results of applying the self-initializing algorithms to the rotating dish sequence, consisting of 10 frames with each 80 seeds; true d_{max} = 42 pixels/sec.

match measurements to tracks. For this approximation we proposed an *optimal* algorithm, the Greedy Optimal Assignment (GOA) tracker, which obviously qualitatively outperforms all other algorithms. The way the proposed algorithm handles detection errors and occlusion turned out to be effective and more accurate than the other described algorithms. Moreover, the experiments show clearly that its computational performance is among the fastest. Also the self-initializing version of the GOA tracker turned out to be adequate and hardly sensitive to the maximum speed constraint (d_{max}) setting. Briefly, for the tracking of a fixed number of feature points the proposed tracker has proven to be efficient and qualitatively the best.

Among the described algorithms the R&S algorithm is completely surpassed because it operates under the same conditions, while the GOA tracker outperforms R&S both qualitatively and computationally. The S&S algorithm, which does not support track continuation, is computationally very demanding. The major drawbacks of the C&V algorithm are its relatively poor performance, especially with respect to the initialization, its restricted track continuation capability, and its sensitivity to the d_{max} setting. Still, S&S and C&V may be considered because both support the tracking of a variable number of points and C&V can be very fast. In the previous section we indicated how their performance can be improved by incorporating GOA features in these algorithms. In a number of experiments we included the statistical multiple hypothesis tracker. Even though the MHT optimizes over several frames, which makes it computationally demanding, it turned out that it does not perform better than the GOA tracker. Possible causes are the effective initialization of the GOA tracker and the fact that the MHT models the tracking of a varying number of points, although we set the respective probabilities as to inform that the number of points is fixed. Most importantly, the MHT has quite a few parameters for which the tuning proved to be far from trivial.

In conclusion, the proposed qualitative motion framework has proven to be an adequate modeling of the motion correspondence problem. As such, it reveals a number of possibilities to achieve qualitative improvements, ranging from more specialized individual models to $S(D)$ approximations with an extended temporal scope.

Acknowledgments

This work was supported by the foundation for Applied Sciences (STW). The authors would like to thank Dr. Dmitry Chetverikov for the discussions on the details of his tracking algorithm and the anonymous reviewers for their comments and suggestions.

Bibliography

[1] D. Chetverikov and J. Verestóy. Feature point tracking for incomplete trajectories. *Computing, Devoted Issue on Digital Image Processing*, 62:321–338, 1999.

[2] I.J. Cox. A review of statistical data association techniques for motion correspondence. *International Journal of Computer Vision*, 10(1):53–66, 1993.

[3] I.J. Cox and S.L. Hingorani. An efficient implementation of Reid's multiple hypothesis tracking algorithm and its evaluation for the purpose of visual tracking. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 18(2):138–150, February 1996.

[4] I.J. Cox and M.L. Miller. On finding ranked assignments with applications to multi-target tracking and motion correspondence. *IEEE Transactions on Aerospace and Electronic Systems*, 32(1):486–489, January 1995.

[5] I.J. Cox, M.L. Miller, R. Danchick, and G.E. Newnam. A comparison of two algorithms for determining ranked assignments with application to multi-target tracking and motion correspondence. *IEEE Transactions on Aerospace and Electronic Systems*, 33(1):295–300, January 1997.

[6] R. Danchick and G.E. Newnam. A fast method for finding the exact N-best hypotheses for multitarget tracking. *IEEE Transactions on Aerospace and Electronic Systems*, 29(2):555–560, April 1993.

[7] S. Deb, K.R. Pattipati, and Y. Bar-Shalom. A new algorithm for the generalized multidimensional assignment problem. *IEEE International Conference on Systems, Man and Cybernetics; Emergent Innovations in Information Transfer Processing and Decision Making*, pages 249–254, 1992.

[8] S. Deb, M. Yeddanapudi, K. Pattipati, and Y. Bar-Shalom. A generalized S-D assignment algorithm for multisensor-multitarget state estimation. *IEEE Transactions on Aerospace and Electronic Systems*, 33(2):523–538, 1997.

[9] T.E. Fortmann, Y. Bar-Shalom, and M. Sheffe. Sonar tracking of multiple targets using joint probabilistic data association. *IEEE Journal of Oceanic Engineering*, 8(3):173–184, July 1983.

[10] K.I. Hodges. Adaptive constraints for feature tracking. *Monthly Weather Review*, 127:1362–1373, 1998.

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

[12] V.S.S. Hwang. Tracking feature points in time-varying images using an opportunistic selection approach. *Pattern Recognition*, 22(3):247–256, 1989.

[13] H.W. Kuhn. The hungarian method for solving the assignment problem. *Naval Research Logistics Quarterly*, 2:83–97, 1955.

[14] R. Mehrotra. Establishing motion-based feature point correspondence. *Pattern Recognition*, 31(2):23–30, 1998.

[15] M.L. Miller, H.S. Stone, and I.J. Cox. Optimizing Murty's ranked assignment method. *IEEE Transactions on Aerospace and Electronic Systems*, 33(3):851–861, 1997.

[16] V. Nagarajan, M.R. Chidambara, and R.N. Sharma. Combinatorial problems in multitarget tracking - a comprehensive solution. *IEE Proceedings*, 134(1):113–118, February 1987.

[17] A.B. Poore. Multidimensional assignments and multitarget tracking. In *Partitioning Data Sets; DIMACS Workshop*, pages 169–196, New Brunswick, USA, 1995. American Mathematical Society.

[18] A.B. Poore and X. Yan. Data association in multi-frame processing. In *Proceedings of the Second International Conference on Information Fusion*, volume II, pages 1037–1044, Sunnyvale, USA, 1999.

[19] K. Rangarajan and M. Sha. Establishing motion correspondence. *CVGIP: Image Understanding*, 24(6):56–73, July 1991.

[20] D.B. Reid. An algorithm for tracking multiple targets. *IEEE Transactions on Automatic Control*, 24(6):843–854, December 1979.

[21] V. Salari and I.K. Sethi. Feature point correspondence in the presence of occlusion. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(1):97–91, January 1990.

[22] I.K. Sethi and R. Jain. Finding trajectories of feature points in a monocular image sequence. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 9(1):56–73, January 1987.

[23] P.J. Shea and A.B. Poore. Computational experiences with hot starts for a moving window implementation of track maintanance. In *Proceedings of the SPIE: The International Society for Optical Engineering*, volume 3373, pages 428–439, 1998.

[24] R.Y. Tsai and T.S. Huang. Uniqueness and estimation of three-dimensional motion parameters of rigid objects with curved surface. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 6(1):13–26, January 1984.

[25] S. Ullman. *The Interpretation of Visual Motion*. M.I.T. Press, Cambridge, MA, 1979.

[26] C.J. Veenman, E.A. Hendriks, and M.J.T. Reinders. A fast and robust point tracking algorithm. In *IEEE International Conference on Image Processing*, volume III, pages 653–657, Chicago, USA, October 1998.

[27] J. Verestóy and D. Chetverikov. Experimental comparative evaluation of feature point tracking algorithms. In *Proceedings Workshop on Evaluation and Validation of Computer Vision Algorithms*, pages 183–194. Kluwer series in Computational Imaging and Vision, 2000.

Chapter 3

Establishing Motion Correspondence using Extended Temporal Scope

This chapter has been submitted to *Artificial Intelligence*, November 2000.

Abstract This paper addresses the motion correspondence problem: the problem of finding corresponding point measurements in an image sequence solely based on positional information. The motion correspondence problem is most difficult when the target points are densely moving. It becomes even harder when the point detection scheme is imperfect or when points are temporarily occluded. Available motion constraints should be exploited in order to rule out physically impossible assignments of measurements to point tracks. The performance can be further increased by deferring the correspondence decisions, that is, by examining whether the consequences of candidate correspondences lead to alternate and better solutions. In this paper, we concentrate on the latter by introducing a scheme that extends the temporal scope over which the correspondences are optimized. The consequent problem we are faced with is a multi-dimensional assignment problem, which is known to be NP-hard. To restrict the consequent increase in computation time, the candidate solutions are suitably ordered and then additional combined motion constraints are imposed. Experiments show the appropriateness of the proposed extension, both with respect to performance as well as computational aspects.

Keywords: Computer vision, feature point tracking, token tracking, multi-target tracking, motion correspondence, multi-frame optimization, multi-dimensional assignment problem.

3.1 Introduction

Computer vision deals with the interpretation of image sequences. Because the problem of semantic labeling of an arbitrary scene is far from solved, any information that can help the scene interpretation should be exploited. The known temporal dependencies between frames in a sequence together with known physical properties like inertia and rigidity have been proven to be very helpful. More than that, temporal relations can be crucial in circumstances in which the objects in the recorded scene are difficult to distinguish, either because of poor recordings, poor recording conditions, restricted recording devices/media, or because the objects appear identical anyway. The research fields concerned with these issues are among others object tracking [21], feature or token tracking [3] [10] [23] [34], and optical flow or motion estimation [9] [15]. Applications range from surveillance [16] [26] [32], motion analysis, and structure from motion [27] [28] [31] [33] to (multi-)target tracking [8] [17] [20].

Here, we restrict ourselves to the case that the objects have indeed an identical appearance, which leaves us with the positional information as sole feature for identification. For this reason the objects are simply referred to as points in the remainder of this paper. The consequent problem that has to be solved is called the *motion correspondence* problem, that is, finding corresponding measurements through an image sequence solely based on the measured positions and derived motion characteristics (Fig. 3.1(a)). For this problem appearance-based methods like optical flow estimation do not apply. Additionally, like among others [3] [6] [20] [23], we adopt a uniqueness constraint which states that a measurement originates from (at most) one point and a point results in (at most) one measurement.

There is a number of conditions under which establishing motion correspondence is especially difficult: 1) the points move densely together, 2) the detection is imperfect, i.e. there

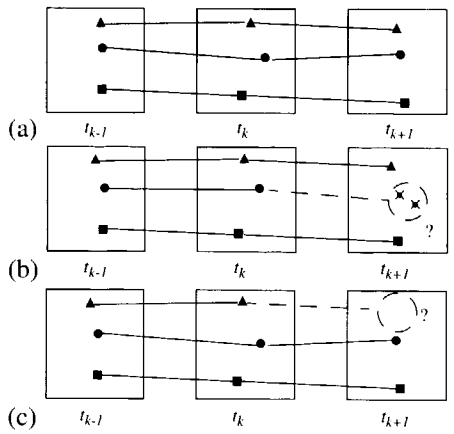


Figure 3.1: Three moving points are measured at three time instances. The lines represent the point correspondences over time. In (a) all points are measured at every time instance. In (b) there is an extra or spurious measurement at t_{k+1}, and in (c) there is a missing measurement at t_{k+1}.

are spurious (Fig. 3.1(b)) and missing (Fig. 3.1(c)) measurements, 3) points are temporarily occluded, and 4) the number of points varies. Here, we consider such difficult cases, except we assume that the number of points is fixed. Namely, without additional constraints, coping with both condition 3) and 4) gives rise to conflicting requirements for a tracking algorithm, as noted in [29]. Recently, in [29] we proposed a qualitative motion model together with the Goa tracker, which is a greedy matching method that efficiently finds optimal correspondences between two frames given a smoothness of motion criterion. We showed that the Goa tracker outperforms other greedy trackers [3] [19] [22] and even the presumed optimal multiple hypothesis tracker (MHT) [20] for the tracking of a fixed number of points[1]. In [29], we have also suggested a global matching model that optimizes over the whole sequence, but did not report an algorithm that satisfies the model. However, the more difficult the problem, the more important it becomes to perform a global matching, i.e. to defer correspondence decisions. Especially when there are many spurious measurements, deciding greedily is dangerous, although the introduction of motion constraints can limit the damage.

In this paper, we propose an algorithm that enables us to defer correspondence decisions by introducing a temporal scope parameter s. With scope $s = 1$ this algorithm equals the previously introduced Goa tracker [29] and when $s = n$ the algorithm performs a global matching, where n is the number of frames in the sequence. The extended temporal scope tracking resembles the beam-search principle in [34], trajectory aging in [10] and the N-scan-back principle in the statistical data association filters [4] [5] [6] [7] [17] [18] [20] [24] [25][2]. In contrast with our problem setting, [10] [25] [34] do not adopt the uniqueness constraint,

[1] In this paper, we only compare our results with the MHT [20] since it had the most competitive performance.
[2] A temporal scope $s = 1$ is similar to $N = 0$ in N-scan-back filters.

hence, they optimize the tracks independently. Clearly, that problem is less complex, though it may lead to unrealistic assignments. In [6] [7] [17] [18] [24] the statistical data association problem has been formulated as a multi-dimensional assignment problem and approximate solutions have been presented using Lagrangian relaxations techniques. With respect to the multi-frame optimization, our approach is more similar to the multiple hypothesis tracker (MHT) in [4] [5] [20] in the way that we also rank the best assignments per frame and finally decide for the 'optimal' assignment after a certain number of frames has been processed. However, our *optimization strategy* is quite different. We search the alternative solutions up to s levels in a depth-first way, whereas the track maintaining algorithms (like the MHT) can be said to search in a breadth-first way. The advantage of our method is that it needs less memory and allows for more effective pruning of unlikely alternatives. In the experiments section we give an indication of a suitable value for the temporal scope s.

In the next section we formulate the problem and give the notation we use. Then, we summarize and modify the motion models that we proposed in [29]. In Section 3.4 we introduce the new tracking algorithm that embodies the extended scope optimization scheme. In the experiments section, we show the appropriateness of the new algorithm.

3.2 Problem Statement

Given is a sequence of n time instances for which at each time instance t_k there is a set of m_k measurements $\mathbf{x}_j^k$ of points p_i moving in a 3-D world, with $1 \leq j \leq m_k$, $1 \leq k \leq n$, and $1 \leq i \leq M$. The measurements are vectors in a two-dimensional space, with dimensions S_w (width) and S_h (height), representing 2-D coordinates. The number of measured points m_k can be either smaller (occlusion or missing detections) or larger (spurious measurements) than M.

The problem is to find a set of M tracks that represents the (projected) motion of the M points through the 2-D space from t_1 to t_n. A track T_i is an ordered n-tuple of corresponding measurements: $\langle \mathbf{x}_{j_1}^1, \mathbf{x}_{j_2}^2, ..., \mathbf{x}_{j_n}^n \rangle$, with $1 \leq j_k \leq m_k$. It is assumed that points do not enter or leave the scene (ignoring condition 4) of the problem). A point track that has been formed up to t_k is called a track head and is denoted as T_i^k.

We use two additional ways to denote which measurement corresponds to which track head. First, we introduce the assignment matrix $A^k = [a_{ij}^k]$, where $a_{ij}^k = 1$ if and only if $\mathbf{x}_j^{k+1}$ corresponds to T_i^k and zero otherwise. Alternatively, we use $\alpha_j^k = i$ if $a_{ij}^k = 1$. Further, a concatenation of s assignment matrices from t_k to t_{k+s-1} is called a multi-assignment, denoted as an s-tuple: $A^{k:s} = \langle A^k, A^{k+1}, ..., A^{k+s-1} \rangle$, where $A^{k:s}[1] = A^k$, $A^{k:s}[2] = A^{k+1}$, etc.

3.3 Modeling

Here, we only give a brief description of our way to model the motion correspondence problem. For a more detailed description and analysis we refer to [29].

In order to select the corresponding measurement for a track head from the list of candidate measurements we need to have a model of the point motion: the *individual* motion model. In addition to prior motion models the parameters of such a model can be constructed on-line from the tracked measurements. Since it is impossible to rule out model errors, or, in other words, to predict the point positions perfectly, usually there will be correspondence ambiguities. Therefore, additional *combined* and *global* motion models have been proposed to make prediction errors dependent. Here, we summarize the individual, combined, and global models.

3.3.1 Individual Motion Model

The individual motion model expresses predictions about the position of a moving point based on historical track information. Further, it states the cost when deviating from these predictions. Here, we formulate two different individual motion models.

*im*1 The nearest-neighbor model does not incorporate velocity information. It only states that a point moves as little as possible from t_k to t_{k+1}. Consequently, the model uses only measurements of one previous time instance for the position prediction.

$$c_{ij}^k = \| \mathbf{x}_j^{k+1} - \mathbf{x}_i^k \|, \quad \text{where} \quad 0 \leq c_{ij}^k \leq \sqrt{S_w^2 + S_h^2}. \tag{3.1}$$

*im*2 The smooth-motion model as first introduced in [23] assumes that the velocity magnitude and direction both change gradually. This model uses measurements from two previous time instances. The smooth motion is formulated quantitatively with the following criterion:

$$\begin{aligned} c_{ij}^k \quad = \quad & 0.1 \left[1 - \frac{\left(\mathbf{x}_i^k - \mathbf{x}_{\alpha_i^k}^{k-1}\right) \cdot \left(\mathbf{x}_j^{k+1} - \mathbf{x}_i^k\right)}{\left\| \mathbf{x}_i^k - \mathbf{x}_{\alpha_i^k}^{k-1} \right\| \left\| \mathbf{x}_j^{k+1} - \mathbf{x}_i^k \right\|} \right] + \\ & 0.9 \left[1 - 2 \frac{\sqrt{\left\| \mathbf{x}_i^k - \mathbf{x}_{\alpha_i^k}^{k-1} \right\| \left\| \mathbf{x}_j^{k+1} - \mathbf{x}_i^k \right\|}}{\left\| \mathbf{x}_i^k - \mathbf{x}_{\alpha_i^k}^{k-1} \right\| + \left\| \mathbf{x}_j^{k+1} - \mathbf{x}_i^k \right\|} \right], \end{aligned} \tag{3.2}$$

where $0 \leq c_{ij}^k \leq 1$.

To enable the modeling of spurious and missing measurements, we first need to modify the assignment matrix format. To this end, we extend A^k such that it has $M + m_{k+1}$ rows and $M + m_{k+1}$ columns. The first M rows represent the track heads of the target points and the first m_k columns represent the true measurements. The remaining rows and columns represent *false tracks* (to assign spurious measurements to) and *slave measurements* (to replace missing measurements), respectively. Additionally, the matrix $D^k = [c_{ij}^k]$ contains the individual

motion criterion coefficients, where c_{ij}^k expresses the deviation from the predicted position for measurement $\mathbf{x}_j^{k+1}$ to track head T_i^k. For true track heads to true measurements these coefficients are computed as defined above. All other entries in D^k equal ϕ_{max}, which is a known maximum of the individual motion criterion. For candidate correspondences that exceed a certain maximum speed (d_{max}) we set $c_{ij}^k = \phi_{max} + \epsilon$ to effectively disregard them[3].

Slave Interpolation

If any of the measurements in the vectors $(\mathbf{x}_i^k - \mathbf{x}_{\alpha_i^k}^{k-1})$ and $(\mathbf{x}_j^{k+1} - \mathbf{x}_i^k)$ are missing, the vectors are estimated by interpolation according to:

$$\mathbf{x}_i^k - \mathbf{x}_{\alpha_i^k}^{k-1} = \frac{\mathbf{x}_{\alpha_i^{k\to q}}^q - \mathbf{x}_{\alpha_i^{k\to p}}^p}{q-p}; \qquad \mathbf{x}_j^{k+1} - \mathbf{x}_i^k = \frac{\mathbf{x}_i^{k+1} - \mathbf{x}_{\alpha_i^{k\to q}}^q}{k+1-q}, \tag{3.3}$$

where $\mathbf{x}_{\alpha_i^{k\to p}}^p$ and $\mathbf{x}_{\alpha_i^{k\to q}}^q$ are true measurements in the nearest past in T_i^k, $1 \le p < q \le k$ and $\alpha_i^{k\to q}$ means $k-q$ times recursive application of α_i^k.

3.3.2 Combined Motion Model

The combined motion model serves to make individual model errors dependent between two successive frames. Here, we give only one such combined motion cost definition $C^k(A^k)$ (see [29] for alternative ones) that aims at spreading the errors as much as possible.

$$C^k(A^k) = \frac{1}{M} \sum_{i=1}^{M+m_{k+1}} \sum_{j=1}^{M+m_{k+1}} a_{ij}^k c_{ij}^k, \tag{3.4}$$

3.3.3 Global Motion Model

The global motion model serves to model the overall motion from t_1 to t_n. It averages out the combined motion errors over time, and in this way it ensures that the combined motion errors depend on each other.

$$S(D) = \min_{A^{[2:n-1]}} \sum_{k=2}^{n-1} C^k(A^k) \tag{3.5}$$

This global model is, however, hard to optimize. Therefore, we redefine the global model with the temporal scope s as parameter.

[3]Where ϵ is a arbitrary (small) positive number

$$S_s(D) = \sum_{k=2}^{n-1} C^k(A^{k:s}_{min}[1]), \tag{3.6}$$

where

$$A^{k:s}_{min} = \arg\min_{A^{k:s}} C^{k:s}(A^{k:s}) \quad \text{with} \quad C^{k:s}(A^{k:s}) = \sum_{p=1}^{s} C^{k+p-1}(A^{k:p}[p]) \tag{3.7}$$

When $s = 1$ Eq.3.6 equals $\hat{S}(D)$ in [29] and when $s = n$ Eq.3.6 equals Eq.3.5.

3.4 Restrained Optimal Assignment Decision (ROAD) Tracker

As we already mentioned, the computation of the global motion model (Eq.3.5) is intractable in general. The complexity can be reduced by using a limited temporal scope s as in Eq.3.6. However, the problem to be solved is an $(s+1)$-dimensional assignment problem with non-decomposable cost [1], which is known to be NP-hard for $s + 1 \geq 3$ [11]. Finding a greedy matching solution ($s = 1$) can be formulated as a classical (2-dimensional) assignment problem [29], for which a number of efficient solutions has been reported in the literature, among which the Hungarian method [12] is the best known. When the scope is larger, $s > 1$, the problem is still NP-hard. Therefore, it is important to limit s and to use a specific strategy to search the alternatives efficiently. Here, we propose the ROAD tracker, a recursive algorithm that searches the alternatives depth first up to s levels. Since it has been shown that the greedy solution is close to optimal [29], a best-first heuristic per recursion level is an obvious strategy.

Because the global motion criterion is additive and monotonic increasing, we also propose to use the *branch-and-bound* mechanism, where the initial bound is determined as the algorithm searches best first per recursion level in a depth-first way. Moreover, we adaptively lower the bound by introducing a combined motion constraint γ_{max}. For the computation of γ_{max} we introduce two assumptions. First, we assume that the cost of the solution $A^{k:s}_{min}$ is more or less uniformly spread over the recursion levels (scope). So after finding a temporary solution $A^{k:s}_{sol}$ with global cost (bound) C_b we stop testing alternative assignments if their cost exceeds $F^{\gamma}_g C_b/s$, where s is the (remaining) scope and F^{γ}_g ($F^{\gamma}_g \geq 1$) is a factor that expresses the maximum allowed deviation from C_b. Second, we assume that the optimal assignment $A^{k:s}_{min}[1]$ cannot have a much higher cost than the cost C^k_{min} of the locally best assignment A^k_{min}. So we additionally stop the testing of alternatives if their cost exceeds $F^{\gamma}_l C^k_{min}$, where F^{γ}_l ($F^{\gamma}_l \geq 1$) expresses the maximum allowed deviation from C^k_{min}. This leads to the following combined motion constraint:

$$\gamma_{max} = \min(F_l^{\gamma} C_{min}^k, F_g^{\gamma} C_b/s) \tag{3.8}$$

In contrast with the constraints on the individual motion (d_{max} and ϕ_{max}), γ_{max} is not physically motivated. Consequently, when γ_{max} is used, the solution can no longer be guaranteed to be optimal with respect to the individual motion models.

Now we only need a way to generate the assignments between t_k and t_{k+1} in best-first C^k-order. To this end we use Murty's algorithm [14], which is an efficient algorithm to rank assignments in order of increasing cost. This algorithm was used before in [4] to enumerate hypotheses for the statistical multiple hypothesis tracker. In short, the Murty algorithm returns the minimum cost assignment for an assignment problem given a number of assignments Y is no longer allowed, where $Y \subseteq U^k$:

$$A_{min}^k(Y) = \arg \min_{A \in U^k - Y} C^k(A), \tag{3.9}$$

where U^k is the set of all possible assignment matrices at t_k.

3.4.1 Basic ROAD Tracker

After having introduced the main elements, we now describe the complete ROAD tracker algorithm. One of the parameters of the ROAD tracker is A^{k-1}, which serves to initialize the individual motion models, hence to compute D^k (for *im*2). So far, we did not include this parameter in any of the criterion definitions (Eq.3.4-3.9). In the recursive calling of the ROAD algorithm, however, we include the A^{k-1} parameter, because the controlled permutation of A^{k-1} is the main ingredient of this recursive algorithm. Clearly, in the first frame the assignment for the previous frame is not available, leading to an initialization problem. We return to this afterwards.

The algorithm works as follows (see Fig 3.2). First, it computes the criterion matrix D^k using A^{k-1}. It constructs a bi-partite graph from D^k and prunes all edges with weights exceeding ϕ_{max}. Then, if the scope $s = 1$, it just returns the minimal cost assignment, which is the same as the Goa tracker result. Otherwise it starts enumerating the assignments according to increasing cost. For every generated assignment the algorithm calls itself recursively to figure out if there is a multi-assignment $A_{min}^{k+1:s-1}$ for this assignment matrix that results in a lower total cost than the given bound C_b. If the tracker finds such an improved multi-assignment, this assignment becomes the new solution.

3.4.2 Self-Initializing ROAD Tracker

We solve the just mentioned initialization problem in the same way as in [29], that is, by initializing A^1 with the minimal cost assignment A_{min}^k using the individual model *im*1 and running the algorithm once *up* and once *down*. This results in the algorithm shown in Fig. 3.3.

```
ROAD(A^{k-1}, k, s, C_b, A^{k:s}_{sol})
A^{k-1}   : assignment between previous and current frame
k         : frame number
s         : remaining scope
C_b       : cost bound for assignments in the remaining scope
A^{k:s}_{sol} : best solution for the remaining scope
begin
    C^k_{min} = C^k(A^k_{min})                          ; find minimum cost assignment
    if s = 1 then                                       ; at lowest recursion level?
        if C^k_{min} < C_b then                         ; better than global bound?
            A^{k:s}_{sol} = ⟨A^k_{min}⟩                 ; update solution
        end
    else
        Y = ∅                                           ; set of processed matrices
        do
            A = A^k_{min}(Y)                            ; get next best with Murty
            Y = Y ∪ {A}                                 ; add to processed set
            C_0 = C^k(A)                                ; compute cost
            T = A^{k:s}_{sol}[2...s]                    ; get default solution
            T = ROAD(A, k+1, s-1, C_b - C_0, T)         ; call recursively to improve T
            A^{k:s} = ⟨A⟩ ∘ T                           ; concatenate A with new tail
            if C^{k:s}(A^{k:s}) < C_b then              ; better than global bound?
                C_b = C^{k:s}(A^{k:s})                  ; update global bound
                A^{k:s}_{sol} = A^{k:s}                 ; update solution
            end
            γ_max = min(F^γ_l C^k_{min}, F^γ_g C_b/s)   ; compute combined constraint
        while (Y ≠ U^k ∧ C_0 < C_b ∧ C_0 < γ_max)       ; stop when global bound or
    end                                                 ; combined constraint exceeded
                                                        ; or no alternatives are left
    return A^{k:s}_{sol}                                ; return solution
end
```

Figure 3.2: The ROAD tracker for recursive multiple frame assignment optimization

	let $A^1 = A^1_{min}$	initialize first assignment with $im1$
	let $k = 2$	start at second frame
up:	$A^k = \text{ROAD}(A^{k-1}, k, s, \infty, \langle\rangle)$	find optimal assignment using scope s
	increase k	
	if $k < n$ go to *up*	
	otherwise go to *down*	
down:	decrease k	
	$A^k = \text{ROAD}(A^{k+1}, k, s, \infty, \langle\rangle)$	find optimal assignment using scope s
	if $k > 2$ go to *down*	
	otherwise done	

Figure 3.3: The self-initializing ROAD tracker.

3.5 Experiments

With the experiments, we intend to show the improved performance that can be achieved by restraining the assignment decisions, that is, by increasing the temporal scope ($s > 1$). We used the PSMG data generator [30] and ran tests with varying scope s and constraint factors F_l^γ and F_g^γ. For details on the used PSMG parameter settings, see [29]. We always set F_l^γ equal to F_g^γ (both denoted as F^γ) and we fed the algorithm with the true (known) maximum speed in order to disregard physically impossible correspondences. In all experiments, we compared the results with those of the original Goa tracker, which is the same as the ROAD tracker with scope $s = 1$. As a reference tracking algorithm we added the well-known statistical multiple hypothesis tracker (MHT) [20] as described and implemented by Cox and Hingorani [4]. The essential parameters of the MHT were trained with a genetic algorithm on labeled data sets of 50 points; there were no missing or spurious measurements. Both for the ROAD tracker and the MHT we set (additional) pruning parameters to limit the solution space in addition to the model constraints. The ROAD tracker evaluates no more than 300 candidates at each recursion level. The MHT has at maximum 300 global hypotheses[4] per group, a track tree depth of 3, while the minimum ratio between the likelihoods of the best and the worst hypothesis is 0.005. All displayed results are the average of 500 runs. When the average run time of an experiment exceeded 10 seconds, the experiment was stopped. As a consequence, some curves in the figures are incomplete.

3.5.1 Variable Density Experiment.

In the first experiment, we explored the performance of the ROAD tracker as a function of the point density. The performance is expressed as the ratio of incorrect tracks and total number of tracks, which we call the track error (E_{track}) after [30]. Fig.3.4(a) clearly shows that the ROAD tracker outperforms both the Goa tracker and the MHT. Further, the less constrained

[4]The number of candidates for the ROAD tracker and the number of global hypotheses per group for the MHT have different meanings.

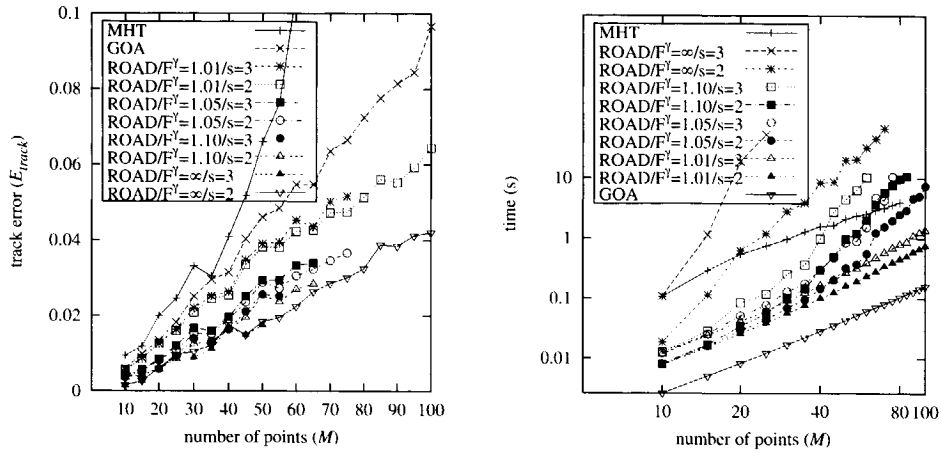


Figure 3.4: (a) Shows the track error as a function of the number of points and (b) shows the computation time as a function of the number of points.

the combined motion is, the better the performance. Remarkably, the track error with scope $s = 3$ is larger than with $s = 2$ when the same combined constraint setting is used. This is because with a larger scope $s = 3$ the global cost C_b can decrease faster, resulting in a stricter combined motion constraint at the highest search levels. However, the unconstrained experiments ($F_l^\gamma = F_g^\gamma = \infty$) show that with $s = 3$ the best results can be accomplished[5]. Nevertheless the computation time quickly becomes a bottleneck, as the next experiment will demonstrate.

3.5.2 Variable Volume Experiment.

In the next experiment, we varied the number of points while the point density remained the same. Consequently, the problem remains equally difficult. Fig.3.4(b) shows that the Goa tracker is the fastest and has polynomial complexity. The ROAD tracker has exponential complexity, but when $s = 2$ and F_l^γ and F_g^γ are low ($1 \leq F_l^\gamma, F_g^\gamma \leq 1.05$), the exponential order is also quite low, so that near-polynomial behavior is achieved over a range from 10 to 100 points. The MHT is slow but, because of the pruning parameters, it has near-polynomial complexity. It has, however, to be mentioned that the track error increases considerably with the number of points[6].

[5] In the unconstrained experiments we did only 200 runs and set the maximum average run time to 1000 sec. in order to show that the performance indeed improves.

[6] In contrast to the MHT, the Goa tracker has a constant track error, and with the ROAD tracker the track error even decreases with the number of points.

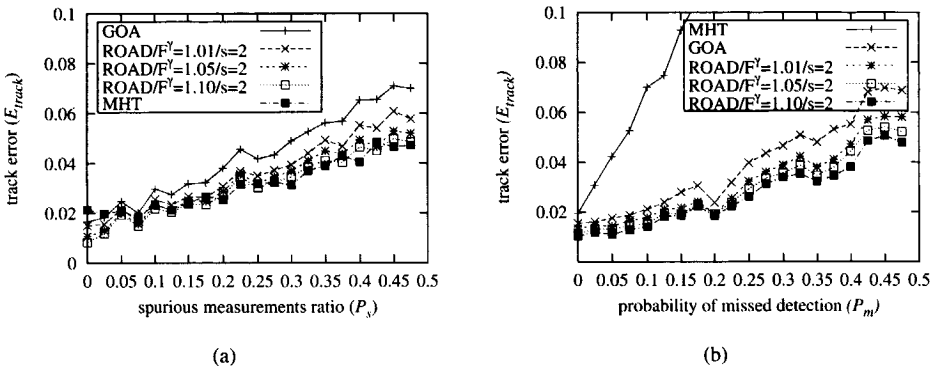


Figure 3.5: (a) Shows the track error as a function of the spurious measurements ratio P_s and (b) shows the track error as a function of the probability of missed detection P_m.

3.5.3 Variable Number of Spurious Measurements.

In order to show the importance of deferring correspondence decisions in the presence of noise, we did an experiment where we gradually incremented the number of spurious measurements. The number of spurious measurements is normally distributed around the displayed mean ratio P_s of the number of points ($M = 20$). As an example: $P_s = 0.5$ implies an average of 10 spurious measurements per frame. The position of these measurements is uniformly spatially distributed. Fig.3.5(a) indeed shows that the track error is lower when $s > 1$. Moreover, deferring the assignment decisions even has as a result that the difference between the ROAD tracker and the Goa tracker becomes larger as the amount of spurious measurements grows. With the specifically trained MHT the track error hardly increases. When the noise ratio $P_s > 0.1$, the MHT clearly performs best. It followed from additional experiments that the Goa tracker and the ROAD tracker performed worse in these cases because of the noise sensitivity of their initialization scheme. That is, especially the spurious measurements in the second frame result in deviant initial motion vectors. Then, during the *up* optimization, some true measurements may be considered as noise. Because the *down* optimization scheme only operates on tracks that have measurements in the last frames, it is not always possible to undo the effects of such an ill initialization. However, if there are no spurious measurements in the second frame, the Goa tracker and the ROAD tracker always perform better than the MHT for all settings of P_s.

3.5.4 Variable Number of Missing Measurements.

Finally, we did an experiment with simulated missing detections. In Fig.3.5(b) we display the track error as a function of the probability that a point was not detected or missed P_m. According to the problem definition, i.e. no scene entrance and exit, all points are detected in

the first and last two frames. Again the number of points is $M = 20$. The ROAD tracker with various settings performs better than the Goa tracker. Also in this experiment the difference between the ROAD tracker and the Goa tracker increases as the problem becomes more difficult. The MHT turns out to be extremely sensitive to occlusion. Part of the problem is that, although the probability of detection is set properly, the MHT easily divides tracks into separate parts.

3.6 Conclusion

In this paper, we described the ROAD tracker, a recursive algorithm that establishes motion correspondence by optimizing over several frames using a qualitative motion model. At each recursion level the tracker evaluates candidate assignments in best-first order using Murty's algorithm.

As an extension of the Goa tracker, the ROAD tracker uses the same *individual*, *combined* and *global* motion models. We introduced an approximation of the global motion model, which additionally has a temporal scope parameter. Further, we introduced an adaptive combined motion constraint γ_{max} on top of the branch-and-bound mechanism to reduce the exponential growth in computation time. The various experiments show that the deferment of assignment decisions indeed improves the tracking performance. Even with a very strict combined constraint setting ($1 \leq F_l^\gamma, F_g^\gamma \leq 1.05$), the ROAD tracker clearly outperforms the Goa tracker in all experiments. Relaxing this constraint further improves the performance, but care must be taken since unconstrained assignment optimization over several frames is intractable in general, as the experiments have shown. The experiments have also shown that setting the temporal scope to $s = 2$ gives the best compromise between qualitative and computational performance. The ROAD tracker also outperforms the specifically trained MHT, except when there are a lot of spurious measurements in the first frames. This issue needs further investigation.

Some additional remarks about the computational aspects. The Goa tracker is the fastest and the only algorithm with polynomial complexity. The MHT is the slowest, though the MHT is difficult to judge in terms of computational complexity. That is, it was not possible to configure the MHT such that it had a constant track error in the variable-volume experiment.

As the unconstrained experiments show, the track error performance can hardly be improved given the applied composite motion model. The efficiency of the proposed algorithm can, however, be improved by implementing optimizations to Murty's algorithm, as reported in [2] [13].

Finally, the next extension to the ROAD tracker must be to allow for the tracking of a variable number of feature points. When points or objects can enter and leave the scene, the algorithm can be applied in an even broader domain.

Bibliography

[1] H-J. Bandelt, Y. Crama, and F.C.R. Spieksma. Approximation algorithms for multidimensional assignment problems with decomposable costs. *Discrete Applied Mathematics*, 49:25–50, 1994.

[2] M. Bellmore and J.C. Malone. Pathology of traveling-salesman subtour-elimination algorithms. *Operations Research*, 19:278–307, 1971.

[3] D. Chetverikov and J. Verestóy. Feature point tracking for incomplete trajectories. *Computing, Devoted Issue on Digital Image Processing*, 62:321–338, 1999.

[4] I.J. Cox and S.L. Hingorani. An efficient implementation of Reid's multiple hypothesis tracking algorithm and its evaluation for the purpose of visual tracking. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 18(2):138–150, February 1996.

[5] R. Danchick and G.E. Newnam. A fast method for finding the exact N-best hypotheses for multitarget tracking. *IEEE Transactions on Aerospace and Electronic Systems*, 29(2):555–560, April 1993.

[6] S. Deb, K.R. Pattipati, and Y. Bar-Shalom. A new algorithm for the generalized multidimensional assignment problem. *IEEE International Conference on Systems, Man and Cybernetics; Emergent Innovations in Information Transfer Processing and Decision Making*, pages 249–254, 1992.

[7] S. Deb, M. Yeddanapudi, K. Pattipati, and Y. Bar-Shalom. A generalized S-D assignment algorithm for multisensor-multitarget state estimation. *IEEE Transactions on Aerospace and Electronic Systems*, 33(2):523–538, 1997.

[8] T.E. Fortmann, Y. Bar-Shalom, and M. Sheffe. Sonar tracking of multiple targets using joint probabilistic data association. *IEEE Journal of Oceanic Engineering*, 8(3):173–184, July 1983.

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

[10] V.S.S. Hwang. Tracking feature points in time-varying images using an opportunistic selection approach. *Pattern Recognition*, 22(3):247–256, 1989.

[11] R.M. Karp. Reducibility among combinatorial problems. In R.E. Miller and J.W. Thatcher, editors, *Complexity of Computer Computations*, pages 85–103. Plenum Press, New York, 1972.

[12] H.W. Kuhn. The hungarian method for solving the assignment problem. *Naval Research Logistics Quarterly*, 2:83–97, 1955.

[13] M.L. Miller, H.S. Stone, and I.J. Cox. Optimizing Murty's ranked assignment method. *IEEE Transactions on Aerospace and Electronic Systems*, 33(3):851–861, 1997.

[14] K.G. Murty. An algorithm for ranking all the assignments in order of increasing cost. *Operations Research*, 16:682–687, 1968.

[15] H.H. Nagel. Displacement vectors derived from second order intensity variations in image sequences. *Computer Vision Graphics and Image Processing*, 21(1):85–117, 1983.

[16] R. Pless, T. Brodský, and Y. Aloimonos. Detecting independent motion: The statistics of temporal continuity. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8):768–773, August 2000.

[17] A.B. Poore. Multidimensional assignments and multitarget tracking. In *Partitioning Data Sets; DIMACS Workshop*, pages 169–196, New Brunswick, USA, 1995. American Mathematical Society.

[18] A.B. Poore and X. Yan. Data association in multi-frame processing. In *Proceedings of the Second International Conference on Information Fusion*, volume II, pages 1037–1044, Sunnyvale, USA, 1999.

[19] K. Rangarajan and M. Sha. Establishing motion correspondence. *CVGIP: Image Understanding*, 24(6):56–73, July 1991.

[20] D.B. Reid. An algorithm for tracking multiple targets. *IEEE Transactions on Automatic Control*, 24(6):843–854, December 1979.

[21] J.W. Roach and J.K. Aggarwal. Determining the movement of objects from a sequence of images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2(6):554–562, 1980.

[22] V. Salari and I.K. Sethi. Feature point correspondence in the presence of occlusion. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(1):97–91, January 1990.

[23] I.K. Sethi and R. Jain. Finding trajectories of feature points in a monocular image sequence. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 9(1):56–73, January 1987.

[24] P.J. Shea and A.B. Poore. Computational experiences with hot starts for a moving window implementation of track maintanance. In *Proceedings of the SPIE: The International Society for Optical Engineering*, volume 3373, pages 428–439, 1998.

[25] P. Smith and G. Buechler. A branching algorithm for discriminating and tracking multiple objects. *IEEE Transactions on Automatic Control*, 20:101–104, February 1975.

[26] C. Stauffer and W.E.L. Grimson. Learning patterns of activity using real-time tracking. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8):747–757, August 2000.

[27] R.Y. Tsai and T.S. Huang. Estimating three dimensional motion parameters of a rigid planar patch, III: Finite point correspondences and the three view problem. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 32:213–220, 1984.

[28] S. Ullman. *The Interpretation of Visual Motion*. M.I.T. Press, Cambridge, MA, 1979.

[29] C.J. Veenman, M.J.T. Reinders, and E. Backer. Resolving motion correspondence for densely moving points. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(1):54–72, January 2001.

[30] J. Verestóy and D. Chetverikov. Experimental comparative evaluation of feature point tracking algorithms. In *Proceedings Workshop on Evaluation and Validation of Computer Vision Algorithms*, pages 183–194. Kluwer series in Computational Imaging and Vision, 2000.

[31] J.A. Webb and J.K. Aggarwal. Structure from motion from rigid and jointed objects. *Artificial Intelligence*, 19:107–130, 1982.

[32] L. Wixson. Detecting salient motion by accumulating directionally-consistent flow. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8):774–780, August 2000.

[33] Z. Zhang. Estimating motion and structure from correspondences of line segments between two perspective images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17(12):1129–1139, 1995.

[34] Z. Zhang and O. Faugeras. Three-dimensional motion computation and object segmentation in a long sequence of stereo frames. *International Journal of Computer Vision*, 7(3):211–241, 1992.

Chapter 4

Motion Tracking as a Constrained Optimization Problem

This chapter has been submitted to *International Journal of Computer Vision*, April 2001.

Abstract In this paper we pose the problem of tracking of a varying number of points through an image sequence as a multi-objective optimization problem with additional hard constraints. One of the objectives is to find smooth tracks for a number of points optimized over several frames. The smooth tracks are established using a composite motion model, based on second-order motion characteristics. Within the model, the inevitable prediction errors are averaged out and measurement uniqueness is imposed. The optimization of the smoothness objective alone results in a multi-dimensional assignment problem that is known to be NP-hard. Clearly, in order to find a solution to the motion tracking problem modeled as such, the objectives must be ordered and a number of approximations must be made. The optimization algorithm we present is a sequential heuristic search algorithm. It adequately prunes the search tree in such a way that its exponential order remains low. When the algorithm is compared to other tracking algorithms, it turns out that the proposed algorithm is easy to tune and generally more efficient and more accurate.

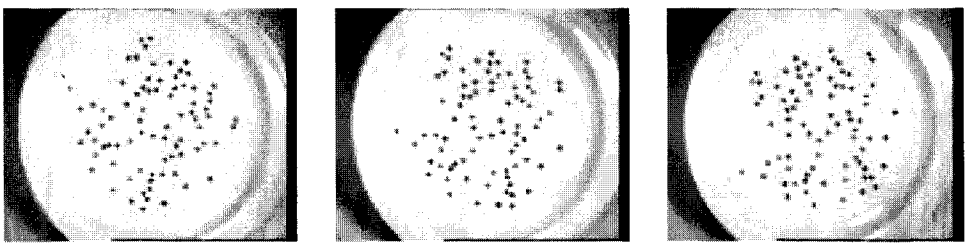
Keywords: Feature-point tracking, motion tracking, multi-target tracking, motion correspondence, multi-frame optimization, multi-objective optimization.

4.1 Introduction

Motion tracking is a very important task for the identification of objects and activities in video sequences [27]. One of the ways to approach the motion tracking problem is based on optical flow fields that are computed directly on the image sequence data [1], [11], [20]. Alternatively, feature or token tracking methods can be applied to segments or features that are retrieved by segmentation or filter operations [3], [13], [26], [44]. Applications of motion tracking range from surveillance [21], [30], [41] motion analysis, and structure from motion [12], [27], [33], [34], [39], [43] to (multi-)target tracking [9], [22], [24].

In certain cases, the color or brightness information is left out of consideration because it is unreliable or insignificant. Then, only positional information is used for the tracking, for which reason the problem is called the *motion correspondence problem*. The main cause of unreliable color information is changing light conditions. Additionally, the color information is insignificant in case of poor recording conditions, restricted recording devices or media [7], [9], [24], and when the points appear identical anyway [35], see for example Fig. 4.1. In this paper we especially address the motion correspondence problem, though color or brightness information can be included in order to improve the quality of the tracking method that we propose. We elaborate on this in the concluding section. In the remainder we refer to the entities that we track as *points* and we to the detected segments or features as *measurements*.

In order to track a number of points without using appearance information, one needs another way to find the *corresponding* measurements in consecutive images. One method to achieve this is to match measurements using the nearest-neighbor criterion [5], [8], [32]. Though it is the simplest method, it is also the least accurate, especially when the point density or speed is high. Second-order motion characteristics appear to give enough information to reasonably differentiate motion tracks from each other, and hence to find the correspond-



(a) image 1 (b) image 4 (c) image 7

Figure 4.1: Three images from an image sequence of a rotating dish with 80 black seeds. Clearly, on this scale most seeds look identical, which makes it impossible to track the seeds based on appearance information.

ing measurements [4], [13], [23], [26], [37]. Besides the limited order of the motion model, there are other causes that make establishing of motion correspondence problematic. The first problem relates to whether the tracking is done in the 2-D or 3-D space. Usually, points move in a 3-D world. When the scene is recorded with one camera, only 2-D positional information is retrieved. The consequent loss of information by the projection disturbs the assumed motion characteristics. Second, an imperfect detection scheme may result in a shortage or surplus of measurements. Third, a point may be occluded by other points or non-tracked objects such that it *cannot* be detected. Together with the fact that point manoeuvres may be under-sampled, a motion model can hardly be made free of errors. A more effective approach than (over)fitting the motion model to certain applications, a more effective approach is to accept an imperfect motion model and to postpone the correspondence decisions [7], [13], [22], [24], [36], [44].

In this paper, we focus on solving the motion correspondence problem for a monocular image sequence. It should be mentioned that besides the used 2-D motion models there are no restrictions that prevent the presented solutions to apply in 3-D as well. This work is complementary to [37], where a *non-statistical* motion framework is proposed for the tracking of a fixed number of points together with an efficient optimization scheme. The framework aims at establishing smooth tracks using a second-order motion model, while allowing for missing and falsely detected measurements. In contrast to [13], [29], [44], the proposed model imposes a uniqueness constraint, which states that a measurement originates from (at most) one point and a point results in (at most) one measurement. Without this constraint, tracks can be optimized independently, making the problem less complex, though at the cost of possibly unrealistic correspondences. The optimization algorithm in [37] optimizes the framework greedily by only considering two consecutive frames at the same time. In [36], an improved optimization scheme is reported which establishes the correspondence decisions using an extended temporal scope. This has indeed improved the tracking performance at a limited computational cost. To summarize, the method turned out to be qualitatively better, more efficient, and less sensitive to its parameter settings than an efficient implementation [4]

of Reid's multiple hypothesis tracker [24], which is the best-known statistical method, as well as other non-statistical methods [3], [23], [25] for tracking a fixed number of points. Optimizations to [4] have been reported [18], though similar optimizations apply to the algorithm described in [36] as well.

The limitation of the framework in [37] is that it allows for the tracking of a *fixed* number of points. The contribution of this paper is that we generalize the problem by lifting this restriction, so that the number of tracked points may vary over time. Accordingly, similar fundamental problems arise as when clustering data sets into an unknown number of clusters. That is, whereas in clustering ultimate homogeneity is achieved with as many clusters as data samples, here the tracks are ultimately smooth when they contain only two measurements. Additional constraints and objectives are needed in order to rule out these trivial solutions. Therefore, in this paper the motion framework is modified accordingly. Other methods that support the tracking of a time-varying number of points are [3], [7], [13], [17], [24], [25], [44]. Among these methods [3], [24] and [25] have been surpassed computationally and qualitatively in case the number of tracked points is fixed, as was demonstrated in [37]. In the performance evaluation in this paper, we again include algorithms [3] and [24] to compare their tracking performance for a varying number of points. We left out of consideration the method described in [7], since it uses a similar statistical model as [24] and only has a different approximation scheme. Further, we do not consider [13], [17], and [44] because they do not impose the just mentioned uniqueness constraint, which limits their application, especially in case of high point densities or high point speeds.

The outline of the paper is as follows. In the next section, we elaborate on the new problem. For the implementation of the extended tracking features, additional objectives are defined, resulting in a multi-objective optimization problem. In order to rule out the undesirable trivial solutions, two continuity constraints are proposed. Then in Section 4.3 we quantify the posed objectives, motion constraints and continuity constraints in a composite motion framework. In Section 4.4, we describe the new optimization algorithm and summarize its previously published components. In Section 4.5, we validate the method with artificial test data and real image sequences and compare its performance to other known tracking algorithms with similar capabilities. In the artificial test data experiments, we test the performance as a function of the point density, the average track length, and the average point motion dependence. With real image sequences we study the influence of other types of point motion, point dependence, noise, and occlusion. Finally, in Section 4.6, we draw conclusions about the proposed method and consider some directions for improvement with respect to the efficiency and performance quality.

4.2 The Motion Correspondence Problem

Establishing motion correspondence comes down to finding corresponding measurements between time instances solely based on positional information (Fig 4.2). Besides the development of a suitable motion model for individual points, the main problem is the resolution of correspondence ambiguities. These ambiguities relate to the imperfectness of the motion

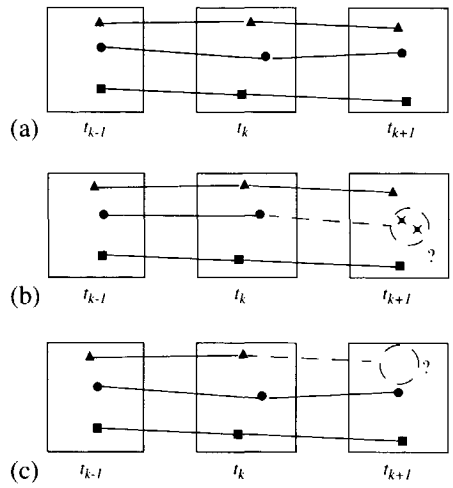


Figure 4.2: Three moving points are measured at three time instances. The lines represent the point correspondences over time. In (a) all points are measured at every time instance. In (b) there is an extra or spurious measurement at t_{k+1}, and in (c) there is a missing measurement at t_{k+1}.

model, occlusion, measurement noise (Fig. 4.2), segmentation problems (Fig. 4.2), as well as some more fundamental problems to which we return later. First, we list the requirements that a solution to the motion correspondence problem should meet.

R-1. *smoothness of motion*: measurements are assumed to belong to the same point track if the resulting track adheres to general physical constraints like inertia and rigidity.

R-2. *point uniqueness*: a measurement originates from at most one point. Consequently, a measurement may be part of at most one point track.

R-3. *measurement uniqueness*: a point results in at most one measurement. Derived from that, a point track contains at most one measurement at each time instance.

R-4. *track termination*: a point may disappear from the scene.

R-5. *track initiation*: new points can appear in the scene.

R-6. *missing measurements*: points may not be measured due to detection errors or occlusion.

R-7. *spurious measurements*: false measurements may be caused by detection errors.

Together the requirements R-2 and R-3 depict the previously mentioned uniqueness constraint. Since the only evidence of the possible presence of a point are its measurements, one can never be sure if a track really represents the motion of a single point or that the measurements represent appearing and disappearing points that just happen to form a smooth track. The latter is possible, though very unlikely. Therefore, the goal is to find a number of tracks that fit the just listed requirements, whether they represent true point tracks or not.

Problem Statement

Given is a sequence of n time instances for which at each time instance t_k there is a set X^k of m_k measurements $\mathbf{x}_j^k$, with $1 \leq j \leq m_k, 1 \leq k \leq n$. The measurements represent projected point locations of M_k points moving in a 3-D space. Accordingly, the measurements are vectors in a 2-D space, with dimensions S_w (width) and S_h (height) The number of measured points, m_k can be either smaller (because of occlusion or missing detections) or larger (because of false measurements) than M_k.

The problem to be solved is to find a set of M tracks T that represents the (projected) motion of all the points that have been in the scene between t_1 and t_n, where a track T_i is an ordered n-tuple of corresponding measurements: $\langle \mathbf{x}_{j_1}^1, \mathbf{x}_{j_2}^2, ..., \mathbf{x}_{j_n}^n \rangle$. A track may contain dummy entries ζ at its head when the point entered the scene after t_1. It may also contain dummy entries at its tail when the point left the scene before t_n. In case the point was temporarily occluded or missed for another reason the track may also contain dummies in between. The number of measurements in a track is denoted as $|T_i|$. We define the true track set representing the motion of all points in the scene as T^*.

We use two alternative ways to denote which measurements are linked together to form a track. First, we introduce the assignment matrix $A^k = [a_{ij}^k]$ $(1 \leq i \leq M_k, 1 \leq j \leq m_k)$, where $a_{ij}^k = 1$ if and only if $\mathbf{x}_j^{k+1}$ corresponds to $T_i[k]$ and zero otherwise, where $T_i[k]$ is the k-th element in T_i. Alternatively, we use $\alpha_j^k = i$ if $a_{ij}^k = 1$. Further, a concatenation of s assignment matrices from t_k to t_{k+s-1} is called a multi-assignment, denoted as an s-tuple: $A^{k:s} = \langle A^k, A^{k+1}, ..., A^{k+s-1} \rangle$, where $A^{k:s}[1] = A^k$, $A^{k:s}[2] = A^{k+1}$, etc. A candidate solution $A^{1:n-1}$, or A, is a multi-assignment of length $n-1$. A track set T can be derived from A by collecting all paths in A with more than one measurement.

4.3 The Motion Correspondence Model

Given a set of measurements per time instance, the problem is to find corresponding points in order to form tracks. Without further assumptions, numerous (undesirable) point tracks could be generated as solutions. If only some smoothness criterion is optimized, it is a trivial solution to qualify all measurements as being spurious, i.e. to form no point track at all, since partly visible tracks have a higher overall track smoothness than point tracks that are completely visible. Generally, if we do not constrain the number of tracks, there are virtually infinitely many partly visible point tracks that satisfy the given requirements. This results in the earlier noted [37] conflict between temporary occlusion and track termination

and initiation. In order to prevent these solutions, we define the objectives of the motion correspondence problem as follows:

O-1. Minimize the overall deviation from track smoothness. This objective demands that not only single tracks are smooth, but that all tracks as a group are as smooth as possible. Together with the uniqueness requirement R-2, this objective ensures that the optimal track smoothness of all tracks depend on each other.

O-2. Maximize the number of measurements that together form a track. We demand that as many as possible measurements are linked together to form point tracks.

O-3. Minimize the total number of tracks. Since it is necessary to constrain the number of tracks and that we do not know the number of points present in the scene a priori, we aim to minimize the number of tracks.

In order to rule out certain undesirable solutions, we add two continuity constraints that should be satisfied. These two continuity constraints show resemblance to the probabilities of detection, false alarms, track initiation, and track termination in the statistical tracking models [7], [9], [24]. The main difference is that the constraints that we present here are deterministic rather than statistical. Further, we need less a priori knowledge and model parameters in order to deal with the same phenomena.

C-1. *maximal absence* a_{max}: A point may be absent in at most a_{max} consecutive images. In [37] it has been noted that tracking a varying number of points conflicts with the handling of temporarily occlusion. The problem is that without semantic knowledge of the tracked points and their environment, it is almost impossible to decide whether a track has terminated or not. If a point has not been measured for a long time, there is still a chance that it will be measured again. However, the longer it has been invisible, the less is known about its state. In other words, such a point may fit perfectly to *any measurement*. To overcome this conflict we define a maximum absence parameter a_{max} that makes it possible to differentiate between *long occlusion* and *disappearance* of one point followed by the *appearance* of another. The true maximum absence of all points in the scene is denoted as a^*_{max}.

The following problems may result when the true constraint value is not known. If $a_{max} < a^*_{max}$ then some point tracks *will* incorrectly be divided up in separate parts. If $a_{max} > a^*_{max}$ then tracks of different points *may* be linked together. It depends on the structure of the data and the setting of the other constraints whether the latter kind of error will occur.

C-2. *minimal presence* p_{min}: A point must be measured in at least p_{min} consecutive images. In other words, a track may not consist of gathered spurious measurements. The underlying assumption is that points can be occluded or missed but are generally visible for a certain time in between. Accordingly, we want to rule out tracks that are formed by points that are detected once every $a_{max} + 1$ frames. The minimal presence parameter

p_{min} demands that a point *must* be measured in at least p_{min} consecutive frames. The true minimum presence of all points in the scene is denoted as p^*_{min}.

If $p_{min} > p^*_{min}$ then some point tracks *will* incorrectly be divided up in separate parts. On the other hand, setting $p_{min} < p^*_{min}$ *may* lead to the linking of track parts of different points. Again, it depends on the structure of the data and the values of the other constraints whether setting the constraint too loose will lead to errors.

Proposition 1 *If* $p^*_{min} = n$ *then*

I. $a^*_{max} = 0$

II. $M_k = M, 1 \leq k \leq n$

Proof:

I. If $p^*_{min} = n$ then there are no missing measurements in *any track*. Hence, $a^*_{max} = 0$. ■

II. If $p^*_{min} = n$ then $|T^*_i| = n, 1 \leq i \leq M$. Then, no track initiates after t_1 or terminates before t_n. Then it follows that the number of points in the scene is constant over time, i.e. $M_k = M$. ■

Additionally, physical properties of the recorded objects limit their speed and acceleration. Especially when the number of points varies over time, imposing the following motion constraints helps to differentiate between track extension and new track initiation.

M-1. *maximum speed* d_{max}: Points cannot move faster than d_{max} per time step.

M-2. *maximum deviation from smoothness* ϕ_{max}. Points cannot accelerate unrestrictedly.

So far we only stated that point tracks must be as smooth as possible. However, we did not quantify this statement. Next, we describe the composite motion framework (see also [37]), which gives quantitative expressions for overall smoothness in the point tracks and the motion constraints. Since we complemented the problem, we modified the framework such that it incorporates the additional tracking capabilities.

In order to select the corresponding measurement for a track from the list of candidate measurements, we need to have a model of the point motion, which we call the *individual* motion model. In addition to prior motion models, the parameters of such a model can be constructed on-line from the tracked measurements. Additionally, *combined* and *global* motion models are proposed to make prediction errors dependent between tracks and over time. Both the combined and the global model have been modified (w.r.t. [37]) in order to integrate the new objectives and constraints.

4.3.1 Individual Motion Model

The individual motion model expresses predictions about the position of a moving point based on historical track information. Further, it states the cost when deviating from these predictions. Here, we formulate two different individual motion models that are used throughout this paper.

im1 The nearest-neighbor model does not incorporate velocity information. It only states that a point moves as little as possible from t_k to t_{k+1}. Especially when the tracked points represent features on non-rigid objects the inertia principle may be violated. If other suitable models are lacking, the nearest-neighbor model may be considered [8], [35]. Later we will show that this model can also be used as a first estimate in order to initialize the next model.

$$c_{ij}^k = \left\| \mathbf{x}_j^{k+1} - \mathbf{x}_i^k \right\|, \quad \text{where } 0 \leq c_{ij}^k \leq \sqrt{S_w^2 + S_h^2}. \tag{4.1}$$

It can be easily seen that if a point does not move, $c_{ij}^k = 0$ for the corresponding measurements .

im2 The smooth-motion model as introduced in [26] assumes both that the velocity direction and magnitude change gradually. The smooth motion is formulated quantitatively with the following criterion:

$$\begin{aligned} c_{ij}^k \;=\; & w \left[1 - \frac{\left(\mathbf{x}_i^k - \mathbf{x}_{\alpha_i^k}^{k-1}\right) \cdot \left(\mathbf{x}_j^{k+1} - \mathbf{x}_i^k\right)}{\left\| \mathbf{x}_i^k - \mathbf{x}_{\alpha_i^k}^{k-1} \right\| \left\| \mathbf{x}_j^{k+1} - \mathbf{x}_i^k \right\|} \right] + \\ & (1-w) \left[1 - 2 \frac{\sqrt{\left\| \mathbf{x}_i^k - \mathbf{x}_{\alpha_i^k}^{k-1} \right\| \left\| \mathbf{x}_j^{k+1} - \mathbf{x}_i^k \right\|}}{\left\| \mathbf{x}_i^k - \mathbf{x}_{\alpha_i^k}^{k-1} \right\| + \left\| \mathbf{x}_j^{k+1} - \mathbf{x}_i^k \right\|} \right], \end{aligned} \tag{4.2}$$

where $0 \leq c_{ij}^k \leq 1$. The first term in this equation accounts for the angular deviation of the displacement vectors by computing their dot product. The second term accounts for the speed deviation of the displacement vectors as the ratio of their geometric and arithmetic means. We set the weight as proposed in [26] to $w = 0.1$.

If points move uniformly $\mathbf{x}_i^k - \mathbf{x}_{\alpha_i^k}^{k-1} = \mathbf{x}_j^{k+1} - \mathbf{x}_i^k$, then it follows that $c_{ij}^k = 0$ for the corresponding measurements.

Slave Interpolation

If any of the measurements in the vectors $(\mathbf{x}_i^k - \mathbf{x}_{\alpha_i^k}^{k-1})$ and $(\mathbf{x}_j^{k+1} - \mathbf{x}_i^k)$ are missing, the vectors are estimated by interpolation according to:

$$\mathbf{x}_i^k - \mathbf{x}_{\alpha_i^k}^{k-1} = \frac{\mathbf{x}_{\alpha_i^{k\to q}}^{q} - \mathbf{x}_{\alpha_i^{k\to p}}^{p}}{q-p}; \qquad \mathbf{x}_j^{k+1} - \mathbf{x}_i^k = \frac{\mathbf{x}_i^{k+1} - \mathbf{x}_{\alpha_i^{k\to q}}^{q}}{k+1-q}, \tag{4.3}$$

where $\mathbf{x}_{\alpha_i^{k\to p}}^{p}$ and $\mathbf{x}_{\alpha_i^{k\to q}}^{q}$ are the first true measurements in the nearest past in T_i^k, $1 \le p < q \le k$ and $\alpha_i^{k\to q}$ means $k-q$ times recursive application of α_i^k. The missing measurements are filled in with so-called *slave measurements*.

Hard Motion Constraints

The motion constraints M-1 and M-2 as proposed in Section 4.2 apply to (4.1) and (4.2). These constraints serve to disqualify certain correspondences. If either of the constraints is violated, we set $c_{ij}^k = \phi_{max} + \epsilon$ to effectively disregard the respective correspondence (where ϵ is an arbitrary positive number). We denote the true d_{max} as d_{max}^* and the true ϕ_{max} as ϕ_{max}^*, as derived from the point motion in the scene.

Detection Errors and Occlusion

In order to cope with spurious and missing measurements, we extend the assignment matrix A^k such that it has $M_k + m_{k+1}$ rows and $M_k + m_{k+1}$ columns. The first M_k rows represent the tracks of the target points and the first m_{k+1} columns represent the true measurements. The remaining rows and columns represent *false tracks* (to which spurious measurements are assigned) and *slave measurements* (which replace missing measurements), respectively. Additionally, the matrix $D^k = [c_{ij}^k]$ contains the individual motion criterion coefficients, where c_{ij}^k expresses the deviation from the predicted position for measurement $\mathbf{x}_j^{k+1}$ to track T_i. For true tracks to true measurements these coefficients are computed as defined above. All other entries in D^k equal d_{max} or ϕ_{max}, depending on the used individual motion model.

Proposition 2 *The assignment matrix A^k is a square matrix with maximum size $\sum_{i=1}^{k+1} m_i$.*

Proof: We prove this by induction. At t_1 the maximum number of points M_1 equals the number of measurements m_1. Hence, the size of A^1 is $M_1 + m_2 = m_1 + m_2$. Suppose that the size of A^p is $\sum_{k=1}^{p+1} m_k$. Suppose further that until t_p all measurements have been qualified as being spurious. Still, all of the measurements before t_p may be the first measurement of a track. So, $M_{p+1} = \sum_{k=1}^{p+1} m_k$. Then, the size of the assignment matrix A^{p+1} is $M_{p+1} + m_{p+2} = \sum_{k=1}^{p+1} m_k + m_{p+2} = \sum_{k=1}^{p+2} m_k$. If, on the other hand, any measurement before t_p was assigned to another measurement, say at t_q then, at t_{q+1} the number of tracks $M_{q+1} <$

$M_q + m_{q+1}$. Consequently, if any two measurements are assigned to each other the size of A^k shrinks. Hence, the size of A^k is less than or equal to $\sum_{i=1}^{k+1} m_i$. ∎

Clearly, when the point motion is modeled accordingly, the resulting assignment matrices become impractically large for long sequences.

4.3.2 Combined Motion Model

The combined motion model serves to make individual model errors dependent between two successive frames. Here, we give only one such combined motion cost definition $C^k(M_k, A^k)$ that aims at spreading the errors as much as possible, assuming all inter-point dependencies to be similar. Further, it implements the uniqueness requirements (R-2, R-3). In [37] the number of points was assumed fixed (M), but here we have a time-varying number of points M_k, which therefore is an additional parameter.

$$C^k(M_k, A^k) = \frac{1}{M_k} \sum_{i=1}^{M_k+m_{k+1}} \sum_{j=1}^{M_k+m_{k+1}} a_{ij}^k c_{ij}^k \tag{4.4}$$

subject to:

$$\text{(R-2)} \qquad \sum_{i=1}^{M_k+m_{k+1}} a_{ij}^k = 1, \quad 1 \le j \le M_k + m_{k+1}$$

$$\text{(R-3)} \qquad \sum_{j=1}^{M_k+m_{k+1}} a_{ij}^k = 1, \quad 1 \le i \le M_k + m_{k+1}$$

$$a_{ij}^k \in \{0, 1\}$$

The minimization of (4.4) is an assignment problem or a minimum-weight perfect matching problem (e.g. [40]), where the size of the problem is $M_k + m_{k+1}$. We define the assignment matrix A_{min}^k resulting from the minimization of (4.4) as:

$$A_{min}^k = \arg\min_{A \in U^k} C^k(M_k, A), \tag{4.5}$$

where U^k is the set of all possible assignment matrices at t_k.

There are several (optimal) algorithms that can be used to compute A_{min}^k, e.g. [6], [14], [16]. The Goa tracker for the tracking of a fixed number of points [37] uses the Hungarian method [16], which has $O(n^3)$ worst-case performance.

4.3.3 Global Motion Model

The global motion model serves to model the overall motion from t_1 to t_n. It averages out the combined motion errors over n time instances, and in this way it ensures that also the combined motion errors depend on each other. Here, we quantify the model objectives introduced in Section 4.2 in order to express the extended tracking features of the model.

$$\text{(O-1)} \qquad \min_{A \in U} \sum_{k=2}^{n-1} C^k(M_k, A^k) \tag{4.6}$$

$$\text{(O-2)} \qquad \max_{A \in U} \sum_{i=1}^{M} |T_i| \tag{4.7}$$

$$\text{(O-3)} \qquad \min_{A \in U} M \tag{4.8}$$

where U is the set of all multi-assignments of length $n-1$.

From the resulting multi-assignment A the set of tracks T can be extracted and the track constraints imposed. Only paths in the multi-assignment A with more than one corresponding measurement are considered tracks. Otherwise, when there is a single isolated measurement it is spurious according to the model.

The derived tracks should satisfy the following continuity and motion constraints:

$$\text{(C-1)} \qquad T_i[k] \neq \zeta \wedge T_i[k+1] = \zeta \Longrightarrow \bigvee_{p=1}^{a_{max}+1} T_i[k+p] \neq \zeta$$

$$\text{(C-2)} \qquad T_i[k] = \zeta \wedge T_i[k+1] \neq \zeta \Longrightarrow \bigwedge_{p=1}^{p_{min}} T_i[k+p] \neq \zeta$$

$$\text{(M-1)} \qquad \| T_i[k+1] - T_i[k] \| \leq d_{max}$$

$$\text{(M-2)} \qquad c^k_{pq} \leq \phi_{max}$$

where c^k_{pq} is the criterion for $\mathbf{x}^k_p = T_i[k]$ and $\mathbf{x}^{k+1}_q = T_i[k+1]$.

4.3.4 Model Properties

Here, we list some propositions that express the importance of selecting significant values for the model constraints.

Proposition 3 *If $p_{min} = n$ then $M \leq \min_{k=1}^{n} m_k$.*

Proof: Let $m_p = \arg\min_{k=1}^{n} m_k$. Suppose $M > m_p$. Then at t_p there must be a point track that misses a measurement. This contradicts with $p_{min} = n$. Hence, $M \leq \min_{k=1}^{n} m_k$. ■

Proposition 4 *If $d_{max} = \infty$ and $\phi_{max} = \infty$ then $M \geq \min_{k=1}^{n} m_k$.*

Proof: Let $m_p = \min_{k=1}^{n} m_k$. Suppose $M < m_p$. Then at t_p there is at least one measurement not linked to a track. With respect to the motion constraints, that measurement can be linked to any measurement at t_{p-1} and t_{p+1}. So either at the left or at the right side one

of the continuity constraints must have been violated. p_{min} cannot be violated at t_p because there is a measurement. If a_{max} is violated then $m_{p-1} < m_p$ or $m_{p+1} < m_p$, which is in contradiction to the assumption that $m_p = \min_{k=1}^{n} m_k$. ■

Collary 1 *If* $d_{max} = \infty$ *and* $\phi_{max} = \infty$ *and* $p_{min} = n$ *then* $M = \min_{k=1}^{n} m_k$.

Proof: Follows directly from Proposition 3 and 4. ■

It follows from these propositions that when there are many spurious measurements, d_{max} and ϕ_{max} should be adjusted accurately. Otherwise, there will certainly be tracks (partially) consisting of spurious measurements. Sometimes it is possible to adapt d_{max} and ϕ_{max} in space and in time, e.g. in [10] the motion constraints are adapted spatially.

Finally, setting any of the constraints too strict leads to the exclusion of the true track set T^* from the feasible region[1]. In general the true extremes for the constraints are not known, so care must be taken when setting these constraint values.

4.4 The Algorithm

There are two fundamental problems related to the model as described in the previous section. First, the model gives rise to a multi-objective optimization problem with additional constraints. As is known for multi-objective optimization problems, there is usually a large set of *mathematically equivalent optimal solutions*, i.e. the Pareto optimal set (e.g. [31]). This reflects the motivation for defining the objectives in the first place in Section 4.3. In order to strive for a single best solution, the optimization algorithm we propose gives priorities to the objectives. First, we postulate that the most important objective is to include as many measurements as possible in a point track (O-2). Then, we want the tracks to be as long as possible, or, in other words, we want to form as few tracks as possible (O-3). Finally, the tracks themselves must be as smooth as possible (O-1).

The second fundamental problem is that the *optimization of the model is intractable*. That is, even the single objective optimization problem of tracking a fixed number of points is known to be NP-hard. In order to make the optimization problem computationally feasible, the proposed algorithm has two related approximation steps. The first approximation of the algorithm is to *sequentially optimize* the global model. That is, instead of optimizing the model globally, we optimize it per time instance. Additionally, at each time instance we use a *restricted optimization scope* aiming at an estimation of $A_{min}^{k:s}$, which we define as follows:

$$A_{min}^{k:s} = \arg\min_{A^{k:s}} C^{k:s}(A^{k:s}), \quad \text{where} \quad C^{k:s}(A^{k:s}) = \sum_{p=1}^{s} C^{k+p-1}(A^{k:p}[p]) \tag{4.9}$$

A consequence of the sequential optimization procedure is that we are faced with an *initialization problem*. More precisely, the second-order individual motion criterion (*im2*)

[1] The feasible region of a constrained optimization problem is the set of solutions that satisfy the given constraints.

cannot be computed at the beginning of a point track. We solve this initialization problem with a forward (*up processing*) and backward (*down processing*) sequential optimization phase, similar to the solution to the initialization problem for the tracking of a fixed number of points in [37]. During the up processing, we use a two-stage procedure to estimate the motion vectors at the end of the point tracks. The first estimation stage *init1* hypothesizes the point correspondences for *im2* using *im1*. The second estimation stage *init2* continues the tracks using these hypotheses until the tracks stop. A track may stop because either the end of the sequence is reached, or there are no measurements in the d_{max}, ϕ_{max} range of the last measurement in the track, or the measurements fit better to other tracks. After these two stages, we assume that we have good estimates of the correspondences between the last measurements in the point tracks. Then, these are used in the reverse optimization phase, the down processing. The first goal of the down-processing phase is to make the tracks as smooth as possible. Especially when the second initialization stage has resulted in early track termination the first hypotheses were probably wrong. In that case the number of tracks will decrease while optimizing for smoothness in the down-processing phase. This will be shown in the step-by-step tracking example that we describe later in this section.

During the two-stage initialization procedure, conflicts may arise between the first initialization stage for one track and the second initialization stage for another track. That is, optimal measurements need to be selected for both the extension of a hypothesized track as well as for the initiation of new tracks. Again two criteria need to be optimized. We let the track extension take preference over track initiation as to express our preference for having as little tracks as possible (O-3).

Having defined the objectives and approximations of the proposed algorithm, we can now describe the two tracking phases of the so-called Roads tracker, being the up-processing and the down-processing phase (see Fig. 4.3).

Up processing The up direction aims at estimating the initial motion vectors of the point tracks in a two-stage process. First we execute the *init2* stage of the tracks T that have already been initiated, i.e. we attempt to extend these tracks while imposing the proposed constraints[2]. To this end, we need to compute $A_{min}^{k:s}$ using T and the measurements present in $X^{k+1}, X^{k+2}, ..., X^{k+s}$, where s is the scope of the optimization. We estimate $A_{min}^{k:s}$ with the Restrained Optimal Assignment Decision (Road) tracker [36], which we describe next. Before we determine the optimal assignment for the tracks, we check whether or not some of the tracks have terminated by imposing the continuity constraints. That is, a track may not contain more than a_{max} slave (missing) measurements and no less than p_{min} consecutive valid measurements. The state of the track is maintained with the enabled state E_i, where $E_i = 1$ if and only if the track T_i has not terminated. Then, this state variable is fed to the Road tracker in order to indicate which tracks are involved in the assignment optimization. The resulting assignment A^k is used to update the tracks in T.

After the track extension, we try to initiate new tracks using the remaining measurements in X^{k+1}. As just mentioned we optimize the assignment between t_k and t_{k+1} using *im1*, which

[2]For the first frame the track set T is clearly empty.

	$T = \emptyset$	initialize as empty track set
	let $k = 1$	start at first frame
up:	$E = ValidTracks(T, k)$	check tracks for termination
[*init2*]	$A^k = Road(A^{k-1}, k, s, \infty, \langle\rangle, E)$	find optimal assignment using *im2* and scope s
	$T = UpdateTracks(T, A^k)$	update tracks with found assignment
[*init1*]	$B^k = A^k_{min}$	with Road[s=1, *im1*] (= Goa tracker[*im1*])
	$T' = CreateTracks(B^k)$	create new tracks from this assignment
	$T = T \cup T'$	add tracks to track set
	$k = k + 1$	
	if $k < n$ go to *up*	
	otherwise go to *down*	
down:	$k = k - 1$	
	$E = ValidTracks(T, k)$	check tracks for initiation and termination
	$A^k = Road(A^{k-1}, k, s, \infty, \langle\rangle, E)$	find optimal assignment using *im2* and scope s
	$T = UpdateTracks(T, A^k)$	update tracks with found assignment
	if $k > 2$ go to *down*	
	otherwise done	

Figure 4.3: The Roads tracker for tracking a varying number of points. For the description of the arguments of the Road algorithm see Appendix A.

is the *init1* motion estimation stage. We convert the resulting assignment into a list of new tracks of the form $\langle \zeta, ..., \mathbf{x}_i^k, \mathbf{x}_j^{k+1}, ..., \zeta \rangle$, when $a_{ij}^k = 1 \in A^k$, and we insert these tracks in the track set T. This ends the second processing stage and we advance to the next frame. When all frames have been processed, we can start the down-processing phase.

Down processing In the down phase we use the initial track segments to find the final tracks. Since we assume that these segments can be erroneous, tracks may use measurements present in other tracks as well as measurements that are still unassigned. When all measurements from a track are taken by other tracks, the number of tracks decreases. Clearly, since some of the tracks present in T have ζ measurements at the tail, these tracks cannot be extended yet. In order to indicate whether a track can be extended, we define three track states, being: *before tracking*, *tracking*, and *after tracking*. All tracks start in the *before-tracking* state. Only those tracks that are in the *tracking* state will be extended. After a frame has been processed, the state of the track is updated resulting in a possible update of E_i, i.e. $E_i = 1$ if and only if track T_i is in *tracking* state. The state update works as follows:

1. From *before tracking* to *tracking*: if there are two consecutive true measurements in the track. Because all measurements are involved in the assignment optimization, even partially initialized tracks can lose their initial measurements. In such a case these tracks remain in the *before-tracking* state.

2. From *tracking* to *after tracking*: if either of the continuity constraints is violated, i.e. if

there are more than a_{max} consecutive missing measurements or there are less than p_{min} consecutive true measurements in a track.

If all frames have been processed in the down-processing phase, the algorithm stops. Now, we summarize the Road tracker from [36], which we use for the extension of partial tracks. It optimizes assignments over s frames in order to approximate $A^{k:s}_{min}$.

Road Tracker

As mentioned in [37] the computation of the global motion model for fixed M_k is NP-hard. Therefore, in [37] the global model has been approximated in a greedy way. That is, the model is approximated sequentially by optimizing only assignments between two frames. In [36] the optimization scope has been extended to several frames. Accordingly, the performance improved significantly, as has been demonstrated. Here, we summarize how the Road tracker estimates $A^{k:s}_{min}$.

When the scope of the optimization is larger than $s \geq 2$ the computation of $A^{k:s}_{min}$ results in an $(s+1)$-dimensional assignment problem which is known to be NP-hard for $s+1 \geq 3$ [15]. Accordingly, even if the scope is limited ($s < n$), we need a specific search strategy to constrain the exponential growth. The Road tracker searches the candidate assignments depth first up to s levels using a best-first heuristic per recursion level. Additionally, it prunes the search tree using an adaptive *branch-and-bound* mechanism. The initial bound is determined as the algorithm searches best first per recursion level. Then, the bound is lowered by introducing a *cost-bound constraint* γ_{max}. This cost-bound constraint is derived from the cost of the best local (current recursion level) assignment $C^k_{min} = C^k(A^k_{min})$ and the global cost bound C_b according to:

$$\gamma_{max} = \min(F^{\gamma}_l C^k_{min}, F^{\gamma}_g C_b/s), \tag{4.10}$$

where $F^{\gamma}_l (\geq 1)$ is the local cost factor and $F^{\gamma}_g (\geq 1)$ is the global cost factor and s is the remaining scope.

The best-first ordering of the assignments per recursion level is accomplished using the Murty algorithm [19]. In short, the Murty algorithm returns the minimum cost assignment for an assignment problem given that a number of assignments Y, where $Y \subseteq U^k$, is no longer allowed:

$$A^k_{min}(Y) = \arg \min_{A \in U^k - Y} C^k(M_k, A), \tag{4.11}$$

The Road tracker now works as follows. First, it computes the criterion matrix D^k using A^{k-1}. Then, if the scope $s = 1$, it just returns the minimal cost assignment A^k_{min} (like the Goa tracker in [37] using the Hungarian method [16]). Otherwise it starts enumerating the assignments according to increasing cost. For every generated assignment the algorithm calls itself recursively to figure out if there is a multi-assignment $A^{k+1:s-1}_{min}$ for this assignment

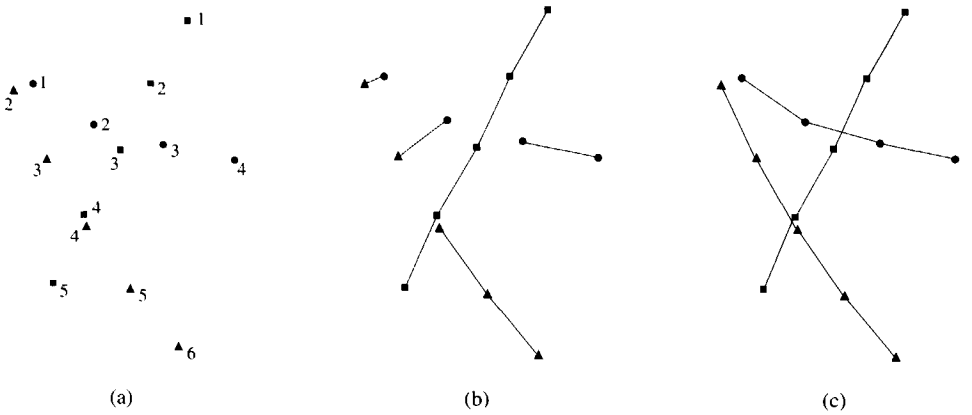


Figure 4.4: (a) shows a set of input points with attached time labels. In (b) the tracking result after *up processing* is displayed and in (c) after *down processing*.

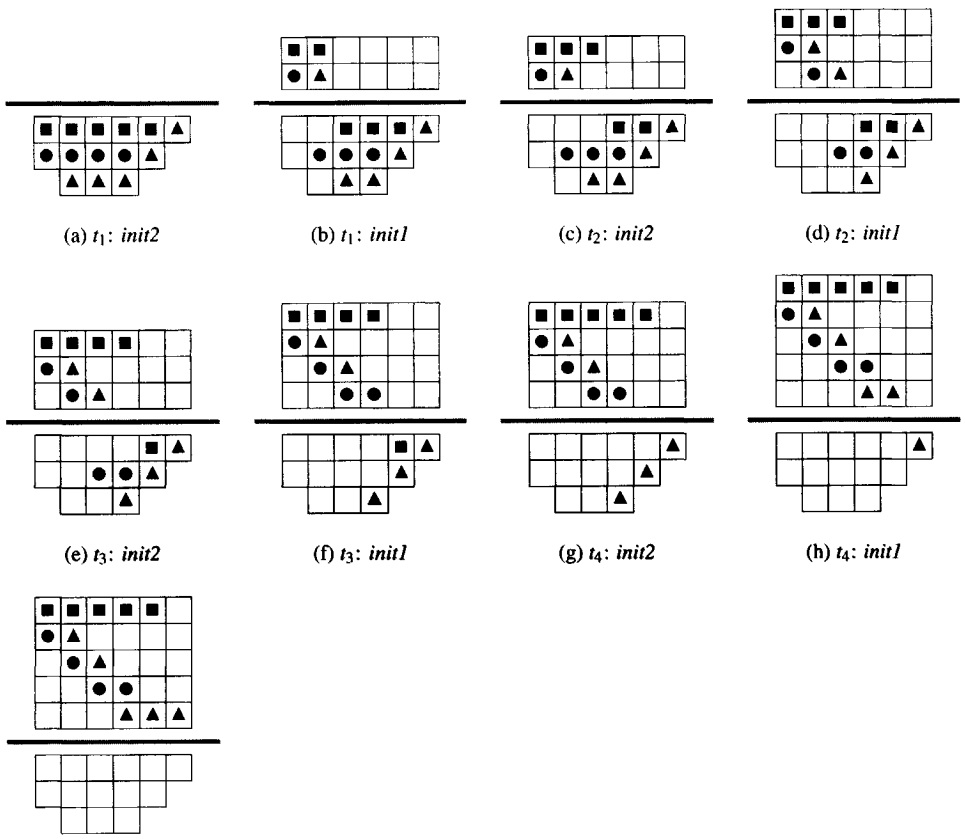
matrix that results in a lower total cost than the given bound C_b. If the tracker finds such an improved multi-assignment, this assignment becomes the new solution. The details of the algorithm are described in Appendix A.

Tracking Example

To show the workings of the Roads algorithm, we process an example step by step. The input data is displayed in Fig. 4.4 and consists of a sequence of six frames with 2, 3, 3, 3, 2, and 1 measurements per frame, respectively.

We start with the *up processing* of the sequence. For the first frame there are no tracks yet (see Fig 4.5(a)), so we skip the track extension. Therefore the only two measurements remain unbound. X^2 contains three measurements from which two are within the d_{max} range to both measurements in frame 1. Using the *im1* model, the correspondences are optimized resulting in two new tracks, as shown in Fig. 4.5(b). In frame two, we start with extending the existing tracks, i.e. the *init2* stage. One of the tracks has two candidate measurements and the other has none, because of the ϕ_{max} violation. Hence, only the first track is extended (see Fig. 4.5(c)). Then, in the *init1* stage, one new track is formed as shown in in Fig. 4.5(d). The frames 3-5 are processed likewise, resulting in the configurations as in Fig. 4.5(e)-4.5(h). Fig. 4.4(b) shows the resulting tracks after the up-processing phase.

The *down-processing* phase starts with the tracks found in the two-stage initialization procedure as shown in Fig. 4.6(a). From the third frame onwards (counting backwards now), correspondences are modified resulting in a decrease in the number of established point tracks as illustrated in Fig. 4.6(b)-Fig. 4.6(d). The final tracks are shown in Fig. 4.4(c).

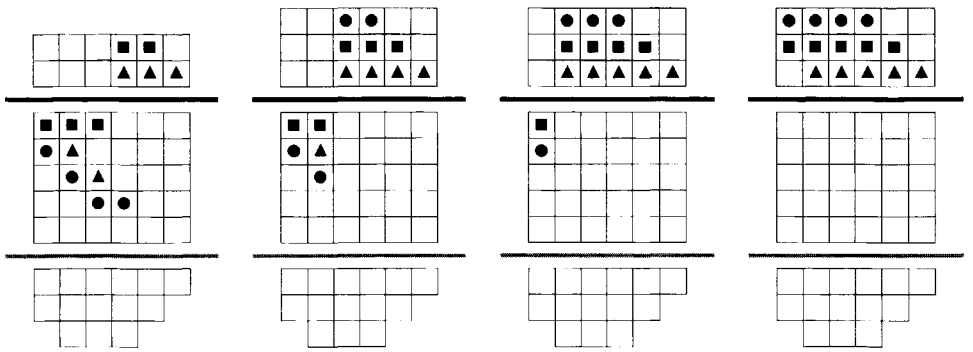


(a) t_1: *init2* (b) t_1: *init1* (c) t_2: *init2* (d) t_2: *init1*

(e) t_3: *init2* (f) t_3: *init1* (g) t_4: *init2* (h) t_4: *init1*

(i) t_5: *init2*

Figure 4.5: *Up processing* of a sequence of six frames with three points. The rows above the thick grey line represent the initialized tracks and the rows below the line represent the measurements, one column per frame.



(a) t_5 (b) t_4 (c) t_3 (d) t_2

Figure 4.6: *Down processing* of a sequence of six frames with three points. The rows above the thick black line represent the final tracks. The rows between the thick lines represent the hypothesized tracks after initialization and the rows below the grey line represent the unbound measurements. In this case there are no unbound measurements left after the up-processing phase.

Parameter Summary

We now summarize all parameters described for the presented tracking method. We differentiate between the model and the optimization parameters. The model parameters should be set as to fit the properties of the tracked points. Setting the model constraint parameters accurately is especially important in case the points move close to each other. The optimization parameters, on the other hand, do not reflect any motion properties of the points. They serve to improve the quality of the tracking results.

Model parameters		
w	*im2* (4.2)	second-order motion model parameter
d_{max}	(M-1)	maximum speed constraint parameter
ϕ_{max}	(M-2)	maximum deviation from smoothness constraint parameter
a_{max}	(C-1)	maximum absence constraint parameter
p_{min}	(C-2)	minimum presence constraint parameter

Optimization parameters		
s	(4.9)	temporal scope parameter
F_l^γ	(4.10)	local cost bound factor
F_g^γ	(4.10)	global cost bound factor

4.5 Experiments

In order to validate the extended capabilities of the Roads tracker, we carried out two types of experiments. First, we tested its performance with generated data. This type of experiment has the advantage that a number of problem parameters can be controlled so that the behavior of the algorithm can be investigated under various conditions. Moreover, the performance can easily be measured since the ground truth is available. Second, we applied the algorithm to some real image sequences to see how it operates when the presented data have possible unexpected real-world characteristics. In both types of experiments we compared the Roads tracker with Reid's Multiple Hypothesis Tracker (MHT) [4], [24] and the tracker by Chetverikov and Verestóy (C&V) [3]. We decided to include these trackers in the experiments because they have the same features as the Roads tracker but they have a different motion model or optimization scheme. In [37] it has already been shown that the original Roads tracker surpasses other tracking methods [23], [25], which we therefore do not consider here.

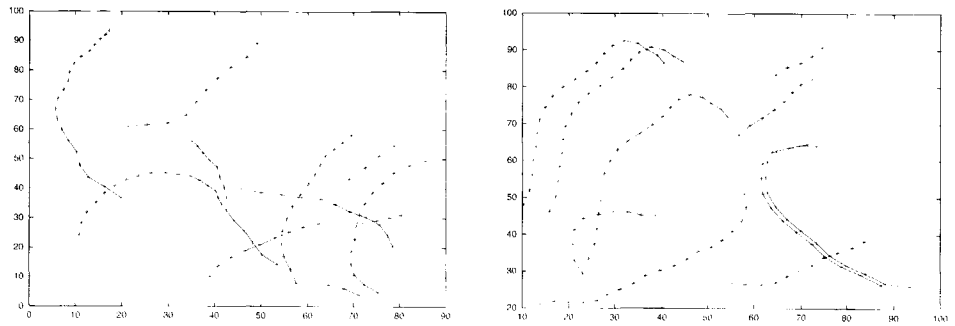
We always set F_l^γ equal to F_g^γ (F^γ) and we fed the Roads tracker and the C&V algorithm with the true d_{max} (given in Section 4.5.1 or by inspection in Section 4.5.2) in order to disregard physically impossible correspondences. In the generated data experiment we varied the scope and the cost-bound constraint, while in the real image sequence experiment we fixed these parameters to $s = 2$ and $F^\gamma = 1.05$. For the generated data experiment, the MHT needed extensive tuning of its relatively high number of parameters. Since the ground truth was available, we could use a genetic algorithm for this purpose. In the image sequence experiments we used the MHT parameter settings as described in [4], in which both image sequences were used too (and appropriate parameter setting can be assumed). For all algorithms we set the minimum track length to three, so that any track that has been formed using second-order motion characteristics is considered valid.

4.5.1 Performance with Generated Data

For the controlled generated data experiments, we used the Point Set Motion Generator (PSMG) [38] that allows for the generation of independent point tracks, with some variance in the speed and direction of motion. We modified the original PSMG since we wanted to test some additional problem parameter settings. In [36], we already added a parameter to vary the number of spurious measurements per time instance. For the new problem setting we needed a way to vary the number of points. Additionally, we added a way to test the sensitivity of the algorithms to point motion dependence.

1. Varying number of points.

   In order to test the behavior of the algorithms in case the number of points varies in time, we added a minimum track length parameter n_{min}. Then, each point track has two additional parameters, being the start frame s_i and its length l_i. The generation of random point tracks then works as follows. For each point track (a total of M) the length l_i is first chosen uniformly distributed between n_{min} and $n - n_{min}$, where



(a) (b)

Figure 4.7: In (a) a data set consisting 10 independent moving points with track length varying between 3 and is 20 frames is displayed. (b) shows a data set with 10 points with track length between 5 and 20 frames moving as 3 independent groups.

n is the total sequence length. Then, the start frame is chosen between 0 and $n - l_i$, again uniformly distributed. The point tracks themselves are generated according to the PSMG. Clearly, by having variable length point tracks that start at variable time instances, the actual number of points M_k varies from t_1 to t_n.

2. Point groups.

   In order to test the sensitivity of the algorithms for point motion dependence, we added a parameter G that represents the number of point groups. The generation of point tracks works as follows. First, G kernel tracks are generated as before. These kernel tracks always have a length of n frames. Then, the M point tracks are generated by selecting a random kernel track and a random initial position (in the given $S_w \times S_h$ window). Then, again a random track length and first frame are chosen as before, but now the remaining point positions directly follow the motion of the selected kernel. Consequently, when $G = 1$ all point motion is dependent and when $G = M$ (on the average) all point motion is independent.

When not explicitly indicated otherwise, we set $n_{min} = 5$, $n = 20$, $M = 50$, and $G = 50$. The other parameters have the same values as used in [37], being:

1. Size ($S_w = S_h$) of the square space ($S = 100$).
2. Uniform distributions for both dimensions of initial point positions between 0 and S.
3. Normal distribution for the magnitude of the initial point velocity vector: $v_i^0 = N(\mu_{v_0} = 5, \sigma_{v_0} = 0.5)$
4. Uniform distribution for the angle of the initial velocity vector, between 0 and 2π.

5. Normal distribution for the *update* of the velocity vector magnitude v_i^k, from t_k to t_{k+1}: $v_i^{k+1} = N(v_i^k, \sigma_{v_u} = 0.2)$

6. Normal distribution for the *update* of the velocity vector angle β_i^k from t_k to t_{k+1}: $\beta_i^{k+1} = N(\beta_i^k, \sigma_{\beta_u} = 0.2)$

7. Probability of occlusion ($p_o = 0$, i.e. no occlusion)

Performance Measure

A number of different measures have been proposed to quantify the quality of the performance, like among others the *distortion measure* [23], the *link-based error* and *track-based error* [38]. Like in [37], we used the track-based error as proposed in [38], which is defined as follows:

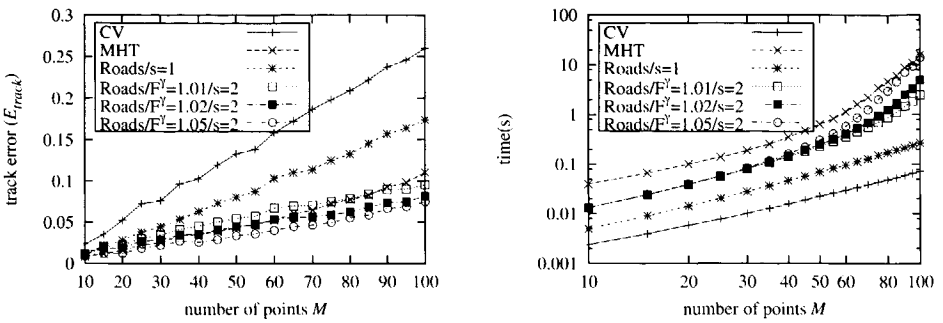
$$E_{track} = 1 - \frac{T_{correct}}{T_{total}}, \tag{4.12}$$

where T_{total} is the total number of true tracks and $T_{correct}$ is the number of *completely* correct tracks.

The choice for this measure is now less obvious than in [37], because in this case, the tracks can have different lengths. Clearly, it is usually more difficult to find completely correct tracks in case they are longer. In the figures the displayed track error and computation time is always averaged over 500 runs.

Variable Density Experiment

With the proposed data generator, we did a variable density experiment. We gradually increased the number of points while keeping the scene size constant. Figure 4.8(a) shows the track error E_{track} as a function of the total number of points M that has appeared in the scene. The figure clearly shows that the C&V algorithm performs worst. With scope $s = 2$ the Roads tracker has the best performance. Even with a strict cost-bound constraint setting ($F^\gamma = 1.01$) the Roads tracker performs better than the specifically trained MHT when the number of points $M \geq 80$. By relaxing F^γ the performance improves further. Relaxing F^γ beyond $F^\gamma = 1.05$ hardly resulted in any improvement. Figure 4.1b shows the average computation time for the variable density experiment. Clearly, the C&V algorithm is the fastest, followed by the Roads tracker with scope $s = 1$. Both have a near-polynomial order of performance. The MHT and the Roads tracker with scope $s = 2$ display the expected exponential growth in computation time, though the order is different. The MHT is clearly the slowest algorithm, while the exponential order of the Roads tracker depends on the cost-bound constraint setting.



(a) (b)

Figure 4.8: Results of a variable density experiment by applying the C&V algorithm, the MHT algorithm, and the Roads tracker to data sets from the modified PSMG data generator. In (a) the track error is displayed and in (b) the accompanying computation time.

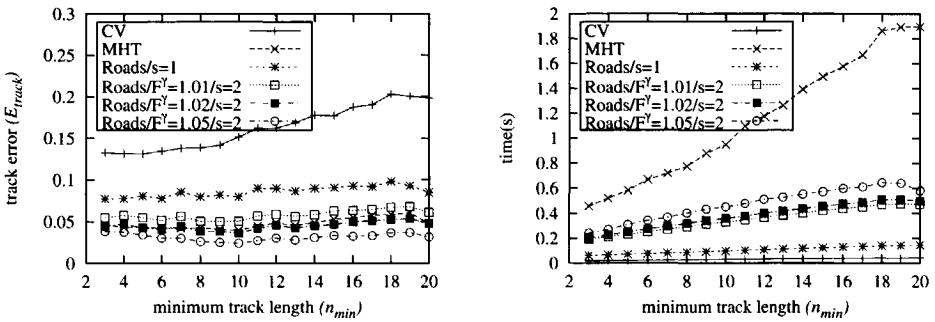
Variable Average-Track-Length Experiment

It can be expected that the initialization schemes of the algorithms are hampered if the tracks are short. To test this sensitivity, we carried out an experiment in which we varied the n_{min} parameter of the track generator (the average track length is proportional to n_{min}). In Fig. 4.9(a) the track error results are displayed. For the C&V algorithm we see that its performance decreases when the average track length increases. Although this is in contrast to our first hypothesis, this behavior can be expected since the measurement density increases when the average track length increases, which clearly makes the problem more complex. The other algorithms suffer from that problem too, as can be seen by the increase of the track error when $n_{min} > 10$. With respect to the initialization problems, we see that the Roads tracker and the MHT indeed display a track error that is slightly inversely proportional to n_{min}. Overall the Roads tracker clearly performs best for all settings of n_{min}. Computationally, the MHT suffers most from a high average track length as demonstrated in Fig. 4.9(b), again because of the increasing measurement density.

Variable Motion Dependence Experiment

So far, we have only considered independent point motion. In order to test the sensitivity of the algorithms to point motion dependence, we varied the number of point groups G. The measure of motion dependence is expressed as the point motion correlation coefficient ρ_m.

Proposition 5 *Given is a set of M points randomly distributed in G independent groups. The point motion correlation ρ_m between two points equals:*



(a) (b)

Figure 4.9: (a) displays the track error as a function of the minimum track length parameter of the modified PSMG. In (b) the corresponding computation time is displayed.

$$\rho_m = \frac{M/G - 1}{M - 1} \tag{4.13}$$

Proof: When we consider two points a and b, the correlation coefficient ρ_m between the motion vectors Y_a and Y_b of these points is defined as:

$$\rho_m = \frac{E[Y_a Y_b] - E[Y_a]E[Y_b]}{\sqrt{Var(Y_a)Var(Y_b)}} \tag{4.14}$$

Since all points move according to the same PDF, the following holds:

$$E[Y_a] = E[Y_b] = E[Y] \tag{4.15}$$

$$Var(Y_a) = Var(Y_b) = Var(Y) \tag{4.16}$$

Accordingly:

$$\rho_m = \frac{E[Y_a Y_b] - E[Y]^2}{Var(Y)} \tag{4.17}$$

Further, since the two points either belong to the same group or to different independent groups, $E[Y_a Y_b]$ can be rewritten as:

$$E[Y_a Y_b] = \begin{cases} E[Y^2], & \text{if } a \text{ and } b \text{ are from the same group} \\ E[Y]^2, & \text{if } a \text{ and } b \text{ are from distinct groups} \end{cases} \tag{4.18}$$

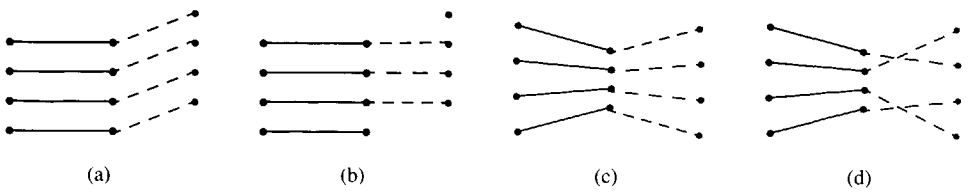


Figure 4.10: In the figure we show some difficult tracking examples. In (a) and (b) the points move in parallel and in (b) and (c) there are possibly some crossing point tracks. See text for explanation.

In other words:

$$E[Y_a Y_b] = P_s E[Y^2] + (1 - P_s) E[Y]^2, \tag{4.19}$$

where P_s is the probability that a and b are points from the same group. Combining (4.17) and (4.19) yields:

$$\rho_m = \frac{P_s E[Y^2] + (1 - P_s) E[Y]^2 - E[Y]^2}{Var(Y)} = P_s \frac{E[Y^2] - E[Y]^2}{Var(Y)} = P_s \tag{4.20}$$

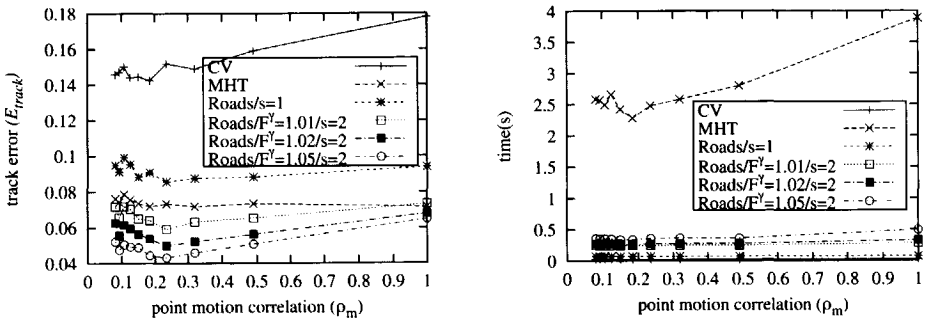
The probability P_s can be determined from the ratio of combinations of two points from the same group and the total number of combinations of two points, resulting in:

$$P_s = \frac{\binom{M/G}{2} G}{\binom{M}{2}} = \frac{\frac{M/G(M/G-1)}{2} G}{\frac{M(M-1)}{2}} = \frac{M/G - 1}{M - 1} \tag{4.21}$$

■

In case the points have correlated motion, we expect them to move more often in *parallel*. On the other hand, independently moving points will have more *crossing tracks*. Both types of motion have certain difficulties. That is, if correlated points move in parallel, they all have the same model deviation in the same direction, see for example Fig. 4.10(a). As a consequence it is possible that to limit the total cost it is better to choose a wrong alternative for all but one track and to terminate (or interpolate) the remaining track as in Fig. 4.10(b). Uncorrelated crossing point tracks, on the other hand, usually result in many assignment ambiguities because most individual motion models are inadequate in these cases, as can be seen in Fig. 4.10(c) and Fig. 4.10(d).

The variable point-motion dependence experiment was again carried out for all three algorithms. In Fig. 4.11 the results of the experiment are displayed. Fig. 4.11(a) shows that the trackers indeed suffer from high point-motion correlation, especially the C&V algorithm. Also with most trackers the error increases when the point motion is highly independent. For



(a) (b)

Figure 4.11: (a) displays the track error as a function of the point motion correlation ρ_m. In (b) the computation time is displayed for the same experiments.

each tracking algorithm the breaking point between the two mentioned tendencies occurs at a different correlation value. In general, the track error does not depend strongly on the point-motion correlation ρ_m. It has, however, to be noted that in order to make ρ_m the only problem variable, we did not clutter the correlated points in a restricted area. This would certainly have led to larger deviations in the track error and it would also be more realistic. However, the tracking of *cluttered* points is problematic, as was already demonstrated in the variable density experiment. Finally, only for the MHT the computation time is significantly higher when the point correlation is high, as can be seen in Fig. 4.11(b).

4.5.2 Image Sequence Experiments

For the real image sequence experiments, we used the PUMA sequence and the Toy-car sequence (courtesy of the University of Massachusetts). Although we analyzed many other sequences we restrict ourselves here to these sequences since they resulted in clearly noticeable differences for the algorithms. In both sequences the motion of the feature points is highly dependent. We used the corner detector as described in [28] in order to get the feature position information. We did not exploit color (brightness) information for any of the tracking algorithms[3].

4.5.3 PUMA Sequence

In Figure 4.12(a)-4.12(d), we show some images from the PUMA sequence and overlayed the detected feature positions. The PUMA sequence has rotational and strongly dependent

[3]The reference MHT as described in [4] exploits color information, but we did not use it here for fair performance comparison. Moreover, similar color validation schemes could be incorporated in the other algorithms too.

motion which violates the (second order) motion model of all algorithms. In Figure 4.13(a)-4.13(d), we show the tracking results of the C&V algorithm (a), the MHT (b) and the Roads tracker with $p_{min} = 2$ (c) and $p_{min} = 3$ (d). Although they are difficult to interpret we can conclude from these figures that the C&V algorithm produces many incorrect tracks. This is mainly caused by its d_{max} sensitivity (as demonstrated in [37]), since the point speed is widely divergent is this scene. Another reason is its greedy optimization scheme.

The MHT sometimes has problems to find the beginning of a track. Further, the MHT adjusts the track state more gradually than the Roads tracker, which can be both disadvantageous and beneficial. For instance, the MHT sometimes prefers straight lines, but on the other hand it makes less irregular tracks through spurious measurements. Especially when p_{min} is low as in Figure 4.13(c), the Roads tracker finds some invalid tracks. Though, when $p_{min} = 3$, the Roads tracker is able to track many more short tracks in the top right of the sequence in contrast to the MHT (see the tracks in the top right corner in Fig. 4.13(b) and Fig. 4.13(d)). With a higher p_{min} value the Roads tracker has in general less artifacts. Similar results can be achieved by lowering a_{max}.

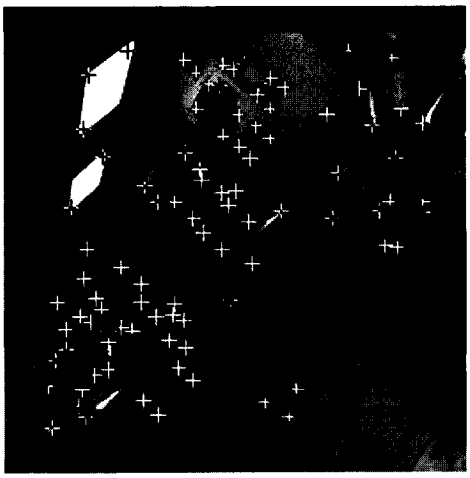
4.5.4 Toy-Car Sequence

In the Toy-car sequence a number of toy cars move quite fast in opposite directions. As a consequence some cars are partly occluded during a number of frames. The motion of the cars is mainly uniform. In Fig. 4.14 some images from the sequence are displayed. The tracking results are shown in Fig. 4.15. The C&V algorithm had among others problems with the occlusion, resulting in vertical and crossing tracks. The MHT again had problems to find the beginning of some of the tracks, e.g. the features at the rear wheel of the toy car that starts in the upper left corner. Besides, all trackers found some crossing tracks. For the Roads tracker this problem could be reduced by increasing p_{min} from $p_{min} = 3$ to $p_{min} = 4$ as can be seen in Fig. 4.15(c) and Fig. 4.15(d) respectively.

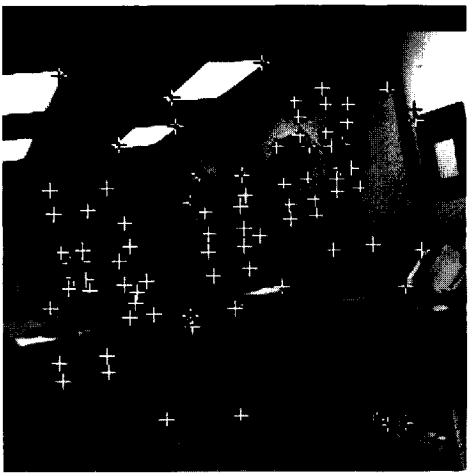
4.6 Conclusion

We addressed the problem of tracking a varying number of points over time through a monocular image sequence. We especially considered the case in which point appearance information is insignificant so that only motion information can be used to identify the points over time. The consequent problem that has to be solved is called the motion correspondence problem.

We formulated the problem as a multi-objective optimization problem with additional motion constraints and continuity constraints. The presented optimization algorithm (Roads tracker) inevitably makes approximations in order to make the optimization computationally feasible, because the optimization of one of the objectives is known to be NP-hard. Further, in the algorithm the objectives have been ordered in order to strive for a single best solution, since multi-objective optimization problems generally have a large set of mathematically equivalent solutions. Appropriate values for the constraint parameters depend on



(a) $m_1 = 94$

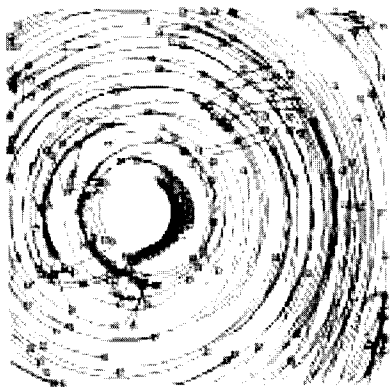


(b) $m_{10} = 83$

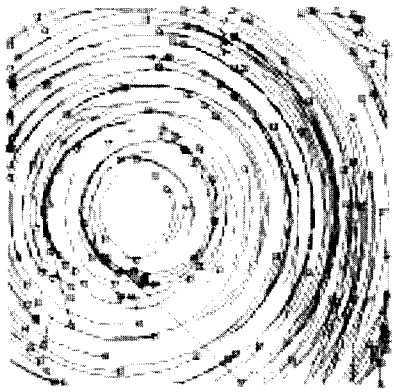
(c) $m_{20} = 83$

(d) $m_{30} = 87$

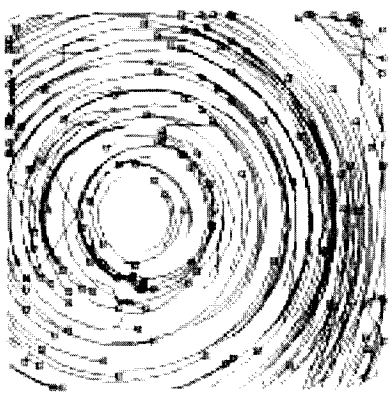
Figure 4.12: Frame 1 (a), 10 (b), 20 (c) and 30 (d) from the PUMA sequence.



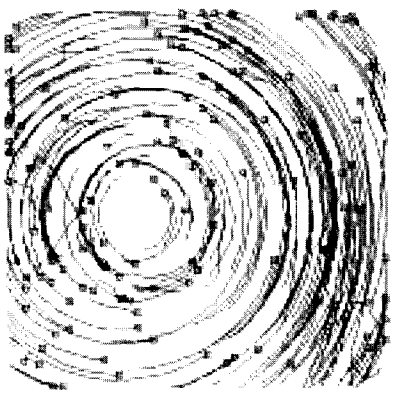
(a) C&V: 227 tracks



(b) MHT: 172 tracks



(c) Roads: 148 tracks



(d) Roads: 146 tracks

Figure 4.13: (a) Tracking results of applying the C&V algorithm, (b) the MHT, (c) the Roads tracker with $a_{max} = 2$, $p_{min} = 2$ and (d) the Roads tracker with $a_{max} = 2$, $p_{min} = 3$ to the features found in the PUMA sequence.

the application. Therefore, we explored the consequences of setting these values wrongly. In short, there is a trade-off between erroneous breaking up of a point track and connecting track parts of different points erroneously.

Both synthetic data experiments and real-image sequence experiments support the appropriateness of the proposed method. In all experiments we compared the performance of the method with other tracking algorithms. The synthetic experiments (with ground truth) were directed towards making an increasingly dense problem, increasing the average track length, and making the point motion increasingly dependent. In these experiments the proposed Roads tracker proved to be the most accurate. The MHT [24] consistently performed second best and the C&V algorithm [3] clearly had the largest track error. Computationally, we can conclude that the C&V algorithm is the fastest followed by the Roads tracker and the MHT, respectively.

The experiments with the real-image sequences showed that the C&V algorithm certainly has problems with dense point sets, occlusion and noise. Based on these sequences, it is, however, much harder to decide between the MHT and the Roads tracker. Partly, this is because of the lack of ground truth and partly because both algorithms make different (types of) errors. However, it can be concluded that the Roads tracker is easier to tune because it has a smaller number of parameters. Moreover, the MHT is much more sensitive to its parameter settings, as was already demonstrated in [37].

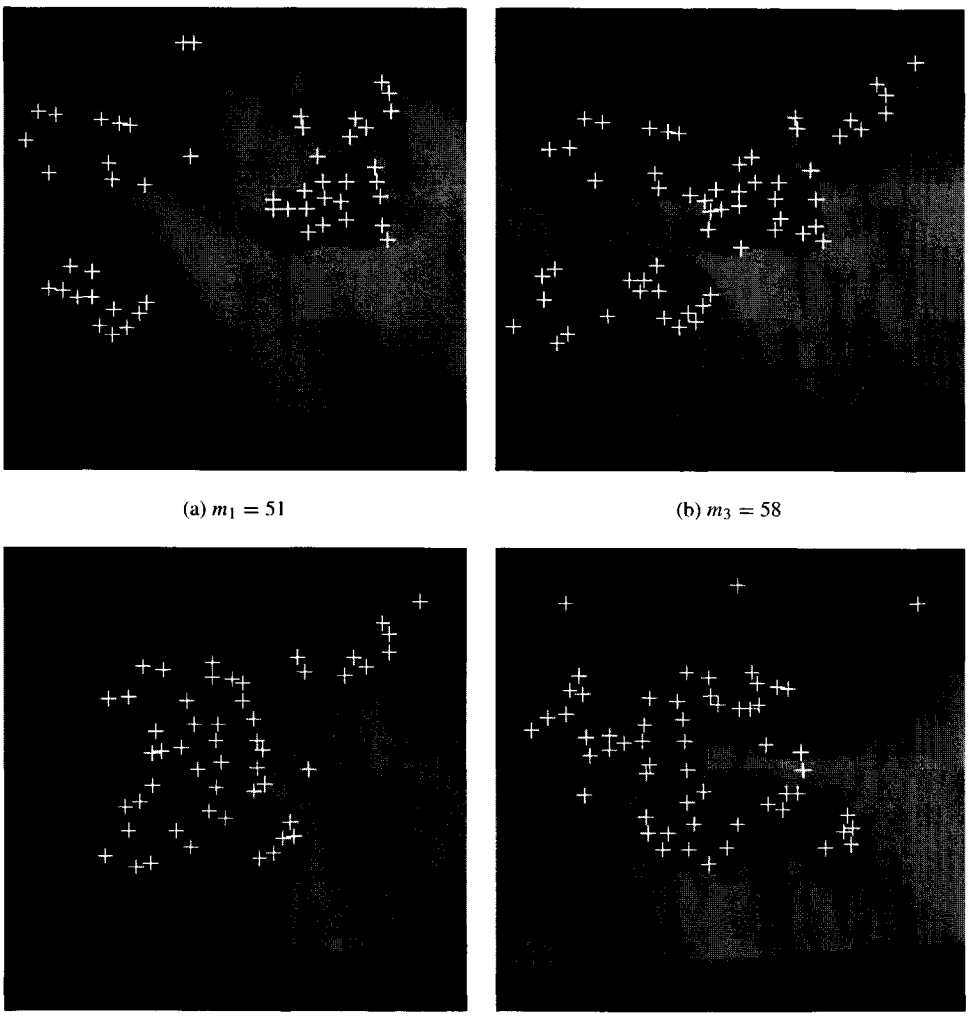
We consider a number of improvements to the proposed method. First, we suggest some modifications to improve the computational performance. By putting related tracks in separate groups, a number of track groups can be tracked independently, resulting in a lower exponential order of the multi-frame optimization scheme. Also optimizations to Murty's algorithm can be implemented as reported in [2] and [18].

Another improvement is the adjustment of the w parameter of the individual model *im2*. Either by estimation beforehand or by on-line adaptation, w can be fit to the motion of the points. A number of experiments we carried out (not included) support this idea.

Finally, in case significant color or texture information is available, it can be used for correspondence validation as in [4], [42], [45]. A small template at the measurement positions can be extracted and the similarity between the measurement template in the track at t_k and the templates of the candidate measurements at t_{k+1} can be computed. Then, only those candidate measurements that are closer than a certain distance with respect to the similarity measure are considered further. This modification will be both qualitatively and computationally beneficial.

Acknowledgments

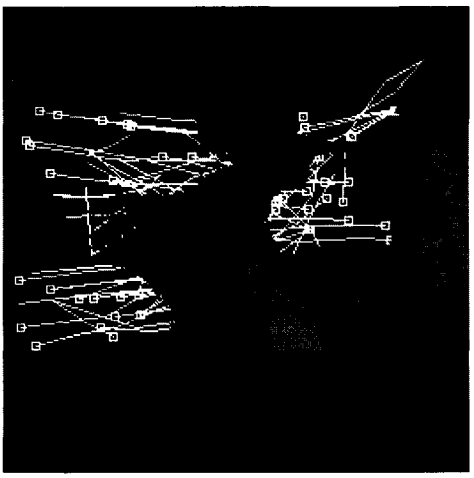
The authors would like to thank Dr. Reginald L. Lagendijk for the discussions on the influences of correlated point motion.



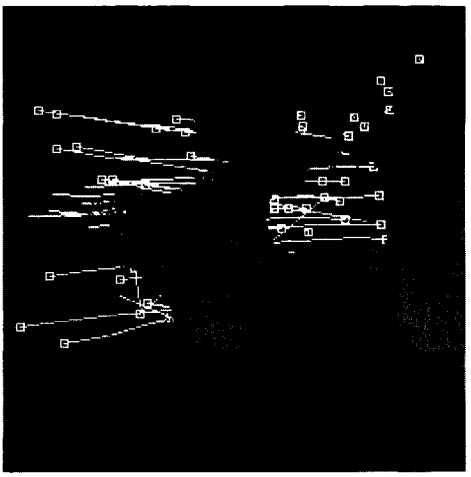
(a) $m_1 = 51$ (b) $m_3 = 58$

(c) $m_5 = 52$ (d) $m_7 = 58$

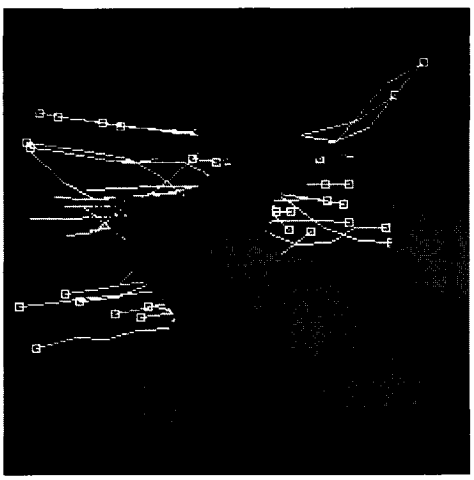
Figure 4.14: Frame 1 (a), 3 (b), 5 (c), and 7 (d) from the Toycar sequence.



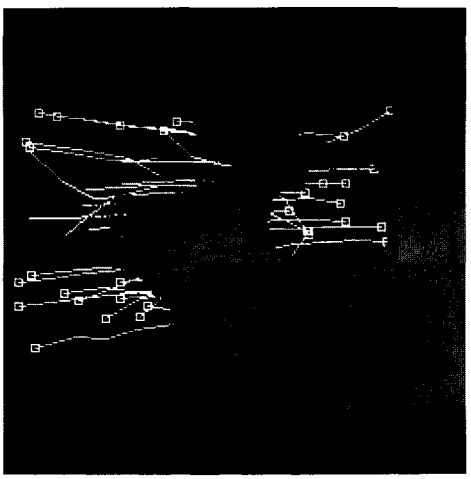
(a) C&V: 56 tracks



(b) MHT: 53 tracks



(c) Roads: 34 tracks



(d) Roads: 35 tracks

Figure 4.15: (a) Tracking results of applying the C&V algorithm, (b) the MHT, the (c) Roads tracker with $a_{max} = 2$, $p_{min} = 3$ and (d) the Roads tracker with $a_{max} = 2$, $p_{min} = 4$ to the features found in the Toycar sequence.

Appendix A

Multi-frame assignment optimization algorithm

```
Road(A^{k-1}, k, s, C_b, A^{k:s}_{sol}, E)
A^{k-1}   : assignment between previous and current frame
k         : frame number
s         : remaining scope
C_b       : cost bound for assignments in the remaining scope
A^{k:s}_{sol} : best solution for the remaining scope
E         : tracks in T that are enabled and may be extended
begin
    C^k_{min} = C^k(A^k_{min})                          ; find minimum cost assignment
    if s = 1 then                                       ; at lowest recursion level?
        if C^k_{min} < C_b then                         ; better than global bound?
            A^{k:s}_{sol} = ⟨A^k_{min}⟩                 ; update solution
        end
    else
        Y = ∅                                           ; set of processed matrices
        do
            A = A^k_{min}(Y)                            ; get next best with Murty
            Y = Y ∪ {A}                                 ; add to processed set
            C_0 = C^k(A)                                ; compute cost
            B = A^{k:s}_{sol}[2...s]                    ; get default solution
            B = Road(A, k + 1, s − 1, C_b − C_0, B, E)  ; call recursively to improve B
            A^{k:s} = ⟨A⟩ ∘ B                           ; concatenate A with new tail
            if C^{k:s}(A^{k:s}) < C_b then              ; better than global bound?
                C_b = C^{k:s}(A^{k:s})                  ; update global bound
                A^{k:s}_{sol} = A^{k:s}                 ; update solution
            end
            γ_{max} = min(F^γ_l C^k_{min}, F^γ_g C_b/s) ; compute cost-bound constraint
        while (Y ≠ U^k ∧ C_0 < C_b ∧ C_0 < γ_{max})     ; stop when global bound or
    end                                                 ; combined constraint exceeded
                                                        ; or no alternatives are left
    return A^{k:s}_{sol}                                ; return solution
end
```

Bibliography

[1] J.K. Aggarwal and N. Nandhankumar. On the computation of motion from sequences of images - a review. *Proceedings of the IEEE*, 76(8):917–935, August 1988.

[2] M. Bellmore and J.C. Malone. Pathology of traveling-salesman subtour-elimination algorithms. *Operations Research*, 19:278–307, 1971.

[3] D. Chetverikov and J. Verestóy. Feature point tracking for incomplete trajectories. *Computing, Devoted Issue on Digital Image Processing*, 62:321–338, 1999.

[4] I.J. Cox and S.L. Hingorani. An efficient implementation of Reid's multiple hypothesis tracking algorithm and its evaluation for the purpose of visual tracking. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 18(2):138–150, February 1996.

[5] J.L. Crowley, P. Stelmaszyk, and C. Discours. Measuring image flow by tracking edge-lines. In *Proceedings of the International Conference of Computer Vision*, pages 658–664, 1988.

[6] G.B. Dantzig. *Linear Programming and Extensions*. Princeton University Press, Princeton, 1963.

[7] S. Deb, K.R. Pattipati, and Y. Bar-Shalom. A new algorithm for the generalized multidimensional assignment problem. *IEEE International Conference on Systems, Man and Cybernetics; Emergent Innovations in Information Transfer Processing and Decision Making*, pages 249–254, 1992.

[8] R. Deriche and O. Faugeras. Tracking line segments. In *Proceedings European Conference on Computer Vision*, pages 259–268, 1990.

[9] T.E. Fortmann, Y. Bar-Shalom, and M. Sheffe. Sonar tracking of multiple targets using joint probabilistic data association. *IEEE Journal of Oceanic Engineering*, 8(3):173–184, July 1983.

[10] K.I. Hodges. Adaptive constraints for feature tracking. *Monthly Weather Review*, 127:1362–1373, 1998.

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

[12] T.S. Huang and A.N. Netravali. Motion and structure from feature correspondences: A review. *Proceedings of the IEEE*, 82(2):252–268, February 1994.

[13] V.S.S. Hwang. Tracking feature points in time-varying images using an opportunistic selection approach. *Pattern Recognition*, 22(3):247–256, 1989.

[14] R. Jonker and A. Volgenant. A shortest augmenting path algorithm for dense and sparse linear assignment problems. *Computing*, 38:325–340, 1987.

[15] R.M. Karp. Reducibility among combinatorial problems. In R.E. Miller and J.W. Thatcher, editors, *Complexity of Computer Computations*, pages 85–103. Plenum Press, New York, 1972.

[16] H.W. Kuhn. The hungarian method for solving the assignment problem. *Naval Research Logistics Quarterly*, 2:83–97, 1955.

[17] R. Mehrotra. Establishing motion-based feature point correspondence. *Pattern Recognition*, 31(2):23–30, 1998.

[18] M.L. Miller, H.S. Stone, and I.J. Cox. Optimizing Murty's ranked assignment method. *IEEE Transactions on Aerospace and Electronic Systems*, 33(3):851–861, 1997.

[19] K.G. Murty. An algorithm for ranking all the assignments in order of increasing cost. *Operations Research*, 16:682–687, 1968.

[20] H.H. Nagel. Displacement vectors derived from second order intensity variations in image sequences. *Computer Vision Graphics and Image Processing*, 21(1):85–117, 1983.

[21] R. Pless, T. Brodský, and Y. Aloimonos. Detecting independent motion: The statistics of temporal continuity. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8):768–773, August 2000.

[22] A.B. Poore. Multidimensional assignments and multitarget tracking. In *Partitioning Data Sets; DIMACS Workshop*, pages 169–196, New Brunswick, USA, 1995. American Mathematical Society.

[23] K. Rangarajan and M. Sha. Establishing motion correspondence. *CVGIP: Image Understanding*, 24(6):56–73, July 1991.

[24] D.B. Reid. An algorithm for tracking multiple targets. *IEEE Transactions on Automatic Control*, 24(6):843–854, December 1979.

[25] V. Salari and I.K. Sethi. Feature point correspondence in the presence of occlusion. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(1):97–91, January 1990.

[26] I.K. Sethi and R. Jain. Finding trajectories of feature points in a monocular image sequence. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 9(1):56–73, January 1987.

[27] M. Shah and R. Jain. *Motion-Based Recognition*. Kluwer Academic Publishers, Norwell, Massachusetss, USA, 1997.

[28] J. Shi and C. Tomasi. Good features to track. In *Proceedings IEEE Conference on Computer Vision and Pattern Recognition*, pages 593–600, 1994.

[29] P. Smith and G. Buechler. A branching algorithm for discriminating and tracking multiple objects. *IEEE Transactions on Automatic Control*, 20:101–104, February 1975.

[30] C. Stauffer and W.E.L. Grimson. Learning patterns of activity using real-time tracking. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8):747–757, August 2000.

[31] R.E. Steuer. *Multiple Criteria Optimization: Theory, Computation, and Application*. John Wiley and Sons, Inc., 1986.

[32] C. Tomasi and T. Kanade. Shape and motion from image streams under orthography: a factorization method. *International Journal of Computer Vision*, 9(2):137–154, 1992.

[33] R.Y. Tsai and T.S. Huang. Estimating three dimensional motion parameters of a rigid planar patch, III: Finite point correspondences and the three view problem. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 32:213–220, 1984.

[34] S. Ullman. *The Interpretation of Visual Motion*. M.I.T. Press, Cambridge, MA, 1979.

[35] C.J. Veenman, E.A. Hendriks, and M.J.T. Reinders. A fast and robust point tracking algorithm. In *IEEE International Conference on Image Processing*, volume III, pages 653–657, Chicago, USA, October 1998.

[36] C.J. Veenman, M.J.T. Reinders, and E. Backer. Establishing motion correspondence using extended temporal scope. *Submitted to Artificial Intelligence*, 2000.

[37] C.J. Veenman, M.J.T. Reinders, and E. Backer. Resolving motion correspondence for densely moving points. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(1):54–72, January 2001.

[38] J. Verestóy and D. Chetverikov. Experimental comparative evaluation of feature point tracking algorithms. In *Proceedings Workshop on Evaluation and Validation of Computer Vision Algorithms*, pages 183–194. Kluwer series in Computational Imaging and Vision, 2000.

[39] J.A. Webb and J.K. Aggarwal. Structure from motion from rigid and jointed objects. *Artificial Intelligence*, 19:107–130, 1982.

[40] D.B. West. *Introduction to Graph Theory*. Prentice-Hall, 1996.

[41] L. Wixson. Detecting salient motion by accumulating directionally-consistent flow. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8):774–780, August 2000.

[42] Y.S. Yao and R. Chellappa. Tracking a dynamic set of feature points. *IEEE Transactions on Image Processing*, 4(10):1382–1395, 1995.

[43] Z. Zhang. Estimating motion and structure from correspondences of line segments between two perspective images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17(12):1129–1139, 1995.

[44] Z. Zhang and O. Faugeras. Three-dimensional motion computation and object segmentation in a long sequence of stereo frames. *International Journal of Computer Vision*, 7(3):211–241, 1992.

[45] Q. Zheng and R. Chellapa. Automatic feature point extraction and tracking in image sequences for arbitrary camera motion. *International Journal of Computer Vision*, 15:31–76, 1995.

Chapter 5

A Cellular Coevolutionary Algorithm for Image Segmentation

This chapter has been submitted to *IEEE Transactions on Image Processing*, October 2001.

Abstract Clustering is inherently a difficult problem, both with respect to the definition of adequate models as well as to the optimization of the models. In this paper we present a model for the cluster problem that does not need knowledge about the number of clusters a priori. This property is among others useful in the image segmentation domain, which we especially address. Further, we propose a cellular coevolutionary algorithm for the optimization of the model. Within this scheme multiple agents are placed in a regular 2-D grid representing the image, which imposes neighboring relations on them. The agents cooperatively consider pixel migration from one agent to the other in order to improve the homogeneity of the ensemble of the image regions they represent. If the union of the regions of neighboring agents is homogeneous then the agents form alliances. On the other hand, if an agent discovers a deviant subject, it isolates the subject. In the experiments we show the effectiveness of the proposed method and compare it to other segmentation algorithms. The efficiency can easily be improved by exploiting the intrinsic parallelism of the proposed method.

Keywords: clustering, image segmentation, modeling, distributed genetic algorithms.

5.1 Introduction

Clustering is an important and difficult task in unsupervised pattern recognition. The clustering problem comes down to finding a separation of a set of objects into an a priori unknown number of subsets while minimizing intra-cluster variability (within scatter) and maximizing the inter-cluster variability (between scatter). There is a huge amount of literature on the subject, ranging from models, algorithms, algorithm parameter estimations to cluster validity studies [19], [51]. The clustering methods can be divided up into exclusive and non-exclusive methods [35]. The best-known non-exclusive method is the fuzzy C-means model [20]. In this method objects are soft clustered such that objects belong to all clusters to a certain degree. For an overview of fuzzy clustering methods see for example [5] and [6]. In exclusive clustering methods, the objects are partitioned into a number of (crisp) subsets, such that each object belongs to exactly one subset. We concentrate on exclusive clustering methods, among which the K-means model (or hard C-means) [37] is the most widely used. As the fuzzy C-means model, the K-means model assumes that the number of clusters is known a priori. There are, however, numerous domains for which this assumption cannot be satisfied. One of these is the image segmentation problem, which we especially address in this paper. In (region-based) image segmentation, pixels are clustered based on their color or texture information, while a hard constraint is imposed on spatial cluster (segment) connectivity. Throughout this paper, we will consider the clustering problem and the segmentation problem as being similar. Accordingly, we consider solution methods for both problems interchangeably.

Since the clustering problem is a known NP-hard problem, deterministic algorithms for the fuzzy C-means and the K-means model use a greedy optimization scheme in order to find a suboptimal solution of their criterion function. Many stochastic optimization schemes that aim at a global maximum have been reported, among which simulated annealing meth-

ods [11], [33], [49] and evolutionary algorithms [17], [24], [26], [29], [36], [54]. As part of some evolutionary approaches also certain domain specific recombination operators have been reported [9], [10], [34], [53].

In this paper, we introduce a cluster model which aims at minimal intra-cluster variability. Additionally, instead of maximizing the inter-cluster variability we impose a hard constraint on the intra-cluster variability for the union of two clusters. We claim that such a constraint is inevitable in order find useful solutions to the cluster problem. As a result the proposed model allows for the clustering of a data set into an a priori unknown number of clusters. Additionally, we specialize the model for image segmentation and propose a *cellular coevolutionary algorithm* (CCA) to optimize the image segmentation model in a distributed way.

The outline of the paper is as follows. After first exploring the characteristics of the clustering problem, we propose a new cluster model in Section 5.2, Then, in Section 5.3, we specialize the model for image segmentation. We give an overview of related work in distributed evolutionary computation while focusing on image segmentation in Section 5.4. In Section 5.5, we describe a coevolutionary algorithm for the optimization of the proposed model. In the experiments in Section 5.6, we demonstrate the effectiveness of the method and compare its performance to some other image segmentation algorithms, both with synthetic and natural images. We finalize the paper with a discussion and some concluding remarks in Section 5.7.

5.2 Cluster Model

In this section, we elaborate on some fundamental characteristics of the cluster problem. We will not consider the definition of an appropriate distance measure, which is a known problem especially when multi-variate features are involved. The characteristics that we focus on are manifest, irrespective of the used distance measures. Before going into detail, we first define the clustering problem.

Given is a data set $X = \{\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_N\}$, where $\mathbf{x}_i$ is a feature vector in a p-dimensional metric space, and $N = |X|$ is the number of objects in X. Then, a valid clustering of X in a set of clusters $C = \{C_1, C_2, ..., C_M\}$, where M is the number of clusters, has the following partition properties:

P-1. $C_i \neq \emptyset, 1 \leq i \leq M$

P-2. $\bigcup_{i=1}^{M} C_i = X$

P-3. $C_i \cap C_j = \emptyset, i \neq j, 1 \leq i, j \leq M$

Additionally, the clusters should reflect the structure of the data such that objects in the same cluster are similar to each other and objects from distinct clusters are different from each other. In order to find a solution to the clustering problem, we need a quantitative way to distinguish between similar and dissimilar objects or, in other words, we need to quantitatively differentiate between homogeneous and inhomogeneous sets of objects. In the literature various alternatives have been reported to approach this task [19], [32], [51].

A common criterion to quantify cluster homogeneity is the sum-of-squared-error criterion:

$$\sum_{i=1}^{M} \sum_{\mathbf{x} \in C_i} \|\mathbf{x} - \mu(C_i)\|^2 \tag{5.1}$$

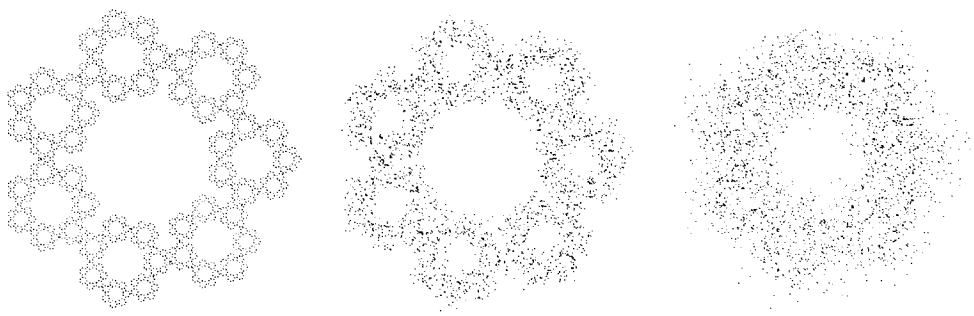
$$\text{where} \quad \mu(Y) = \frac{1}{|Y|} \sum_{\mathbf{x} \in Y} \mathbf{x}$$

Without additional constraints (5.1) has a zero minimum for $M = |X|$. Therefore, in addition to minimizing this criterion, many cluster models, like the K-means and fuzzy C-means model, assume the number of clusters M to be known beforehand. There is, however, also a number of methods that among others assume a certain maximum variability per cluster, such as the split-and-merge and region growing algorithms in the image segmentation domain. We claim that objective clustering is not possible without such additional parameters because of a *scale problem*. We use Fig. 5.1(a) to support this observation. It is impossible to decide which is the right data clustering for the data set displayed in the figure: should this data set be partitioned into 1, 7, 49, or even more clusters? For synthetic data sets with self-similar structures like those displayed here, the problem is clearly undecidable. This may seem an academic problem that does not correspond to real life, however, if we consider the image in Fig. 5.2, we also see a nesting of structures. Again the number of clusters (segments) is arguable. For example, does the image only contain a tree, houses, garden, and sky, or must a valid segmentation result also include clouds, bushes, windows, or even smaller segments/clusters like the flowers and the leaves of the tree?

A typical approach to discover a significant set of clusters is to minimize (5.1) for a range of settings of M, where $1 \leq M \leq M_{max}$. Then, a cluster validity study [8], [16], [21], [31] can help in selecting the 'true' number of clusters by looking for sharp knees or local minima in a cluster validity index function curve, e.g. the Davies-Bouldin index [16]. However, these index functions can have multiple local minima and knees, or they sometimes contain no significant transition at all. Moreover, the deepest local minimum or the sharpest knee are by no means indications of the scale the user expects.

In addition to the scale problem, in many practical situations the features of the objects are noisy. Because the intra-cluster variability of the data increases proportionally with the amount of noise, the determination of the true cluster borders can be severely hampered, leading to a *noise problem*. In some cases the noise can be reduced by filtering, but the uncertainty principle [52] is a limiting factor. That is, by reducing the noise on the features we also decrease the accurateness of the estimated cluster border.

Both the scale and the noise problem should be properly handled in a clustering method in order to find objectively good cluster results. Both aspects are interrelated, since dealing with one problem affects the other. That is, the smallest possible scale depends on the effective noise level. Usually, both problems are implicitly attacked in the same way, which implies that the assumed optimal scale is the one just above the noise level. We illustrate this with the data sets shown in Fig. 5.1(b) and Fig. 5.1(c). If noise is considered as the limiting factor,



(a) (b) (c)

Figure 5.1: In (a) a noiseless 2-D data set consisting of a nested circular structure. In (b) and (c) the same data set is displayed with an increasing amount of Gaussian noise. The number of distinguishable clusters decreases accordingly.

Figure 5.2: An image from the Flower-Garden sequence. Also in this natural image the number of segments is clearly arguable.

then Fig. 5.1(b) contains 7 clusters and Fig. 5.1(c), which has even more noise, contains only one cluster. Once again, also in Fig. 5.1(b) and Fig. 5.1(c) the problem is undecidable, since the original data is the same as in Fig. 5.1(a).

In the image segmentation domain, in which we are especially interested, the number of clusters is not known a priori. Therefore, we do not want to fix the number of clusters in our model. However, as already mentioned, both straight minimization of the intra-cluster variability and maximization of the inter-cluster variability lead to undesirable trivial solutions, being N clusters or 1 cluster, respectively. We choose to minimize the intra-cluster variability while at the same time constraining the intra-cluster variability of the union of two clusters. In this way, this intra-cluster variability constraint defines the scale at which two clusters can be differentiated from each other.

To minimize the intra-cluster variability, we use the sum-of-squared-error criterion (5.1). Here, we rewrite (5.1) such that it fits better to the model and the presented algorithm in the remainder. Further, we implement the joint intra-cluster variability constraint as a minimum variance for the union of two clusters. This leads to the following cluster model:

$$\min_{C} \sum_{C_i \in C} |C_i| Var(C_i), \tag{5.2}$$

where

$$Var(Y) = \frac{1}{|Y|} \sum_{\mathbf{x} \in Y} \|\mathbf{x} - \mu(Y)\|^2 \tag{5.3}$$

subject to

$$\forall C_i, C_j, i \neq j : Var(C_i \cup C_j) \geq \sigma_{max}^2 \tag{5.4}$$

In general, the optimization of this model leads to clusters having a variance below σ_{max}^2. However, there are some rare situations in which the variance of individual clusters can exceed this limit.

Proposition 6 *For a solution to the constrained optimization problem as defined in (5.2) and (5.4) the following holds:*

$$Var(C_a) \geq \sigma_{max}^2 \Rightarrow \forall \mathbf{x} \in C_a : \left(\|\mathbf{x} - \mu(C_a)\|^2 < \sigma_{max}^2 \vee \exists C_b : Var(C_b \cup \{\mathbf{x}\}) < \sigma_{max}^2\right) \tag{5.5}$$

Proof: For the proof we use the inverse implication. So we prove that the variance of a cluster C_a will not exceed σ_{max}^2 if all remote cluster objects, that is, cluster objects for which holds:

$$\|\mathbf{x} - \mu(C_a)\|^2 \geq \sigma_{max}^2 \tag{5.6}$$

are outside the σ^2_{max} range of another cluster, i.e.:

$$\neg\left(\exists C_b : Var(C_b \cup \{\mathbf{x}\}) < \sigma^2_{max}\right) \tag{5.7}$$

Let C be the optimal clustering resulting from the minimization of (5.2) subject to (5.4) and $z = \sum_{C_i \in C} |C_i| Var(C_i)$. Further, assume that there is a cluster C_a for which $Var(C_a) \geq \sigma^2_{max}$. Then, $\exists \mathbf{x} \in C_a$ such that $Var(C_a - \{\mathbf{x}\}) < Var(C_a)$. Especially all objects $\mathbf{x}$ for which $\|\mathbf{x} - \mu(C_a)\|^2 \geq \sigma^2_{max}$ are candidates. Let $C'_a = C_a - \{\mathbf{x}\}$ and $\{\mathbf{x}\}$ form two separate clusters. Now, still $Var(C'_a \cup \{\mathbf{x}\}) \geq \sigma^2_{max}$ and since (5.7) applies, also in general $\forall C_i$: $Var(C_i \cup \{\mathbf{x}\}) \geq \sigma^2_{max}$. However, since $Var(\{\mathbf{x}\}) = 0$ the minimum $z' < z$. This is contradictory to the assumptions, so given (5.6) and (5.7), there is *no* cluster C_a for which $Var(C_a) \geq \sigma^2_{max}$. ∎

Informally, the implication of this proposition is that when a dispersed cluster is enclosed by compact clusters, these compact clusters may prevent the dispersed cluster from splitting. In other words, the presence of the compact clusters makes the cluster separation ambiguous since in that case, the homogeneity criterion (5.2) is at odds with the joint variability constraint (5.4).

Since the cluster variance can both be caused by the distribution of the data and by the noise in the data acquisition process, knowledge about the noise can be exploited in this model. We elaborate on dealing with noise in Section 5.5, where we describe a specialized algorithm for image segmentation.

5.3 Image Segmentation Model

In this section we refine the just introduced cluster model for the image segmentation problem. The image segmentation problem is a special clustering problem where the objects are picture elements (pixels) and the feature vector consists of the pixel position information together with pixel appearance information.

The segmentation problem is defined differently from the cluster problem, since it requires that the segments are spatially connected. Several segmentation methods only impose a soft constraint on segment connectivity, e.g. [14], [22], [40], [41]. Accordingly, these methods are merely quantization methods, though the problems are similar to a certain extent. There are two additional reasons why we impose connectivity as a hard constraint. First, connected segments are usually bigger so that more *reliable statistics* can be maintained. Second, the optimization problem simplifies into a *one criterion optimization* problem, since we do not need to optimize a connectivity criterion as well. Moreover, we avoid the introduction of an additional parameter that is needed in case the resulting multi-criterion optimization problem is solved by weighting a homogeneity and a connectivity term.

We first define the image segmentation problem as a graph-partitioning problem. We represent the image as an undirected graph G, where the N vertices represent the appearance feature vectors of the pixels. For convenience we denote the vertices by the appearance feature vectors $\mathbf{x}_i$. The pixel position information is represented in the edges of the graph.

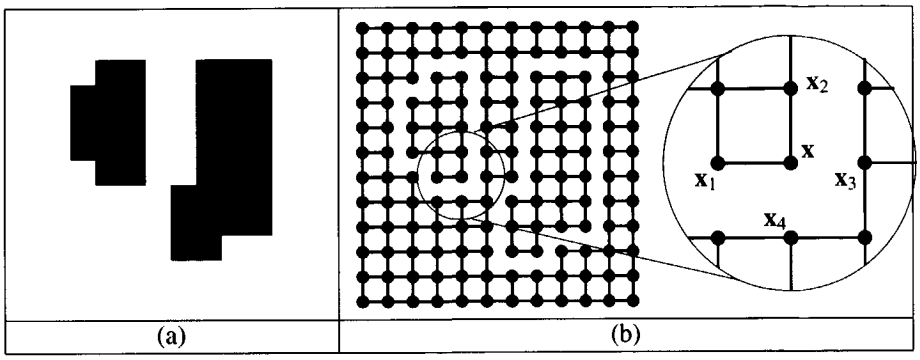


Figure 5.3: (a) is an image containing a background and two 'objects', making three segments in total. (b) shows the corresponding segmentation graph and a blown up detail, where $\mathbf{x}$ has two adjacent vertices $\mathbf{x}_1$ and $\mathbf{x}_2$ and four contiguous vertices, $\mathbf{x}_1$, $\mathbf{x}_2$, $\mathbf{x}_3$, and $\mathbf{x}_4$.

Further, we define *contiguous vertices* as being contiguous on the image grid, so every vertex has exactly four contiguous vertices[1]. We write $\mathbf{x}_1 \rightleftharpoons \mathbf{x}_2$ if $\mathbf{x}_1$ and $\mathbf{x}_2$ are contiguous. Only if two vertices are contiguous, there can be an edge connecting them, which makes them adjacent. That is, every vertex can have at most four edges to contiguous vertices. In Fig. 5.3 we illustrate the relation between contiguous and adjacent vertices.

Definition 1 *A* segmentation graph *is a graph for which the following holds: if there is a path between two vertices and the vertices are contiguous, then the vertices are adjacent, i.e. there is an edge between the vertices.*

Every maximally connected subgraph (component) in a candidate segmentation graph is a candidate segment S_i, where $1 \leq i \leq M$ (and M being the number of segments). A candidate segment S_i has a set of vertices $C_i = V(S_i) = \{\mathbf{x}_1, ..., \mathbf{x}_{N_i}\}$, where N_i is the number of vertices in S_i.

A consequence of the hard connectivity constraint is that (5.4) only has to be satisfied for *contiguous components*, where components S_1 and S_2 are contiguous: $S_1 \rightleftharpoons S_2$ if $\exists \mathbf{x}_1 \in C_1, \exists \mathbf{x}_2 \in C_2 : \mathbf{x}_1 \rightleftharpoons \mathbf{x}_2$, with $C_1 = V(S_1)$ and $C_2 = V(S_2)$.

The objective of the segmentation task is to find a segmentation graph that represents homogeneous appearing regions. In this study we do not consider color images, so the regions must be homogeneous with respect to their brightness or intensity values. In the general cluster model, we stated that a certain variability must be allowed for to establish a certain cluster scale. However, if the intensity feature is noisy the image variability caused by the noise must be incorporated too. So, in addition to the given intra-cluster variability to establish the desired scale, we assume that a variability is given due to noise. We measure both variabilities as pixel intensity variance present at the given scale (σ_s^2) or resulting from the amount of

[1] If eight connectivity is desired, then a slightly different definition for contiguity should be applied.

noise (σ_n^2), respectively. Then, $\sigma_{max}^2 = \sigma_s^2 + \sigma_n^2$, since the underlying processes (signal and noise) can be assumed independent.

5.4 Distributed Genetic Algorithms

Before describing the cellular coevolutionary algorithm (CCA) for the optimization of the proposed image segmentation model, we first shortly overview the research in parallel and distributed evolutionary computation. For a more elaborate survey on parallel genetic algorithms, see [12]. In structuring the work in the distributed evolutionary field we propose a distinction between *complete solution models* and *partial solution models*. By complete solution models we mean methods that have a population of chromosomes that each encode a complete problem solution. In partial solution models the chromosomes from a number of populations together encode a complete solution. This classification resembles the Michigan and Pittsburgh approach for single population learning classifier systems. In the Michigan approach [28] a population evolves competing classification rules, while in the Pittsburgh approach [50] complete rule sets compete each other. Here, we use the terms complete and partial solution models to classify *parallel and distributed* evolutionary computation models.

5.4.1 Complete Solution Models

Though the original evolutionary algorithms are assumed to have inherent parallelism in their search mechanism, also explicit parallel or distributed evolutionary models have been proposed. On the one hand these models serve to ease the implementation of evolutionary algorithms on parallel and distributed computer architectures, but they also offer some interesting opportunities to explore the search space in a different way or just allow for delay of convergence. First, single-population master-slave models only aim at gain in computational efficiency. More interesting are the models with communicating populations, which we consider in the remainder. First, coarse-grained parallel models (island model) typically consist of a number of standard genetic algorithms in a certain topology. Once in a while individuals migrate from one population to the other in order to increase diversity in the populations [1], [13]. Sometimes the individuals only migrate to populations that are neighboring in the topology. This aspect is further exploited in fine-grained or cellular genetic algorithms [13], [23], [38]. In cellular genetic algorithms each individual from the population occupies a position on a spatial structure (e.g. a 2-D grid) and only recombines with individuals in a restricted neighborhood. Then, within the spatial structure, similar solutions emerge in restricted areas.

More recently, coevolutionary models have been developed consisting of interacting populations, from which the host-parasite model is the best known [4], [15], [25], [27], [39], [46], [47], [48]. In this model there is a host population and a parasite population and the fitness of the individuals in one population depends on the individuals in the other population. Finally the solution emerges in the host population.

5.4.2 Partial Solution Models

Partial solution models only exist in coevolutionary approaches, unless the partial solutions can be optimized completely independently. In general, however, if partial solutions are to be composed into a complete solution, the partial solutions are dependent. Accordingly, the populations coevolve the partial solutions in a competitive or cooperative way [45], [55]. Also partial solution models can be coarse- or fine-grained. An example of a coarse-grained model is described by Potter and De Jong [45]. In their model cooperative populations (species) coevolve complete solutions to function optimization problems, where each individual population optimizes a different function parameter.

Distributed Evolutionary Image Segmentation

To our knowledge, fine-grained partial solution models have only been reported in the image segmentation domain. Not surprisingly, since in images the pixels are explicitly structured in a 2-D grid and the colors are locally correlated. In [2] and [3] Andrey and Tarroux describe a distributed genetic algorithm (DGA) for image segmentation. In [2] it is used to find homogeneous regions in grey images, while in [3] an Markov random field texture model is incorporated to segment textured images. The described method optimizes segment model parameters without a constraint on the number of segments. Additionally, segments expand and distribute by a fitness-driven selection operator that copies chromosomes, including their segment index, to neighboring image locations and occasionally also to more remote image locations. Consequently, segments may become disconnected, which violates the strict definition of the segmentation problem. The optimization process itself, which is restricted to chromosomes in the same segment, is similar to that in cellular genetic algorithms. An important difference with the model proposed in this paper is that the optimization within a segment is not based on statistics of the whole segment. Instead only a predefined window size around a pixel is considered. Moreover, the selection operator that copies local model parameters and the segment label applies in a restricted neighborhood regardless of the color model similarities. Both the window size and the genetic operator neighborhood size serve as noise and scale parameters.

Peng et al. [43], [44] report a method derived from the DGA [2], [3]. An important difference is that in their so-called hierarchical distributed genetic algorithm (HDGA) the segment index (label) of the pixel is an essential part of the fitness function. The algorithm aims at minimizing the differences between pixel labels, while the pixel labels are initialized as quantization levels using some histogram dichotomy scheme. Here, the size of the neighborhood area serves as scale and noise parameter. This model resembles the Markov random field image models (MRF) in [14], [18], [22]. The main difference is that Peng et al. use a cellular GA scheme for the optimization of the *local fitness* function. Besides, as MRF segmentation methods, the described method is a quantization method rather than a segmentation method, since connectivity is not guaranteed (soft constraint).

5.5 Algorithm

We now present the cellular coevolutionary algorithm (CCA) to optimize the model described in the previous sections. The proposed model has two important parameters, being the scale-derived image variance (σ_s^2) and the noise variance (σ_n^2). We deal with noise by simply pre-filtering the image with a uniform filter with size f. Consequently, the σ_{max}^2 parameter from (5.4) is affected in the following way:

$$\sigma_{max}^2 = \sigma_s^2 + \frac{\sigma_n^2}{f}. \tag{5.8}$$

since the total variance changes accordingly.

The proposed algorithm dynamically aims at optimizing (5.2), while satisfying (5.4) as well as the connectivity requirement. Accordingly, the algorithm has only one essential parameter, being σ_{max}^2. The segments are maintained by active entities that autonomously try to optimize local criteria. We call these entities *agents* in order to stress their autonomous behavior. For efficiency reasons, each agent manages the outside border $B_i = \{\mathbf{x}_1, ..., \mathbf{x}_{|B_i|}\}$ of the component it represents, where $|B_i|$ is the border length. This border is a list of vertices that are contiguous but *not* adjacent to the component the agent represents. In other words, B_i contains candidate vertices with which the agent's component can be extended. In the following sections, we describe how the segments are initially created, how they evolve during the optimization process, and how the process terminates.

5.5.1 Creation

Let the image have dimensions $W \times H$. Upon initialization the image is converted into a regular grid with $W \times H$ unconnected vertices, hence $N = W \times H$. Accordingly, the initial segmentation graph has as many components as vertices. Further, N agents are created and each component is assigned to an agent. After creation the evolution process starts.

5.5.2 Evolution

The evolution process is a sequence of epochs. During every epoch all agents (conceptually) act in parallel or alternatively sequentially in random order. An agent can take one of three actions. Alliances with neighboring agents are formed if the σ_{max}^2 parameter allows it. The consequent component merge satisfies the scale condition (5.4). If no alliances are possible, an agent considers subject migration from a neighboring agent if this improves the global criterion (5.2). Now, no check is done whether the scale condition is violated. Afterwards (in a later epoch), subject isolation is considered, if in the course of subject migrations the variance exceeds σ_{max}^2. Before considering one of these actions the agent estimates its probability of success as the ratio of successful trials s_i in the last m trials. We add a rest probability r because the tides may turn after a period of failure.

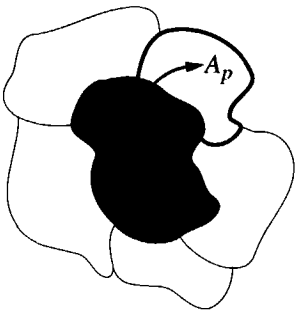


Figure 5.4: Schematic illustration of alliance formation. The agent in the center is the one that attempts to form an alliance.

$$P_i = (1-r)\frac{s_i}{m} + r \tag{5.9}$$

The agent will act in this epoch only if $P_i > U(0...1)$, where $U(0...1)$ is a uniformly distributed number in the range [0...1].

Alliance Formation: Component Merge

An agent first tries to form an alliance with any of its neighboring agents. Agents form alliances with their neighbors if the variance of the union of their components remains under the predefined maximum σ^2_{max} (5.4). Accordingly, for all k_i neighbors the joint component variance is computed:

$$\sigma^2_{ij} = Var(C_i \cup C_j) \tag{5.10}$$

Where the alliance that results in the lowest joint variance is selected. That is:

$$p = \arg\min_{j=1}^{k_i} \sigma^2_{ij} \tag{5.11}$$

If the variance of the selected alliance satisfies (5.4), then agent A_i and A_p become allies, see Fig. 5.4. We let A_i represent the alliance by taking A_p's component. Then, the neighboring agent A_p no longer represents a graph component, so it terminates. Consequently, the number of segments M decreases.

If the selected alliance violates (5.4), then the merge action fails for agent A_i and the agent considers subject migration.

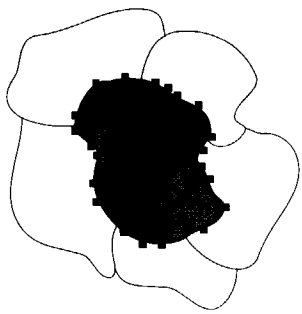


Figure 5.5: Schematic illustration of subject migration. The agent in the center is collecting a number of candidates at its border.

Subject Migration: Vertex Exchange

The aim of the subject migration is to contribute to the minimization of the global criterion (5.2). First, a number of candidate vertices are collected and the one that results in the highest gain is selected.

Agent A_i collects b_i candidate vertices from its border B_i by random sampling, see also Fig. 5.5. It is important to note that at this point, the *greed* of the algorithm can be controlled. Considering all border objects ($b_i = |B_i|$) as candidate would make the algorithm very greedy, while taking very few objects restrains the optimization process.

Once the agent has collected a number of candidate vertices, it chooses that candidate vertex that delivers the highest positive gain with respect to the global criterion (5.2). To this end, the following local gain is computed for all candidates:

$$G_{ij} = g'_{ij} - g_{ij}, \tag{5.12}$$

where $\mathbf{x} \in C_j$ and

$$\begin{aligned} g_{ij} &= |C_i| Var(C_i) + |C_j| Var(C_j) \\ g'_{ij} &= (|C_i| + 1) Var(C_i \cup \{\mathbf{x}\}) + (|C_j| - 1) Var(C_j - \{\mathbf{x}\}) \end{aligned} \tag{5.13}$$

If none of the candidates results in a positive gain, then the expansion trial fails since it does not contribute to the global criterion (5.2). Clearly, this is a *cooperative negotiation scheme* between agent A_i and its neighboring agents.

Since segments must satisfy the connectivity constraint, vertex migrations from one segment to the other that violate this constraint must be recognized[2]. In other words, we have to check whether the candidate vertex $\mathbf{x} \in V(S_j)$ is a *cut vertex* of S_j. In Appendix A, we describe an efficient cut-vertex detection algorithm for this problem.

[2]Clearly, as contiguous vertices always become adjacent, adding a contiguous vertex to a component can never divide it up into two components.

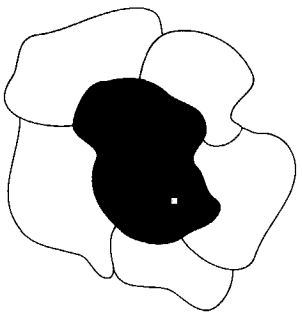


Figure 5.6: Schematic illustration of subject isolation. The agent in the center selects the most deviant subject in case its component is too inhomogeneous.

If the neighboring segment S_j becomes disconnected as a result of the migration (because x was a cut vertex), one or two new agents are *created* that represent the disconnected components.

Subject Isolation: Vertex Removal

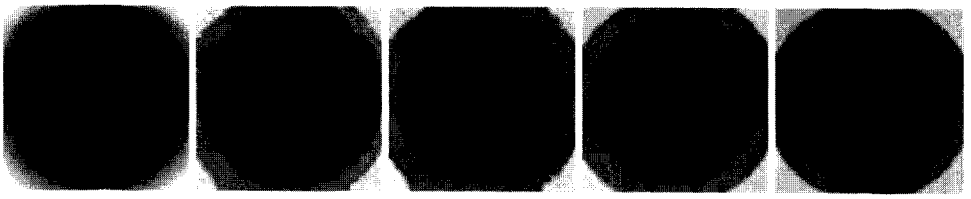
Since the scale constraint was not checked during the subject migration, it can happen that the variance of the agents' component can exceed the given maximum σ^2_{max}. In that case the agent decides to isolate the most deviant subject (see also Fig. 5.6). To this end, the agent first determines the vertex that has the largest (Eucledian) distance to the mean feature of the component. Then, a new agent is created to which this vertex is assigned. This action serves to satisfy the derived property in (5.5).

5.5.3 Termination

For an algorithm to work in practical situations it has to terminate at a certain point. When no agent was able to successfully perform any of its actions for a certain number of epochs, we decide that the algorithm has converged. We have to wait for a number of failed epochs, because of the stochastic border sampling. We use the success counter s_i for this purpose. So the algorithm terminates if $\forall A_i : s_i = 0$. Because the variance of a segment can indeed exceed σ^2_{max} (see (5.5)) local oscillations can occur. To escape from such situations the algorithm also stops when the agent community has evolved for a maximum number of epochs E_{max}.

5.6 Experiments

In order to evaluate the proposed image segmentation method, we performed two types of experiments. First, we explored some specific properties of the method to illustrate how



(a) original (b) 5 epochs (c) 10 epochs (d) 50 epochs (e) 1000 epochs

Figure 5.7: The CCA applied to the Circle image (a) with $\sigma^2_{max} = 20^2$. Figure (b), (c), (d), (e) show the results after 5, 10, 50, 1000 epochs respectively.

it operates under various conditions and in what sense it can be adjusted. After that, we did some experiments in which we compared the CCA performance to other known image segmentation methods. In all experiments, we set the success ratio parameters (5.9) $m = 100$ and $r = 0.01$ and the border sampling parameter $b_i = \lfloor\sqrt{|B_i|}\rfloor$. These values are not very critical so we could fix them for all experiments. If not stated otherwise we set the filter size $f = 1$, which means that we did not pre-filter the images. We show the segmentation results in images, where the color of a segment equals the mean color of the pixels in the segment.

5.6.1 Exploring the CCA

First, we show the course of the segmentation process with the Circle image in Fig. 5.7(a), in which the pixel intensity changes gradually. For this 128 × 128 image we set $\sigma^2_{max} = 20^2$. In a strict sense this image can hardly be segmented, since there are no distinct regions to be separated. However, since gradually changing colors are quite often part of a natural scene, it is interesting to see how segmentation algorithms deal with such transitions.

Fig. 5.7(b) – Fig. 5.7(e) show some intermediate results of the segmentation process. In Fig. 5.7(c) it can be seen that after 10 epochs the final number of segments has been found. The remaining epochs are used to settle the segment borders. As expected, the final result in Fig. 5.7(e) shows a number of clear ring-shaped segments.

Next, we did an experiment to illustrate how to handle noise properly with the CCA segmentation method. We used a 128 × 128 Checkerboard image with squares having either pixel intensity 100 or 160. We added Gaussian noise with $\mu = 0$ and $\sigma^2 = 60^2$ to this image, see Fig. 5.8(a). Fig. 5.8(b) – Fig. 5.8(e) show a number of segmentation results with different pre-filter settings (σ^2_{max} is adjusted accordingly). The σ^2_{max} is set higher than prescribed in (5.8), because in this case the variance in the homogeneous regions increases due to blurring of the edges. The figures clearly show that the pre-filtering results in smoother segment borders.

When we described our model design considerations, we stated that especially in image segmentation it is not desirable to have the number of segments as model parameter. Instead, a scale parameter was introduced to select a maximum allowed variability per segment. In

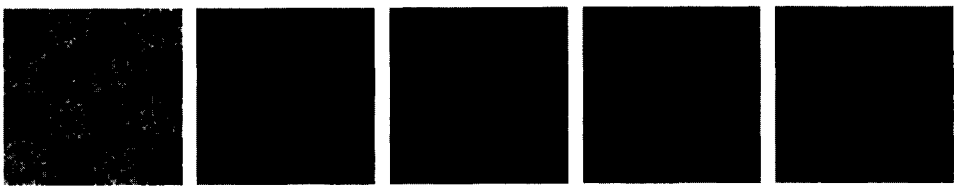


Figure 5.8: Results of applying the CCA segmentation method to the Checkerboard image with various filter size f and corresponding σ^2_{max} settings.

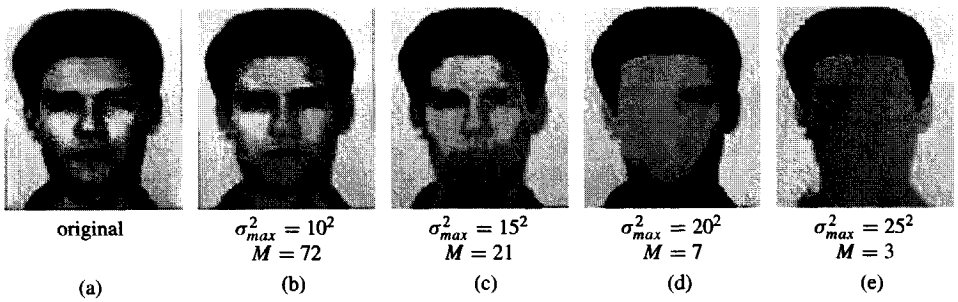


Figure 5.9: CCA segmentation results with different scale parameter settings.

Figure 5.9, we show the results of varying the scale when applying the CCA segmentation method to a face image. Clearly, when the value of σ^2_{max} is increased the smaller details disappear from the resulting segmented images. In Fig. 5.9(b) most facial features are still visible, since their segment variance is below the segment variance constraint. In Fig. 5.9(c) and Fig. 5.9(d) the details gradually disappear, while at the scale in Fig. 5.9(e) only the background, face, and hair can be distinguished.

5.6.2 Comparison with other Methods

In the next experiments, we compared the proposed method to two other typical segmentation methods. The first one is the 'split-and-merge with grouping' algorithm, which has a similar model and parameters. The second one is a Markov random field model-based segmentation algorithm. The latter segmentation method has quite a different model and is often used for segmentation and restoration. Since obtaining the ground truth is difficult in image segmentation, comparing segmentation results is not trivial. Here, we compare the results with respect to the following qualitative criteria: 1) *accurateness*: the segment border represents

a true contour. 2) *continuity*: the segment border may not contain holes if the object that it represents has a continuous contour. 3) *fragmentation*: the union of neighboring segments must be inhomogeneous.

Before discussing the comparative experiments, we first describe the other two segmentation algorithms in more detail.

Split-and-Merge with Grouping Segmentation (SMG)

The split-and-merge implementation we use is based on [30] and [42]. Typically the split-and-merge algorithm uses a scheme in which a square region is recursively split into four quads if the homogeneity predicate is false. Merging is allowed if for neighboring regions the homogeneity predicate is true. The algorithm stops when no splitting or merging is possible. The homogeneity predicate we use is the Chi-squared test, where the rejection probability is set to 0.5. The maximum variance σ^2_{max} used in this test has similar semantics as the maximum variance in (5.4). The specific implementation uses a quad tree structure for the splitting and merging. Since only merges on the same level of the quad tree are allowed, a final grouping step is incorporated. In the grouping stage all regions that together are homogeneous are grouped together. Now, regions are homogeneous if the distance between their means is below a certain maximum δ_μ. Finally, the method usually ends up with many very small regions at the intended segments borders. These regions are grouped together with their neighboring segments if their size is relatively small (default 0.2%) with respect to their largest neighbor. Like the proposed method in this paper, the split-and-merge algorithm does not need to know the number of segments a priori. Moreover, segment connectivity is always satisfied, i.e. the computed segment statistics are always based on connected components. The scale and noise can be regulated by setting σ^2_{max} and δ_μ properly. Clearly, setting σ^2_{max} very low yields many small regions, so the actual segmentation is done in the grouping stage. In this way the algorithm turns into a region-growing algorithm. The consequences of variations in both parameters will become clear in the experiments.

Markov Random Field Model-Based Segmentation (MRF)

In Markov random field models for image segmentation, typically an energy function with two terms is minimized simultaneously for all pixels in the image. It is assumed that the number of color distributions is known together with their mean μ_i and variance σ_i^2. For the natural images, we estimate these values with a mixture of Gaussians method, though we fix $\sigma_i^2 = \sigma^2_{max}$ to make a more fair comparison with the other methods. For the synthetic images we feed the algorithm with the true color distribution values. Then, one of the terms in the energy function accounts for the distance between the pixel color and the color distribution to the pixel color and the other term introduces contextual information through the Markov random field (MRF). That is, the second term imposes the same segment label on pixels in a certain neighborhood (in this case a second-order model). The neighborhood term is weighted with a factor β to stress the relative importance of local connectivity and color matching. Since global optimization of the model is intractable, several approximations have

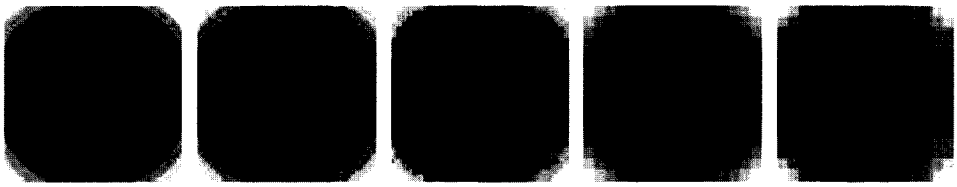


Figure 5.10: Results of applying the SMG algorithm to the Circle image (Fig. 5.7(a)) with various σ^2_{max} and δ_μ parameter settings.

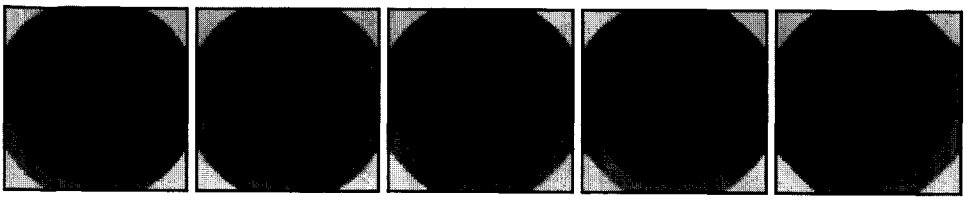
been proposed [7], [14], [22]. We use the well-known ICM method, due to [7], which has been proven to be the most effective method to optimize the MRF-based model [18]. As remarked earlier, as a consequence of the optimization of these pixel criteria, this is rather a quantization than a segmentation method. The scale and noise adaptation is controlled both by the order of the Markov random field (second order) and the number of pre-defined color distributions.

Gradual Changing Color

First, we applied the SMG and MRF methods to the synthetic images that we already used to explore the CCA method in the previous subsection. For the original SMG method the Circle image is especially difficult because the intensity changes gradually. Only if the σ^2_{max} parameter is set to zero the result is similar to the CCA outcome (fig. 5.10(a)). When, on the other hand, σ^2_{max} is set 'properly', the resulting segments become irregular, as can be seen in Fig. 5.10(b) – Fig. 5.10(e). The MRF method has the best results when the contextual information is made unimportant ($\beta = 0.1$) as in Fig. 5.11(a). Then, the segmentation result is mainly based on the image intensities. In this case, where there is no noise, there is strong dependence between image intensity and segment label. Accordingly, the contextual information is indeed superfluous or even harmful, see Fig. 5.11(b) – Fig. 5.11(e).

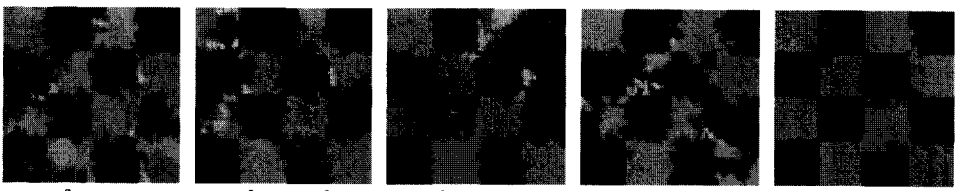
Noisy Image

Next, we applied the algorithms to the noisy Checkerboard image in Fig. 5.8(a). The SMG algorithm clearly has problems with this image. We varied the σ^2_{max} parameter between $\sigma^2_{max} = 0$ and the true variance of the Gaussian noise ($\sigma^2_{max} = 60^2$), and adjusted δ_μ as to achieve the best possible segmentation result. The results achieved with $\sigma^2_{max} = 60^2$ fit quite well to the original noiseless image (Fig. 5.12(e)). We must, however, note that the regular block pattern of the image definitely favors the split-and-merge algorithm. With lower σ^2_{max} values irregular borders occur and, as a consequence of the greedy chaining of small regions, many extra irregular segments occur, see Fig. 5.12(a) – Fig. 5.12(e). Also the MRF method



(a) $\beta = 0.1$ (b) $\beta = 0.2$ (c) $\beta = 0.3$ (d) $\beta = 0.4$ (e) $\beta = 0.5$

Figure 5.11: Results of applying the MRF algorithm to the Circle image (Fig. 5.7(a)) with various β parameter settings.



$\sigma^2_{max} = 0$, $\delta_\mu = 55$ (a)
$\sigma^2_{max} = 10^2$, $\delta_\mu = 55$ (b)
$\sigma^2_{max} = 20^2$, $\delta_\mu = 55$ (c)
$\sigma^2_{max} = 50^2$, $\delta_\mu = 45$ (d)
$\sigma^2_{max} = 60^2$, $\delta_\mu = 25$ (e)

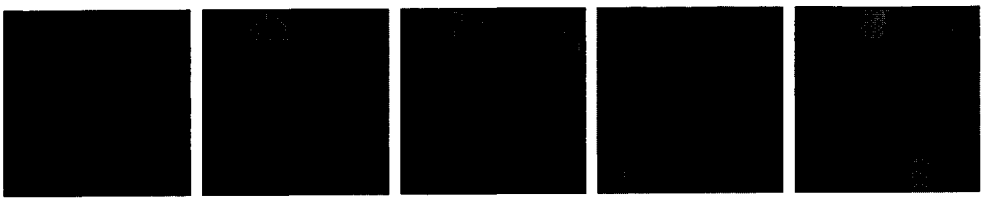
Figure 5.12: Results of applying the split-and-merge segmentation method to the Checkerboard image with various σ^2_{max} and δ_μ settings.

is able to find useful segmentation results for the Checkerboard image. However, when the β parameter is (too) low, the segments are strongly fragmented (connectivity violated) and when β is high, blurring or smearing effects occur, see Fig. 5.13(a) – Fig. 5.13(e).

Natural Images

Finally, we applied all three algorithms to a number of natural images. In the natural images a number of difficult aspects of image segmentation come together, like noise, gradual changing colors, and large and small objects. For all algorithms we chose those parameter settings that turned out to be the best for each of them. The β parameter of the MRF algorithm was fixed to $\beta = 0.3$. We used σ^2_{max} of the SMG equal to σ^2_{max} of the CCA to select a certain scale. Additionally, we set $\sigma^2_{max} = 0$ to turn the SMG into a region-growing algorithm. In all cases the δ_μ parameter of the SMG algorithm is adjusted accordingly.

We started with the Orca image in which the (splashing) water and the sky contain gradually changing colors, see Fig. 5.14(a). Further, there are some well-defined large and small segments on the orca. Like in the Circle image the MRF algorithm creates artefacts in the gradually changing colors, as can be seen in Fig. 5.14(b). Besides, it creates many very small regions around the orca, which is, however, hard to see in the figure. When σ^2_{max} is set to



(a) $\beta = 0.1$ (b) $\beta = 0.2$ (c) $\beta = 0.3$ (d) $\beta = 0.4$ (e) $\beta = 0.5$

Figure 5.13: Results of applying the MRF model-based segmentation method to the Checkerboard image with various β settings and $\mu_1 = 100$, $\mu_2 = 160$, and $\sigma_1^2 = \sigma_2^2 = 60^2$.

the proper scale value, the SMG algorithm has contour artefacts, see Fig. 5.14(c). The SMG with $\sigma_{max}^2 = 0$ makes nice contours, but has some artefacts resulting from the chaining of small regions, like the light-grey areas on the trunk of the Orca and the light-grey region above its fin, see Fig. 5.14(d). The CCA has a good segmentation result, except that (as for the SMG method) some details seem missing, like the small white area on top of the Orca's head. When we measured the corresponding areas it appeared that the variance was indeed below the given σ_{max}^2, hence, the details should indeed not be visible at this scale.

The image from the Flower-Garden sequence (Fig. 5.2) has both very large segments (the tree) as well as many small segments (the flowers). The MRF model-based method shows the expected smearing effects, see Fig. 5.15(a). We had to lower the small-region-removal parameter of the SMG method to 0.01% in order to prevent all flowers from being removed. Besides, with $\sigma_{max}^2 = 15^2$ the same blocking artefacts occurred as before, see Fig. 5.15(b). Configured as region grower ($\sigma_{max}^2 = 0$) the SMG results are better (Fig. 5.15(c)), though in all cases small noisy regions remain. These are not always easy to see in the figures. The CCA method gives segmentation results with both small details and large homogeneous regions.

Finally, we applied the algorithms to a noisy image (Fig. 5.16(a)). The results are similar to those obtained with the Checkerboard image. That is, the MRF has good segmentation results apart from the smearing effects (Fig. 5.16(b)) and the SMG has again blocking artefacts (Fig. 5.16(c) and Fig. 5.16(d)). As expected, the SMG with $\sigma_{max}^2 = 0$ (Fig. 5.16(d)) and the CCA (Fig. 5.16(e)) give similar irregular contours.

5.7 Conclusions

In this paper we presented a cluster model for which the number of clusters does not have to be known a priori. Instead of the number of clusters, we introduced a scale parameter. We justified this choice among others from the image segmentation domain, which we especially addressed. We specialized the cluster model for the image segmentation problem and presented a *cellular coevolutionary algorithm* (CCA) to optimize it in a distributed way. In the experiments, we showed the effectiveness of the method and compared it to two other well-

known segmentation methods. Both the Markov random field model based segmentation method (MRF) and the split-and-merge-and-group (SMG) scheme have quite a number of parameters, some of which we estimated and fixed and others which we varied in the experiments. Though it is hard to draw conclusions without trying all combinations of parameter settings, the proposed CCA is clearly simpler since it has only one essential parameter. The advantage of such a simple model is that it is both easier to tune and to optimize. Having said this, we conclude the following. The MRF method implements segment connectivity as a *soft constraint* and uses a weighting factor β to stress the importance of connectivity. Especially in case of noise and gradual segment transitions, connectivity must be stressed since the pixels are not linked based on intensity only. However, the experiments show that setting β too high results in smearing and blurring artefacts and setting it too low results in fragmented segments, though it has to be noted that the MRF artefacts are not only caused by the MRF model, but also by the (ICM) model optimization scheme. The right trade-off between connectivity and correct border estimation can be hard to find. Though connectivity is a hard constraint for the SMG method, this method also has significant artefacts. First, it strongly needs a small-region elimination step. This is a problem if the image indeed contains small and large segments. Further, if the σ^2_{max} value, which serves to define segment homogeneity, is set properly, then strong blocking effects occur. On the other hand, when it is set very low the SMG method is actually a region-growing algorithm. Then, the results are usually quite good, though borders can be irregular and the grouping procedure has order dependence and chaining artefacts. As regards the SMG method with low σ^2_{max} values, the CCA tends to result in somewhat irregular contours when the image is noisy. However, if an estimate of the amount of noise is known, the proposed method prescribes to pre-filter the image and to adjust σ^2_{max} accordingly, which indeed results in smoother contours. Another problem that the CCA method suffers from is that small regions can be absorbed by large homogeneous regions. This is a consequence of the variance-based scale constraint in the proposed model. The SMG method lacks the same problem in the merging and grouping steps. Other criteria can be developed to prevent this, though it remains a difficult issue.

There is a number of possible extensions to the proposed method. First, the efficiency of the CCA needs improvement, especially if image sequences are to be segmented. The algorithm uses multiple agents that locally optimize a criterion. Accordingly, parallel computer architectures can be exploited very effectively. In image sequences the efficiency can be further improved by starting the evolution with the resulting agent constellations from the previous image. Additionally, it would be very interesting to estimate a texture model for each segment. This is far from trivial since the determination of textured segments conflicts with finding small segments. Another appealing idea is to develop a similar algorithm for the general clustering problem. The main difficulty in that respect is to exploit locality, which is especially easy in the image domain.

(a) original

(b) MRF $\sigma^2_{max} = 15^2, \beta = 0.3$

(c) SMG $\sigma^2_{max} = 15^2, \delta_\mu = 5$

(d) SMG $\sigma^2_{max} = 0, \delta_\mu = 15$

(e) CCA $\sigma^2_{max} = 15^2$

Figure 5.14: (a) shows the Orca image. In (b), (c), (d), and (e) the segmentation results are shown with different algorithms and parameter settings.

(a) MRF $\sigma^2_{max} = 15^2, \beta = 0.3$

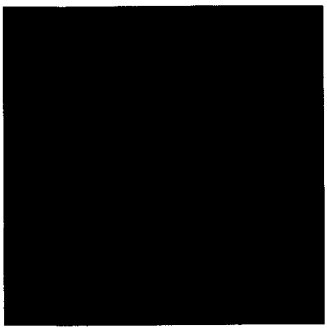
(b) SMG $\sigma^2_{max} = 15^2, \delta_\mu = 20$

(c) SMG $\sigma^2_{max} = 0, \delta_\mu = 15$

(d) CCA $\sigma^2_{max} = 15^2$

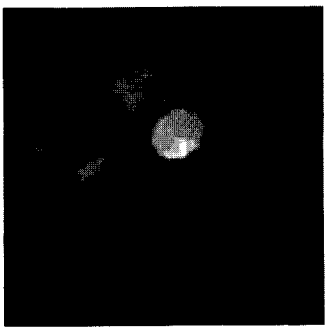
Figure 5.15: (a), (b), (c), and (d) show segmented images resulting from applying different algorithms with several parameter settings to an image from the Flower-Garden sequence.

(a) original



(b) MRF $\sigma^2_{max} = 15^2, \beta = 0.3$

(c) SMG $\sigma^2_{max} = 15^2, \delta_\mu = 10$



(d) SMG $\sigma^2_{max} = 0, \delta_\mu = 15$

(e) CCA $\sigma^2_{max} = 15^2$

Figure 5.16: (a) shows an MRI image with extra artificial noise. In (b), (c), (d), and (e) the segmentation results are shown with different algorithms and parameter settings.

Appendix A

Cut vertex detection algorithm

Detecting whether a segment component becomes disconnected after the deletion of a vertex $\mathbf{x}$ implies detecting whether $\mathbf{x}$ is a cut vertex of a segment component S. The vertex degree of $\mathbf{x} \in V(S)$ is at most three, because otherwise it could not be contiguous to another component. If the vertex degree $d(\mathbf{x}) = 1$, $\mathbf{x}$ cannot be a cut vertex, we are done. For the other cases, we only need to know if there is a path between every pair of adjacent vertices of $\mathbf{x}$ which does not include $\mathbf{x}$ itself. If there is no such path, the graph will be disconnected after the deletion of $\mathbf{x}$, i.e. it is a cut vertex.

First set the visit state of all vertices in $V(S)$ to $state = 0$. Search depth first, starting at an arbitrary adjacent vertex, say $\mathbf{a}_1$, of $\mathbf{x}$ and visit all adjacent vertices of every visited vertex. The searching stops when another (arbitrary) adjacent vertex of $\mathbf{x}$, say $\mathbf{a}_2$, is found or all connected vertices to $\mathbf{a}_1$ are visited. While searching, set the visit state of vertices to $state = 1$ and skip those vertices that have already been visited ($state = 1$). If no path from $\mathbf{a}_1$ to an adjacent vertex of $\mathbf{x}$ is found, then $\mathbf{x}$ is a cut vertex.

Otherwise, if there is one more adjacent vertex, say $\mathbf{a}_3$, start searching from this vertex and continue until a vertex is found with $state = 1$ or until all connected vertices to $\mathbf{a}_3$ are visited. Again mark evaluated vertices while searching, but now set $state = 2$ and skip those vertices that already have $state = 2$. Only if there was no path to a vertex with $state = 1$ then $\mathbf{x}$ is a cut vertex.

Additionally, we speed up the algorithm considerably by using depth-first search with iterative deepening, because usually the path connecting adjacent vertices is very short.

Bibliography

[1] P. Adamidis and V. Petridis. Co-operating populations with different evolution behaviours. In *Proceedings of IEEE International Conference on Evolutionary Computation*, pages 188–191, 1996.

[2] P. Andrey and P. Tarroux. Unsupervised image segmentation using a distributed genetic algorithm. *Pattern Recognition*, 27(5):659–673, 1994.

[3] P. Andrey and P. Tarroux. Unsupervised segmentation of markov random field modeled textured images using selectionist relaxation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(3):252–262, March 1998.

[4] P.J. Angeline and J.B. Pollack. Competitive environments evolve better solutions for complex tasks. In *Proceedings of the Fifth International Conference on Genetic Algorithms*, pages 264–270, 1993.

[5] A. Baraldi and P. Blonda. A survey of fuzzy clustering algorithms for pattern recognition—part I. *IEEE Transactions on Systems, Man, and Cybernetics—Part B*, 29(6):778–785, 1999.

[6] A. Baraldi and P. Blonda. A survey of fuzzy clustering algorithms for pattern recognition—part II. *IEEE Transactions on Systems, Man, and Cybernetics—Part B*, 29(6):786–801, 1999.

[7] J. Besag. On the statistical analysis of dirty pictures. *Journal of the Royal Statistical Society B*, 48:259–302, 1986.

[8] J.C. Bezdek and N.R. Pal. Some new indexes of cluster validity. *IEEE Transactions on Systems, Man, and Cybernetics — Part B*, 28(3):301–315, 1998.

[9] S.M. Bhandarkar and H. Zhang. Image segmentation using evolutionary computation. *IEEE Transactions on Evolutionary Computation*, 3(1):1–21, April 1999.

[10] B. Bhanu, S. Lee, and S. Das. Adaptive image segmentation using genetic and hybrid search methods. *IEEE Transactions on Aerospace and Electronic Systems*, 31(4):1268–1291, October 1995.

[11] D.E. Brown and C.L.Huntley. A practical application of simulated annealing to clustering. *Pattern Recognition*, 25(4):401–412, 1992.

[12] E. Cantú-Paz. *Efficient and Accurate Parallel Genetic Algorithms*. Kluwer Academic Publishers, Boston, 2000.

[13] E. Cantú-Paz. Markov chain models of parallel genetic algorithms. *IEEE Transactions on Evolutionary Computation*, 4(3):216–226, September 2000.

[14] P.B. Chou and C.M. Brown. The theory and practice of bayesian image labeling. *International Journal of Computer Vision*, 4:185–210, 1990.

[15] J.M. Claverie, K.A. De Jong, and A.F. Sheta. Robust nonlinear control design using competitive coevolution. In *Proceedings of the 2000 Congress on Evolutionary Computation*, volume 1, pages 403–409, 2000.

[16] D.L. Davies and D.W. Bouldin. A cluster separation measure. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 1(2):224–227, April 1979.

[17] A. di Nola, V. Loia, and A. Staiano. Genetic-based spatial clustering. In *The Ninth IEEE International Conference on Fuzzy Systems*, pages 953–956, 2000.

[18] R.C. Dubes, A.K. Jain, S.G. Nadabar, and C.C. Chen. MRF model-based algorithms for image segmentation. In *Proceedings IEEE of the 10th International Conference on Pattern Recognition*, volume I, pages 808–814, 1990.

[19] R.O. Duda, P.E. Hart, and D.G. Stork. *Pattern Classification*. John Wiley and Sons, Inc., New York, 2001.

[20] J. C. Dunn. A fuzzy relative of the ISODATA process and its use in detecting compact well-separated clusters. *Journal of Cybernetics*, 3:32–57, 1974.

[21] J. C. Dunn. Well separated clusters and optimal fuzzy partitions. *Journal of Cybernetics*, 4:95–104, 1974.

[22] S. Geman and D. Geman. Stochastic relaxation, gibbs distributions, and the bayesian restoration of images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 6(6):721–741, 1984.

[23] V.S. Gordon and D. Whitley. Serial and parallel genetic algorithms as function optimizers. In *Proceedings 5th International Conference on Genetic Algorithms*, pages 177–183, 1993.

[24] L.O. Hall, I.B. Özyurt, and J.C. Bezdek. Clustering with a genetically optimized approach. *IEEE Transactions on Evolutionary Computation*, 3(2):103–112, july 1999.

[25] H. Handa, N. Baba, and O. Katai. Genetic algorithm involving coevolution mechanism to search for effective genetic information. In *Proceedings of the IEEE International Conference on Evolutionary Computation*, pages 709–714, 1997.

[26] M. Haseyama, M. Kumagai, and H. Kitajima. A genetic algorithm based image segmentation for image analysis. In *Proceedings of the IEEE International Conference of Acoustics, Speech, and Signal Processing*, volume 6, pages 3445–3448, 1999.

[27] W.D. Hillis. Co-evolving parasites improve simulated evolution as an optimization procedure. In C.G. Langton, C. Taylor, J.D. Farmer, and S. Rasmussen, editors, *Artificial Life II*, pages 313–324. Addison-Wesley, 1991.

[28] J.H. Holland. *Adaptation in Natural and Artificial Systems*. University of Michigan Press, 1975.

[29] Y. Horita, T. Murai, and M. Miyahara. Region segmentation using K-mean clustering and genetic algorithms. In *Proceedings IEEE International Conference on Image Processing*, volume 3, pages 1016–1020, 1994.

[30] S.L. Horowitz and T. Pavlidis. Picture segmentation by a tree traversal algorithm. *Journal of the ACM*, 23(2):368–388, 1976.

[31] L.J. Hubert and P. Arabie. Comparing partitions. *Journal of Classification*, 2:193–218, 1985.

[32] A.K. Jain and R.C. Dubes. *Algorithms for Clustering Data*. Prentice-Hall Inc., New Jersey, 1988.

[33] R.W. Klein and R.C. Dubes. Experiments in projection and clustering by simulated annealing. *Pattern Recognition*, 22(2):213–220, 1989.

[34] K. Krishna and M.N. Murty. Genetic K-means algorithm. *IEEE Transactions on Systems, Man, and Cybernetics—Part B*, 29(3):433–439, Jun 1999.

[35] G.N. Lance and W.T. Williams. A general theory of classificatory sorting strategies: II clustering systems. *Computer Journal*, 10:271–277, 1967.

[36] T. Van Le. Evolutionary fuzzy clustering. In *Proceedings IEEE International Conference on Evolutionary Computation*, volume 2, pages 753–758, Killington, Vermont, USA, Oct. 14-16 1995. IEEE Computer Society Press.

[37] J. MacQueen. Some methods for classification and analysis of multivariate observations. In L.M. Le Cam and J. Neyman, editors, *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability*, volume 1, pages 281–297, 1967.

[38] H. Mühlenbein. Parallel genetic algorithms, population genetics and combinatorial optimization. In *Proceedings of the Third International Conference on Genetic Algorithms*, pages 416–421, 1989.

[39] M. Nerome, K. Yamada, S. Endo, and H. Miyagi. Competitive co-evolution based game-strategy acquisition with packaging strategies method. In *IEEE International Conference on Systems, Man, and Cybernetics*, volume 5, pages 4418–4423, 1997.

[40] T.N. Pappas. An adaptive clustering algorithm for image segmentation. *IEEE Transactions on Signal Processing*, 40(4):901–914, April 1992.

[41] E.J. Pauwels and G. Frederix. Finding salient regions in images: Nonparametric clustering for image segmentation and grouping. *Computer Vision and Image Understanding*, 75(1,2):73–85, 1999.

[42] T. Pavlidis. *Structural Pattern Recognition*. Springer Verlag, Berlin Heidelberg New York, 1977.

[43] H. Peng, F. Long, Z. Chi, D.D. Feng, and W. Siu. Hierarchical genetic image segmentation algorithm based on histogram dichotomy. *Electronic Letters*, 36(10):872–874, May 2000.

[44] H. Peng, F. Long, Z. Chi, and W. Siu. A hierarchical distributed genetic algorithm for image segmentation. In *Proceedings of the 2000 Congress on Evolutionary Computation*, volume 1, pages 272–276, 2000.

[45] M.A. Potter and K.A. De Jong. A cooperative coevolutionary approach to function optimization. In *Proceedings of the Third Parallel Problem Solving From Nature*, pages 249–257. Springer-Verlag, 1994.

[46] N. Puppala, S. Sen, and M. Gordin. Shared memory based cooperative coevolution. In *Proceedings of the IEEE International Conference on Evolutionary Computation*, pages 570–574, 1998.

[47] C.D. Rosin and R.K. Belew. New methods for competitive coevolution. *Evolutionary Computation*, 5:1–30, 1997.

[48] J. Rouchier, O. Barreteau, F. Bousquet, and H. Proton. Evolution and co-evolution of individuals and groups in environment. In *Proceedings. International Conference on Multi Agent Systems*, pages 254–260, 1998.

[49] S.Z. Selim and K. Alsultan. A simulated annealing algorithm for the clustering problem. *Pattern recognition*, 24(10):1003–1008, 1991.

[50] S.F. Smith. Flexible learning of problem solving heuristics through adapative search. In *Proceedings of the International Joint Conference on Artificial Intelligence*, pages 422–425, 1983.

[51] S. Theodoridis and K. Koutroumbas. *Pattern Recognition*. Academic Press, London, 1999.

[52] R. Wilson and G.H. Granlund. The uncertainty priciple in image processing. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 6(6):758–767, November 1984.

[53] M. Yu, N. Eua-anant, A. Saudagar, and L. Udpa. Genetic algorithm approach to image segmentation using morphological operations. In *Proceedings of the IEEE International Conference on Image Processing*, pages 775–779, 1998.

[54] L. Zhao, Y. Tsujimura, and M. Gen. Genetic algorithm for fuzzy clustering. In *Proceedings of the IEEE International Conference on Evolutionary Computation*, pages 716–719, 1996.

[55] Q.F. Zhao. A general framework for cooperative co-evolutionary algorithms: A society model. In *Proceedings of the IEEE International Conference on Evolutionary Computation*, pages 57–62, 1998.

Chapter 6

A Maximum Variance Cluster Algorithm

This chapter has been accepted for publication in *IEEE Transactions on Pattern Analysis and Machine Intelligence*, October 2001.

Abstract We present a partitional cluster algorithm that minimizes the sum-of-squared-error criterion while imposing a hard constraint on the cluster variance. Conceptually, hypothesized clusters act in parallel and cooperate with their neighboring clusters in order to minimize the criterion and to satisfy the variance constraint. In order to enable the demarcation of the cluster neighborhood without crucial parameters, we introduce the notion of *foreign* cluster samples. Finally, we demonstrate a new method for cluster tendency assessment based on varying the variance constraint parameter.

Keywords: cluster analysis, partitional clustering, cluster tendency assessment, cluster validity.

6.1 Introduction

Data clustering is an extensively investigated problem for which many algorithms have been reported [18], [26]. Roughly, cluster algorithms can be categorized in hierarchical and partitional algorithms. Hierarchical algorithms deliver a hierarchy of possible clusterings, while partitional cluster algorithms divide the data up into a number of subsets. In partitional cluster analysis most algorithms assume the number of clusters to be known a priori. Because in many cases the number of clusters is not known in advance, additional validation studies are used to find the optimal partitioning of the data [6], [8], [11], [16].

In this paper, we propose an algorithm for partitional clustering that minimizes the within cluster scatter with a constraint on the cluster variance. Accordingly, in contrast to many other cluster algorithms, this method finds the number of clusters automatically. Clearly, a proper value for the variance constraint parameter has to be selected. We present a way to discover cluster tendencies to find significant values for this variance parameter in case this information is not available from the problem domain. We first formally define the cluster problem.

Let $X = \{\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_N\}$ be a data set of $N = |X|$ feature vectors in a p-dimensional metric space. Then, the cluster problem is to find a clustering of X in a set of clusters $C = \{C_1, C_2, ..., C_M\}$, where M is the number of clusters, such that the clusters C_i are homogeneous and the union of clusters is inhomogeneous.

The most widely used criterion to quantify cluster homogeneity is the sum-of-squared-error criterion or simply the square-error criterion[1]:

$$J_e = \frac{\sum_{i=1}^{M} H(C_i)}{N}, \tag{6.1}$$

where

$$H(Y) = \sum_{\mathbf{x} \in Y} \|\mathbf{x} - \mu(Y)\|^2 \tag{6.2}$$

[1] Usually the sum-of-squared-error criterion is not averaged over the whole data set. As defined here, J_e expresses the average distance to the cluster centroids instead of the total distance.

expresses the cluster homogeneity and

$$\mu(Y) = \frac{1}{|Y|} \sum_{\mathbf{x} \in Y} \mathbf{x} \tag{6.3}$$

is the cluster mean.

Straight minimization of (6.1) leads to a trivial clustering with N clusters; one for each sample. Therefore, additional constraints are imperative. For instance, one could fix M, the number of clusters, to an a priori known number like among others the widely used K-means model [23]. In the image segmentation domain, a maximum variance per cluster is sometimes used in addition to a spatial connectivity constraint, e.g. [1], [15]. In this paper, we present an algorithm that is based on a model proposed for intensity-based image segmentation [28]. The constraint that is imposed on the square-error criterion (6.1) within this model states that the variance of the union of two clusters must be higher than a given limit σ^2_{max}, i.e.:

$$\forall C_i, C_j, i \neq j : Var(C_i \cup C_j) \geq \sigma^2_{max}, \text{ where } Var(Y) = \frac{H(Y)}{|Y|} \tag{6.4}$$

A consequence of this model is that the variance of each resulting cluster is generally *below* σ^2_{max} [28]. This does, however, not imply that we could impose a maximum variance constraint on each individual cluster instead. That is, if we would replace the joint variance constraint (6.4) with a constraint for individual clusters ($\forall C_i : Var(C_i) \leq \sigma^2_{max}$) the minimization of (6.1) would lead to a trivial solution with one sample per cluster.

Clearly, since the model imposes a variance constraint instead of fixing the number of clusters, the resulting optimal clustering can be different from the K-means result, even if the final number of clusters is the same.

6.2 Algorithm

For the optimization of the cluster model, we propose a stochastic optimization algorithm. In the literature other stochastic clustering algorithms have been reported that generally optimize the K-means model or fuzzy C-means model either using simulated annealing techniques [7], [20], [25] or using evolutionary computation techniques [10], [13], [22], [29]. Accordingly, these stochastic approaches focus on the optimization of known cluster models. The algorithm we propose, however, shows more resemblance with the distributed genetic algorithm (DGA) for image segmentation as introduced by Andrey and Tarroux [2], [3]. We also dynamically apply local operators to gradually improve a set of hypothesized clusters, but, in contrast with the DGA approach, we consider the statistics of the whole cluster in the optimization process. Before describing the algorithm itself, we first elaborate on the neighborhood relationships of samples, which play a crucial role in the proposed algorithm.

Both for effectiveness and efficiency the algorithm exploits *locality* in the feature space. Namely, the most promising candidates for cluster expansion are in clusters that are close in the feature space. Similarly, the most distant cluster members are the least reliable, hence,

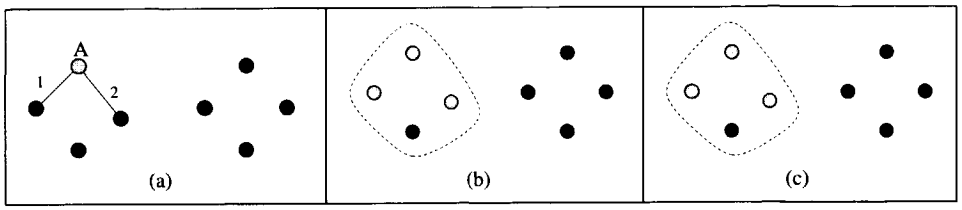


Figure 6.1: Illustration of the cluster neighborhood with the 2 nearest-neighbors method. In (a) the neighbor ranking for sample A is shown. (b) and (c) display the neighborhood of the grey colored cluster with respectively 3 and 4 samples, where in (c) the set of expansion candidates is empty.

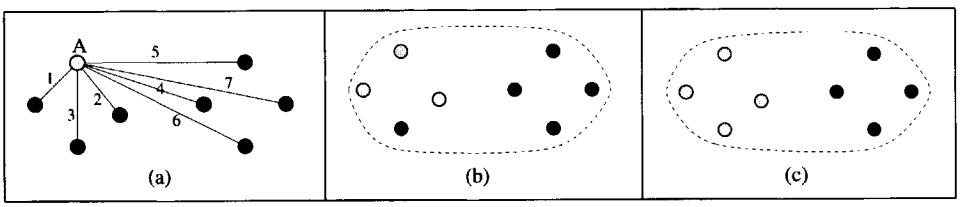


Figure 6.2: Illustration of the cluster neighborhood with the 8 nearest-neighbors method. In (a) the neighbor ranking for sample A is shown. (b) and (c) display the neighborhood of the grey colored cluster with respectively 3 and 4 samples.

they are the first candidates for removal. The computational performance can also profit from feature locality because cluster update operations can be executed in *parallel* when the optimization process applies locally. For these reasons, we consider the optimization process from the individual clusters' point of view, i.e. each cluster can execute a number of actions in order to contribute to an improvement of the criterion as well as to satisfy the variance constraint.

In order to collect the expansion candidates of a cluster, we need to find neighboring samples of that cluster. A common way to define the neighborhood of a sample is to collect its k nearest neighbors using the Eucledian distance measure, where k is a predefined number. In this way, the neighborhood of a cluster would be the union of the neighbors of all samples in the cluster. Accordingly, the set of expansion candidates of a cluster consists of the samples from its neighborhood, excluding the samples from the cluster itself. The problem with this approach is that the value of k becomes an integral part of the cluster model, e.g. [12], [19], [24]. If k is set too low, then even for small clusters all k nearest neighbors are in the cluster itself, so there are no expansion candidates left, see Fig. 6.1. On the other hand, if k is set too high, then the neighborhood is always large, so all clusters have a major part of the samples as expansion candidates, which clearly violates the idea of locality. As a consequence, the set of expansion candidates will be a mix of good and bad candidate samples without preference, see Fig. 6.2.

We take another approach to collect the expansion candidates. First, we call the set of

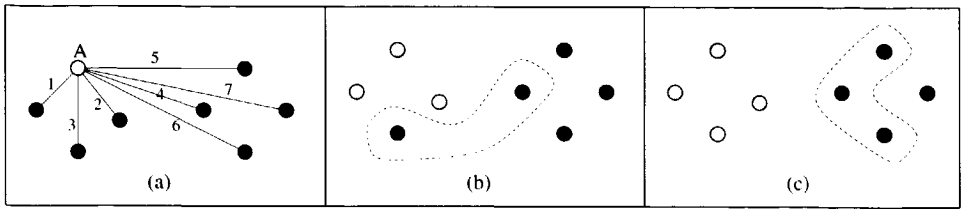


Figure 6.3: The figures illustrate the cluster border construction with respect to sample A and its cluster. In (a) the distance ranking for sample A is shown and (b) and (c) display the second-order border of the grey colored cluster with respectively 3 and 4 samples.

expansion candidates of cluster C_a the *outer border* B_a of cluster C_a. Further, we introduce the notion of *foreign* samples, which we define as neighboring samples that are not in the cluster itself. Accordingly, the k-th order outer border B_a of cluster C_a is the union of the k *nearest foreigners* of all samples in C_a, leading to:

$$B_a(k) = \bigcup_{\mathbf{x} \in C_a} F(\mathbf{x}, k, C_a, X), \tag{6.5}$$

where $F(\mathbf{x}, k, C_a, X)$ is the set of k nearest foreigners of $\mathbf{x}$ according to:

$$F(\mathbf{x}, k, C_a, Y) = \begin{cases} \{nf(\mathbf{x}, C_a, Y)\} \cup F(\mathbf{x}, k-1, C_a, Y - \{nf(\mathbf{x}, C_a, Y)\}), & \text{if } k > 0 \\ \emptyset, & \text{if } k = 0 \end{cases} \tag{6.6}$$

and $nf(\mathbf{x}, C_a, Y)$ is the nearest foreigner of sample $\mathbf{x} \in C_a$ in X defined as:

$$nf(\mathbf{x}, C_a, Y) = \arg \min_{\mathbf{y} \in Y - C_a} \|\mathbf{y} - \mathbf{x}\|^2 \tag{6.7}$$

Consequently, the outer border of a cluster always has a limited number of samples and it never becomes empty (unless there is only one cluster left). In Fig. 6.3 we illustrate how a second-order outer border evolves with the growing of a cluster. An appropriate value for the order of the outer border depends on the constellation of the clusters and the actual data.

Besides the expansion candidates, we also need to collect candidates for removal from the cluster in order to impose the variance constraint. To this end, we introduce the q-th order inner border I_a of cluster C_a. The inner border I_a consists of those samples that are the furthest cluster mates of the samples in C_a. Accordingly, the q-th order inner border can be expressed as follows:

$$I_a(q) = \bigcup_{\mathbf{x} \in C_a} G(\mathbf{x}, q, C_a), \tag{6.8}$$

where $G(\mathbf{x}, q, C_a)$ is the set of q furthest cluster mates of $\mathbf{x}$, or, in other words the q furthest neighbors of $\mathbf{x}$ in C_a according to:

$$G(\mathbf{x}, q, Y) = \begin{cases} \{fn(\mathbf{x}, Y)\} \cup G(\mathbf{x}, q-1, Y - \{fn(\mathbf{x}, Y)\}), & \text{if } q > 0 \\ \emptyset, & \text{if } q = 0 \end{cases} \tag{6.9}$$

and $fn(\mathbf{x}, Y)$ is the furthest neighbor of $\mathbf{x}$ in Y as in:

$$fn(\mathbf{x}, Y) = \arg\max_{\mathbf{y} \in Y} \|\mathbf{y} - \mathbf{x}\|^2 \tag{6.10}$$

Since the set of foreigners of a sample changes every time the cluster is updated, for efficiency reasons we introduce a rank list R_i per sample $\mathbf{x}_i$, containing indices to all other samples in X in order of their distance to the given sample. The rank list R_i is an N-tuple defined as $R_i = Rank(\mathbf{x}_i, X)$ according to:

$$Rank(\mathbf{x}, Y) = \langle nn(\mathbf{x}, Y) \circ Rank(\mathbf{x}, Y - \{nn(\mathbf{x}, Y)\}) \rangle, \tag{6.11}$$

where $\circ$ is the concatenate operator and $nn(\mathbf{x}, Y)$ is the nearest neighbor of $\mathbf{x}$ in Y as in:

$$nn(\mathbf{x}, Y) = \arg\min_{\mathbf{y} \in Y} \|\mathbf{y} - \mathbf{x}\|^2 \tag{6.12}$$

The k nearest foreigners of a cluster sample can now easily be found by scanning the rank list, starting at the head while skipping those elements that are already in the cluster. To this end, some bookkeeping is needed for both the clusters and the samples.

After the definitions of the inner and outer border of a cluster, we now describe the maximum variance cluster (MVC) algorithm. Since the optimization of the model defined in (6.1) – (6.4) is certainly an intractable problem, exhaustive search of all alternatives is out of the question. In order to prevent early convergence of the consequent approximate optimization process, we introduce sources of non-determinism in the algorithm [4].

The algorithm starts with as many clusters as samples. Then in a sequence of epochs every cluster has the possibility to update its content. Conceptually, in each epoch the clusters act in parallel or alternatively sequentially in random order. During the update process, cluster C_a performs a series of tests each of which causes a different update action for that cluster.

1. *Isolation*
   First C_a checks whether its variance exceeds the predefined maximum σ^2_{max}. If so, it randomly takes a number of candidates i_a from its inner border I_a proportional to the total number of samples in I_a. It *isolates* the candidate that is the furthest from the cluster mean $\mu(C_a)$. It takes a restricted number of candidates to control the greed of this operation. Then the isolated sample forms a new cluster (resulting in an increase of the number of clusters).

2. *Union*
   If on the other hand, C_a is homogeneous (its variance is below σ^2_{max}), then it checks if it can *unite* with a *neighboring cluster*, where a neighboring cluster is a cluster that contains a foreign sample of C_a. To this end, it computes the joint variance with its neighbors. If the lowest joint variance remains under σ^2_{max}, then the corresponding neighbor merges with C_a (resulting in a decrease of the number of clusters).

3. *Perturbation*
   Finally, if none of the other actions applies, the cluster C_a attempts to *improve the criterion* by randomly collecting a number of candidates b_a from its outer border B_a. Again, to control the greed, a restricted number of candidates is selected proportional to the size of the border. Then C_a ranks these candidates with respect to the gain in the square-error criterion when moving them from the neighboring cluster C_b to C_a.

   We define the criterion gain between C_a and C_b with respect to $\mathbf{x} \in C_b$ as:

$$G_{ab} = H(C_a) + H(C_b) - H(C_a \cup \{\mathbf{x}\}) - H(C_b - \{\mathbf{x}\}) \tag{6.13}$$

   If the best candidate has a positive gain then this candidate moves from the neighbor to C_a. Otherwise, there is a small probability P_d of *occasional defect*, which forces the best candidate to move to C_a irrespective the criterion contribution.

Because of the occasional defect, no true convergence of the algorithm exists. Therefore, after a certain number of epochs E_{max}, we set $P_d = 0$. Further, since it is possible that at the minimum of the constrained optimization problem (6.1) – (6.4) the variance of some clusters exceeds σ^2_{max} (exceptions to the general rule mentioned in Section 6.1), after E_{max} epochs also isolation is no longer allowed in order to prevent algorithm oscillations. With these precautions the algorithm will certainly converge, since the overall homogeneity criterion only decreases and it is always greater than or equal to zero. In case two clusters unite, the criterion may increase, but the number of clusters is finite. Still, we have to wait for a number of epochs in which the clusters have not changed due to the stochastic sampling of the border.

6.3 Experiments

In this section we demonstrate the effectiveness of the proposed maximum variance cluster (MVC) algorithm with some artificial and real data sets. First, however, we show that the maximum variance constraint parameter can be used for cluster tendency assessment.

Clearly, since the clustering result depends on the setting of σ^2_{max}, the square-error criterion J_e also changes as a function of σ^2_{max}. Accordingly, cluster tendencies can be read from trends in J_e. Consider for instance the data set shown Fig. 6.4(a). The corresponding square-error curve resulting from varying σ^2_{max} can be seen in Fig. 6.4(b). The figure shows some prominent plateaus in the square-error criterion. Clearly, these plateaus can both be caused

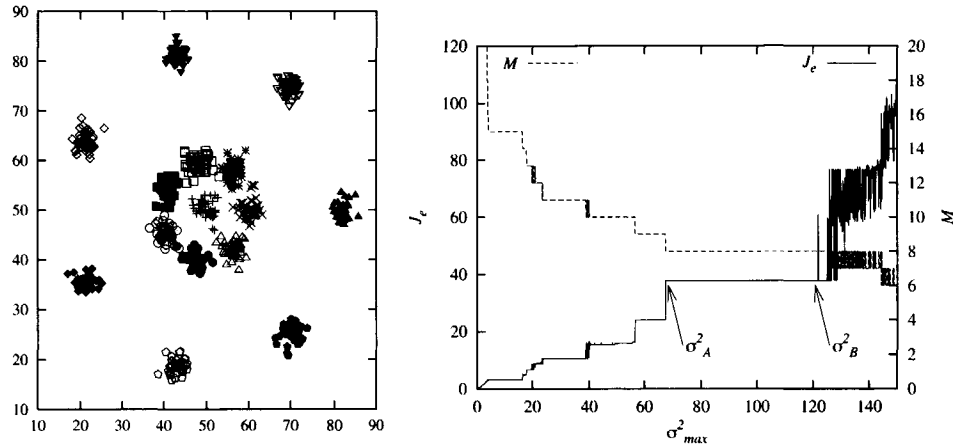


Figure 6.4: In (a) the R15 data set is shown, which is generated as 15 similar 2-D Gaussian distributions. In (b) J_e and M are displayed as a function of σ^2_{max} for the R15 data set.

by hierarchical cluster structures and random sample patterns. If there is real structure in the data then the variance constraint can be increased up to the moment that true clusters are lumped together. On the other hand, if the resulting clustering is random in character, then the clusters will easily be rearranged when the variance constraint is increased.

Let σ^2_A be the starting point of a J_e plateau and σ^2_B the end point of that plateau, as for example in Fig. 6.4(b). Then, we define the strength S of a plateau as the ratio:

$$S(\sigma^2_A, \sigma^2_B) = \frac{\sigma^2_B}{\sigma^2_A} \tag{6.14}$$

Accordingly, the strength of a plateau gives an indication of whether or not the corresponding clustering represents real structure in the data. Intuitively, we expect that two real clusters will be lumped together if the variance constraint is higher than roughly twice their individual variance. This implies that the strength of a plateau in J_e should be greater than 2 in order to be a *significant plateau*, that is, to represent real structure. To test this hypothesis, we did experiments with uniform random data and various numbers of samples. For each data set, we measured the maximum plateau strength S_{max} and subsequently computed the distribution of S_{max}. In Fig. 6.5 we show the cumulative distribution of S_{max} for different sizes of the random data sets. Only when N was very low ($N < 50$), significant plateaus were occasionally found, which is to be expected with low numbers of samples. On the other hand, the experiments with structured data, among which the ones that we describe in this section, indeed resulted in significant J_e plateaus.

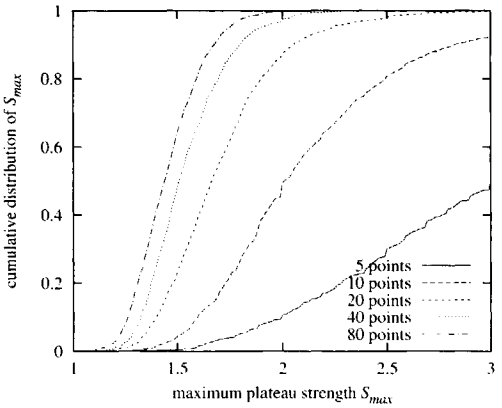


Figure 6.5: Cumulative distribution of the maximum plateau strength for differently sized random 2-D data sets. The distributions have been calculated from 1000 independent draws.

The advantage of this cluster tendency assessment approach is that the same model is used for the clustering as for the cluster tendency detection. In our view the usual approach, where different criteria are used for the clustering and the detection of the cluster tendency, is undesirable, like for instance in [6], [8], [11], [16]. That is, the cluster algorithm may not be able to find the clustering corresponding to the local minimum or knee in the cluster tendency function[2].

Some additional remarks need to be made about the significance of J_e plateaus. First, it has to be noted that because of fractal like effects, σ_A^2 must be higher than a certain value in order to rule out extremely small 'significant' plateaus. Second, in case there are *multiple* significant plateaus, these plateaus represent the scales at which the data can be considered. Then, the user can select the appropriate scale and corresponding plateau. In our view, there is no best scale in these cases, so the selection is fully subjective.

In all experiments, we compared the performance of the MVC algorithm to the K-means algorithm [23], and the Gaussian mixtures modeling (GMM) method with likelihood maximisation [18] using the EM algorithm [9]. For both the K-means and the GMM method the numbers of clusters is set to the resulting number (M) found by the MVC algorithm. Further, since both the MVC and the K-means algorithm prefer circular shaped clusters we constrained the Gaussian models of the GMM to be circular too in order to reduce its number of parameters. For the MVC algorithm we set $P_d = 0.001$, $E_{max} = 100$, $k = 3$, $q = 1$, $i_a = \lfloor\sqrt{|I_a|}\rfloor$, and $b_a = \lfloor\sqrt{|B_a|}\rfloor$. It has to be noted that these parameter values appeared to be not critical (experiments not included). They merely serve to tune the convergence behavior similar as in other non-deterministic optimization algorithms, like for instance the mutation rate and population size parameters in genetic algorithms [14]. Since all algorithms

[2]When additional criteria are used to discover cluster tendencies, they are usually called cluster validity functions or indices.

method	parameter	hit	M	time (ms)
MVC	$\sigma^2_{max} = 10$	100	15	122
MVC	$\sigma^2_{max} = 100$	100	8	99
K-means	$M = 15$	3	–	20
K-means	$M = 8$	10	–	12
GMM	$M = 15$	4	–	280
GMM	$M = 8$	10	–	160

Table 6.1: Statistical results of applying the algorithms to the R15 data set.

have non-deterministic components[3], we ran them 100 times on each data set and display the result with the lowest square-error (MVC and K-means) or highest likelihood (GMM), i.e. the best solution found. Further, we measured the average computation time, and the number of times the best solution was found (hit rate). For the MVC algorithm we did not add the computation time of the rank lists, since this time only depends on the size of the data set and not on the structure. Moreover, these lists have to be computed only once for a data set and can then be used for tendency assessment and the subsequent runs to find the optimal clustering.

We start with the already mentioned data set from Fig. 6.4(a) consisting of 15 similar 2-D Gaussian clusters that are positioned in rings (R15). Though we know an estimate of the variance of the clusters, we first varied the σ^2_{max} constraint in order to discover the cluster tendency. Fig. 6.4(b) shows the resulting curves for J_e and the number of found clusters M. The figure shows a number of prominent plateaus in J_e, from which the first [4.20...16.5] has strength $S = 3.93$. This significant plateau corresponds to the originating structure of 15 clusters. Further, there is a large plateau [67.5...122] with strength $S = 1.80$ which corresponds to the clustering where all inner clusters are merged into one cluster. This plateau is, however, not significant according to our definition. This is because the total variance of the clusters lumped in the center is much higher than the variance of the outer clusters. The resulting clusterings for MVC ($\sigma^2_{max} = 10$), K-means ($M = 15$) and GMM ($M = 15$) were the same (Fig. 6.6), and also for MVC ($\sigma^2_{max} = 100$), K-means ($M = 8$) and GMM ($M = 8$). Table 6.1 shows that the MVC algorithm is clearly more robust in converging towards the (possibly local) minimum of its criterion. That is, the hit rate for the MVC algorithm is much higher than for the K-means and the GMM algorithm. Further, the K-means algorithm that is known to be efficient is indeed the fastest.

The next artificial data set consists of three clusters with some additional outliers (O3), see Fig. 6.7(a). Again, we first varied the σ^2_{max} parameter for the MVC algorithm in order to discover cluster tendencies. Although we roughly know the variance of the clusters, in this case it is certainly useful to search for the proper σ^2_{max} value, since the outliers may disrupt the original cluster variances.

Fig. 6.7(b) clearly shows only one prominent plateau [22.0...82.0]. This plateau is significant, because its strength is $S = 3.73$. In the corresponding clustering result all three

[3]The K-means and GMM algorithm are initialized with randomly chosen cluster models.

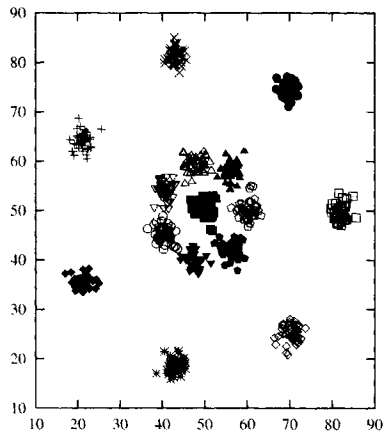
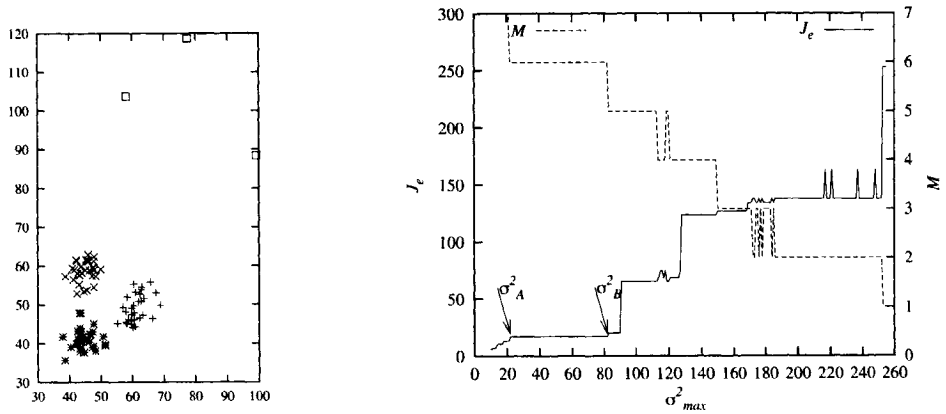


Figure 6.6: Results of applying the clustering algorithms to the R15 data set. In (a) the results of the MVC and K-means algorithm with 15 clusters is shown and in (b) the results of the GMM method with 15 clusters is shown.

outliers are put in separate clusters leading to a total of six clusters, as is shown in Fig. 6.8(a). Because the K-means algorithm does not impose a variance constraint it could find a lower square-error minimum and corresponding clustering with $M = 6$ than the MVC as can be seen in Fig. 6.8(b). The algorithm split one cluster instead of putting the outliers in separate clusters. This supports the statement that using one model for the detection of cluster tendencies and another for the clustering is undesirable. Also the GMM algorithm was not able to find the MVC solution (see Fig. 6.8(b)), though the MVC solution indeed had a higher likelihood. When $M = 3$, the K-means and the GMM algorithm merged two true clusters and put the outliers in one clusters. Table 6.2 shows the statistics of this experiment. Again, the K-means and GMM algorithm were clearly less robust in finding their respective (local) criterion optimum than the MVC and the K-means was the fastest.

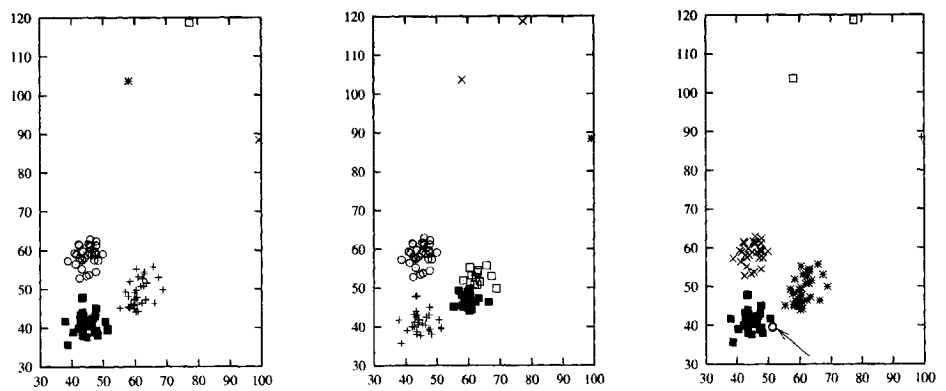
We repeated this experiment several times with different generated clusters and outliers. The results were generally the same as described above, i.e. if there was a difference between the cluster results of the algorithms, the MVC handled the outliers better by putting them in separate clusters or it converged more often to its criterion optimum.

For the last synthetic experiment, we used a larger data set (D31) consisting of 31 randomly placed 2-D Gaussian clusters of 100 samples each, see Fig. 6.9(a). The tendency curve resulting from varying σ^2_{max} for the MVC algorithm shows one significant plateau [0.0030...0.0062] ($S = 2.07$), which corresponds to the original 31 clusters. Remarkably, the K-means and the GMM algorithm were not able to find the originating cluster structure, not even after 10000 trials. The statistical results in Table 6.3 show that the MVC algorithm consistently found the real structure, while the difference in computation time between the algorithms becomes small.



(a) (b)

Figure 6.7: In (a) the O3 data set is shown which is generated as 3 similar 2-D Gaussian distributions with some additional outliers. In (b) J_e and M are displayed as a function of the σ^2_{max} constraint parameter.

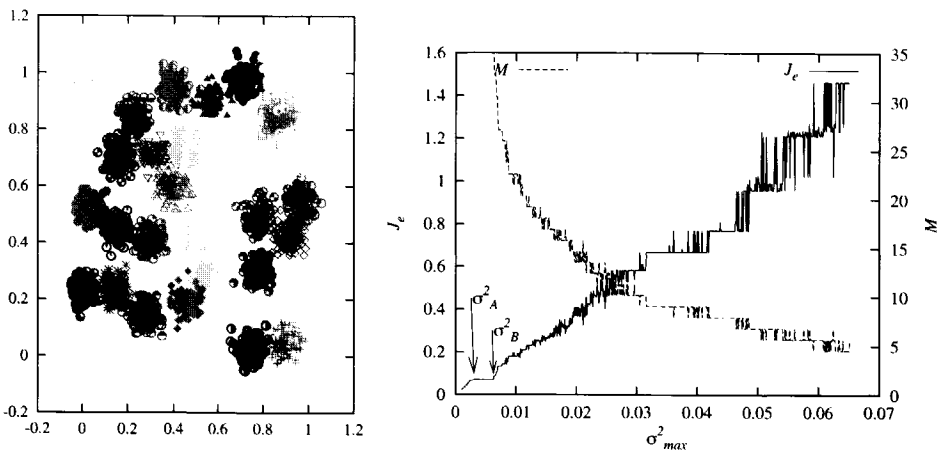


(a) $\sigma^2_{max} = 50$ (b) $M = 6$ (c) $M = 6$

Figure 6.8: Results of applying the clustering algorithms to the O3 data set. In (a) the results of the MVC algorithm are shown resulting in 6 clusters. (b) and (c) show the results of the K-means and GMM algorithm, respectively. The GMM puts a remote cluster sample in a separate cluster.

method	parameter	hit	M	time (ms)
MVC	$\sigma^2_{max} = 50$	94	6	27
K-means	$M = 6$	1	–	2.5
K-means	$M = 3$	14	–	1.2
GMM	$M = 6$	1	–	17
GMM	$M = 3$	10	–	4.3

Table 6.2: Statistical results of applying the algorithms to the O3 data set. The hit rate of the GMM method with $M = 6$ certainly refers to a local maximum.

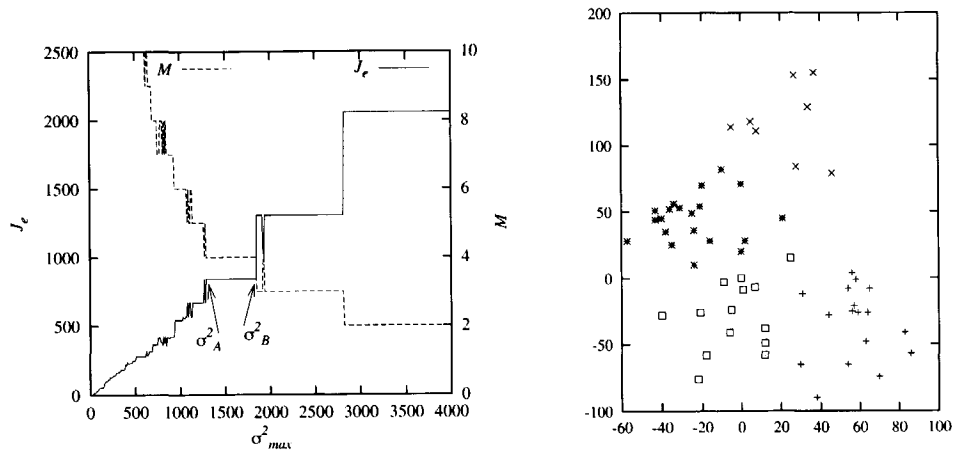


(a) (b)

Figure 6.9: In (a) the D31 data set is shown which is generated as 31 similar 2-D Gaussian distributions. In (b) J_e and M are displayed as a function of the σ^2_{max} constraint parameter.

method	parameter	hit	M	time (ms)
MVC	$\sigma^2_{max} = 0.004$	100	31	930
K-means	$M = 31$	1	–	390
GMM	$M = 31$	1	–	220

Table 6.3: Statistical results of applying the algorithms to the D31 data set. The K-means and GMM algorithm were not able to find the originating structure, so the hit rate refers to a local optimum.



(a) (b) $\sigma^2_{max} = 1500$, $M = 4$

Figure 6.10: (a) shows J_e and M as a function of the σ^2_{max} constraint parameter for the *German Towns* data set. In (b) the clustering result of the MVC and the K-means algorithm with 4 clusters is displayed.

method	parameter	hit	M	time (ms)
MVC	$\sigma^2_{max} = 1500$	100	4	27
K-means	$M = 4$	29	–	0.56
GMM	$M = 4$	1	–	12

Table 6.4: Statistical results of applying the algorithms to the *German Towns* data set.

Next, we applied the algorithms to some real data sets. We started with the *German Towns* data set which consists of 2-D coordinates of 59 German towns (pre-'Wende' situation). In order to find a significant clustering result, we again varied the σ^2_{max} parameter for the MVC algorithm. The resulting curves of J_e and M are displayed in Fig. 6.10(a). The two plateaus [1290...1840] and [1940...2810] have strengths $S = 1.43$ and $S = 1.45$ respectively. Although both plateaus are not significant, we show the clustering results of the first plateau with 4 clusters in Fig. 6.10(b), which equals the result of the K-means algorithm with $M = 4$. The GMM algorithm came up with a different solution consisting of three main clusters and one cluster containing a single sample. When we visually inspect the data in Fig. 6.10(b), we can conclude that it is certainly arguable if this data set contains significant structure. Table 6.4 shows similar hit rates as before and the K-means algorithm was again the fastest.

Finally, we processed the well-known *Iris* data set with both algorithms. The *Iris* data set is actually a labeled data set consisting of three classes of irises each characterized by four features. Fig. 6.11 illustrates the cluster tendencies resulting from varying σ^2_{max} for the MVC

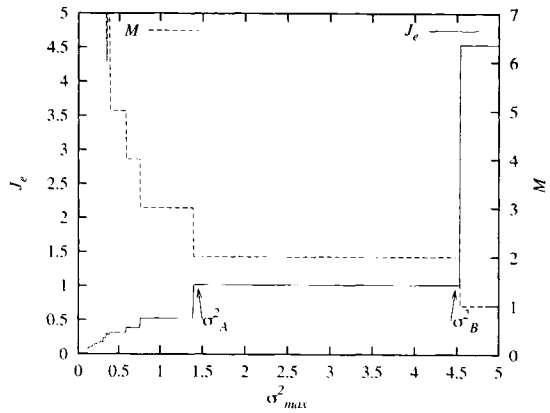


Figure 6.11: J_e and M as a function of the σ^2_{max} constraint parameter for the *Iris* data set.

method	parameter	hit	M	time (ms)
MVC	$\sigma^2_{max} = 1.0$	100	3	43
MVC	$\sigma^2_{max} = 2.0$	100	2	25
K-means	$M = 3$	36	–	1.6
K-means	$M = 2$	99	–	0.83
GMM	$M = 3$	8	–	4.2
GMM	$M = 2$	99	–	1.6

Table 6.5: Statistical results of applying the algorithms to the *Iris* data set.

algorithm. The figure displays several plateaus, from which [0.76...1.39] and [1.40...4.53] are the strongest. The plateaus with strengths $S = 1.83$ and $S = 3.24$ correspond to three and two clusters, respectively. Hence, only the latter is significant. All three algorithms found similar results for the same number of clusters. Since it is known that the three classes cannot be separated based on the given features, it is not surprising that the clustering with $M = 3$ does not correspond to the given labels. However, from the clustering with $M = 2$ (corresponding to the significant plateau), one cluster almost perfectly matches the samples of class I and the other cluster matches the samples of class II+III of the Iris class labels. The statistics in Table 6.5 show similar differences between the MVC, K-means, and GMM algorithm as in the other experiments.

6.4 Discussion

We presented a maximum variance cluster algorithm (MVC) for partitional clustering. In contrast to many other algorithms, the MVC algorithm uses a *maximum variance constraint* instead of the number of clusters as parameter. In the experiments, we showed that the method

is effective in finding a proper clustering and we compared its results to those of the widely used K-means algorithm and the Gaussian mixtures modeling (GMM) method with likelihood maximisation with the EM algorithm. In contrast to the proposed MVC method both the K-means and the GMM method need the number of clusters to be known a priori.

We showed that the MVC method copes better with *outliers* than the K-means algorithm. The GMM method is in principle able to separate the outliers, but has problems with the optimization process leading to convergence into local criterion optima. The MVC algorithm is more robust in finding the optimum of its criterion than both the K-means and GMM algorithm. We must note that other and better optimization schemes for both the K-means model (e.g. [5], [21]) and the Gaussian mixtures modeling (e.g. [17], [27]) have been developed. However, the improved optimization of these algorithms is achieved at the cost of (considerable) additional computation time or algorithm complexity.

The MVC algorithm is up to 100 times slower than the very efficient K-means algorithm, especially for small data sets and a low number of clusters. This is partially caused by the fact that we did not adjust the maximum number of epochs parameter E_{max} to the size of the data set. For larger data sets with a higher number of clusters the differences in computation time between both algorithms almost disappear. An advantage of the MVC algorithm with respect to computational efficiency is that it can be implemented on *parallel and distributed computer architectures* relatively easily. Accordingly, for large data sets the MVC algorithm may be advantageous also for efficiency reasons. In such a distributed computing environment, clusters can be maintained by separate processes. Then, only clusters that are neighbors communicate with each other. The main point of consideration will be how to balance the cluster processes on the available computers when clusters merge and when samples are isolated into new clusters.

An interesting property of the proposed method is that it enables the *assessment of cluster tendencies*. Generally, the curve resulting from varying the maximum variance constraint parameter as a function of the square-error displays some prominent plateaus that reveal the structure of the data. We indicated a way to find *significant* structure in the data by rating the strength of the plateaus. Accordingly, we were able to find proper settings of the maximum variance constraint parameter, which is the only model parameter.

A drawback of the MVC algorithm may be that it uses a distance rank list for every sample. The size of this rank list grows proportional to the square of the number of samples, so the amount of storage needed can become substantially. The main problem, however, lies in the computation of these rank lists. Since these lists are sorted, their construction costs $O(Nlog(N))$ operations. In order to prevent the rank list from becoming a bottleneck for the application of the MVC algorithm, a maximum distance constraint d_{max} can be imposed in addition to the maximum cluster variance constraint, e.g. $d_{max} = 2\sigma_{max}$. Then only those samples need to be ranked that are within the d_{max} range of the reference sample.

Bibliography

[1] R. Adams and L. Bishop. Seeded region growing. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 16(6):641–647, 1994.

[2] P. Andrey and P. Tarroux. Unsupervised image segmentation using a distributed genetic algorithm. *Pattern Recognition*, 27(5):659–673, 1994.

[3] P. Andrey and P. Tarroux. Unsupervised segmentation of markov random field modeled textured images using selectionist relaxation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(3):252–262, March 1998.

[4] P.J. Angeline and J.B. Pollack. Competitive environments evolve better solutions for complex tasks. In *Proceedings of the Fifth International Conference on Genetic Algorithms*, pages 264–270, 1993.

[5] G.P. Babu, N. Murty, and S.S. Keerthi. A stochastic connectionist approach for global optimization with application to pattern clustering. *IEEE Transactions on Systems, Man, and Cybernetics—Part B*, 30(1):10–24, 2000.

[6] J.C. Bezdek and N.R. Pal. Some new indexes of cluster validity. *IEEE Transactions on Systems, Man, and Cybernetics — Part B*, 28(3):301–315, 1998.

[7] D.E. Brown and C.L.Huntley. A practical application of simulated annealing to clustering. *Pattern Recognition*, 25(4):401–412, 1992.

[8] D.L. Davies and D.W. Bouldin. A cluster separation measure. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 1(2):224–227, April 1979.

[9] A.P. Dempster, N.M. Laird, and D.B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society B*, 39:1–38, 1977.

[10] A. di Nola, V. Loia, and A. Staiano. Genetic-based spatial clustering. In *The Ninth IEEE International Conference on Fuzzy Systems*, pages 953–956, 2000.

[11] J. C. Dunn. Well separated clusters and optimal fuzzy partitions. *Journal of Cybernetics*, 4:95–104, 1974.

[12] K.C. Gowda and G. Krishna. Agglomerative clustering using the concept of mutual nearest neighborhood. *Pattern Recognition*, 10:105–112, 1978.

[13] L.O. Hall, I.B. Özyurt, and J.C. Bezdek. Clustering with a genetically optimized approach. *IEEE Transactions on Evolutionary Computation*, 3(2):103–112, july 1999.

[14] J.H. Holland. *Adaptation in Natural and Artificial Systems*. University of Michigan Press, 1975.

[15] S.L. Horowitz and T. Pavlidis. Picture segmentation by a tree traversal algorithm. *Journal of the ACM*, 23(2):368–388, 1976.

[16] L.J. Hubert and P. Arabie. Comparing partitions. *Journal of Classification*, 2:193–218, 1985.

[17] S. Ingrassia. A comparison between simulated annealing and the EM algorithms in normal mixtures decompositions. *Statistics and Computing*, 2:203–211, 1992.

[18] A.K. Jain and R.C. Dubes. *Algorithms for Clustering Data*. Prentice-Hall Inc., New Jersey, 1988.

[19] R.A. Jarvis and E.A. Patrick. Clustering using a similarity measure based on shared near neighbors. *IEEE Transactions on Computers*, 22:1025–1034, 1973.

[20] R.W. Klein and R.C. Dubes. Experiments in projection and clustering by simulated annealing. *Pattern Recognition*, 22(2):213–220, 1989.

[21] K. Krishna and M.N. Murty. Genetic K-means algorithm. *IEEE Transactions on Systems, Man, and Cybernetics—Part B*, 29(3):433–439, Jun 1999.

[22] T. Van Le. Evolutionary fuzzy clustering. In *Proceedings IEEE International Conference on Evolutionary Computation*, volume 2, pages 753–758, Killington, Vermont, USA, Oct. 14-16 1995. IEEE Computer Society Press.

[23] J. MacQueen. Some methods for classification and analysis of multivariate observations. In L.M. Le Cam and J. Neyman, editors, *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability*, volume 1, pages 281–297, 1967.

[24] E.J. Pauwels and G. Frederix. Finding salient regions in images: Nonparametric clustering for image segmentation and grouping. *Computer Vision and Image Understanding*, 75(1,2):73–85, 1999.

[25] S.Z. Selim and K. Alsultan. A simulated annealing algorithm for the clustering problem. *Pattern recognition*, 24(10):1003–1008, 1991.

[26] S. Theodoridis and K. Koutroumbas. *Pattern Recognition*. Academic Press, London, 1999.

[27] D.M. Titterington, A.F.M. Smith, and U.E. Makov. *Statistical Analysis of Finite Mixture Distributions*. John Wiley and Sons, New York, 1990.

[28] C.J. Veenman, M.J.T. Reinders, and E. Backer. A cellular coevolutionary algorithm for image segmentation. *Submitted to IEEE Transactions on Image Processing*, 2001.

[29] L. Zhao, Y. Tsujimura, and M. Gen. Genetic algorithm for fuzzy clustering. In *Proceedings of the IEEE International Conference on Evolutionary Computation*, pages 716–719, 1996.

Summary

In this thesis we address three different computer vision and machine learning problems, being the motion correspondence problem, the image segmentation problem, and the clustering problem. In a number of chapters organized as journal papers, models and corresponding optimization schemes are proposed to solve the respective problems. In the introductory chapter we attempt to find unifying aspects in the properties of the problems and in the way the problems have been attacked.

Chapter 2 studies the motion correspondence problem, for which a diversity of qualitative and statistical solutions exist. We concentrate on qualitative modeling especially in situations where assignment conflicts arise, either because multiple features compete for one detected point or because multiple detected points fit a single feature point. We leave out the possibility of separate point track initiation and termination, because that principally conflicts with allowing for temporary point occlusion. We introduce individual, combined, and global motion models and fit existing qualitative solutions in this framework. Additionally, we present a new efficient tracking algorithm that satisfies these — possibly constrained — models in a greedy matching sense, including an effective way to handle detection errors and occlusion. The performance evaluation shows that the proposed algorithm outperforms existing greedy matching algorithms. Finally, we describe an extension to the tracker that enables automatic simultaneous initialization of the point tracks. Several experiments show that the extended algorithm is efficient, hardly sensitive to its few parameters, and qualitatively better than other algorithms, including the presumed optimal statistical multiple hypothesis tracker.

In Chapter 3, we enhance the greedy optimization scheme introduced in Chapter 2. We propose to defer the correspondence decisions to achieve performance improvement, that is, we examine whether the consequences of a number of candidate correspondences lead to alternate and better solutions. The consequent problem we are faced with is a multi-dimensional assignment problem, which is known to be NP-hard. To restrict the consequent increase in computation time, the candidate solutions are suitably ordered and additional combined motion constraints are imposed. Experiments show the appropriateness of the proposed extension, both with respect to performance and to computational aspects.

In Chapter 4, we adjust the proposed motion correspondence model such that the number of tracked feature points may vary in time. Accordingly, the feature points may enter and leave the video scene. Since this feature principally conflicts with the feature of coping with temporary occlusion, additional measures are needed. Firstly, we add an optimization criterion that aims to minimize the number of established point tacks. Secondly, we add two continuity constraints that disqualify point tracks with more than a_{max} consecutive missing

measurements, and point tracks with less than p_{min} consecutive measurements. The corresponding optimization algorithm we present is a sequential heuristic search algorithm. A comparison between the algorithm and other tracking algorithms shows that the proposed algorithm is easy to tune and generally more efficient and more accurate.

In Chapter 5, we study the clustering and the related image segmentation or spatial clustering problem. These problems are inherently difficult, both with respect to the definition of adequate models as well as to the optimization of the models. We present a model for the clustering problem where the number of clusters does not need to be known a priori. Instead it minimizes the sum-of-squared-error while imposing a hard constraint on the cluster variance. The absence of the number of clusters as parameter in the model is among others useful in the image segmentation domain, for which we specialize the model. Further, we propose a cellular coevolutionary algorithm for the optimization of the model in the image segmentation domain. Within this scheme multiple agents are placed in a regular 2-D grid representing the image, which imposes neighboring relations on them. The agents cooperatively consider pixel migration from one to the other in order to improve the homogeneity of the ensemble of the image regions they represent. If the union of the regions of neighboring agents is homogeneous then the agents form alliances. On the other hand, if an agent discovers a deviant subject, it isolates the subject. In the experiments we show the effectiveness of the proposed method and compare it to other segmentation algorithms. The efficiency can be easily improved by exploiting the intrinsic parallelism of the proposed method.

In Chapter 6, we present a partitional cluster algorithm that fits the cluster model proposed in Chapter 5. Conceptually, hypothesized clusters act in parallel and cooperate with their neighboring clusters in order to minimize the sum-of-squared-error criterion and to satisfy the variance constraint similar to the algorithm in Chapter 5. In order to enable the demarcation of the cluster neighborhood without crucial parameters, we introduce the notion of foreign cluster samples. Finally, we demonstrate a new method of cluster tendency assessment based on the variation of the variance constraint parameter.

In the introductory chapter, we attempt to find unifying aspects of the motion correspondence problem, the image segmentation problem, and the clustering problem. Since there are no examples given that show how the structure in the data is established, these problems are all unsupervised pattern recognition problems. We posed these problems as optimization problems that typically contain a number of conflicting generalization and specialization criteria. Both straight optimization of generalization criteria and of specialization criteria leads to trivial solutions.

Because unsupervised pattern recognition problems inherently contain conflicting criteria, we focus on multiple criterion related modeling and optimization aspects in the introductory chapter. It follows that as such these problems are fundamentally undecidable, since a large set of mathematically equivalent optimal solutions exists.

Summary of the thesis: *Motion Correspondence, Image Segmentation, and Clustering: Modeling and Optimization Aspects".*

C.J. Veenman, Delft, November 2001.

Samenvatting

In deze thesis behandelen we drie verschillende computer vision en machine learning problemen, te weten het bewegingscorrespondentieprobleem, het beeldsegmentatieprobleem, en het clusterprobleem. In een vijftal hoofdstukken, geschreven als tijdschrift artikelen, worden modellen met bijbehorende optimaliseringsmethoden voor deze problemen gepresenteerd. In het inleidende hoofdstuk bekijken we wat de genoemde problemen en oplossingsmethoden gemeen hebben.

Om te beginnen bestuderen we in Hoofdstuk 2 het bewegingscorrespondentieprobleem, waarvoor een verscheidenheid aan kwalitatieve en statistische oplossingsmethoden bestaat. Wij concentreren ons op de kwalitatieve modellering, waarbij we met name kijken naar situaties waar toekenningsconflicten ontstaan, ofwel veroorzaakt doordat verschillende kenmerkpunten passen op één gemeten punt, ofwel andersom. We laten afzonderlijke puntspoorinitialisering en -beëindiging buiten beschouwing, omdat dit principiëel conflicteert met het modelleren van tijdelijk niet gedetecteerde kenmerkpunten. We introduceren individuele, gecombineerde en globale bewegingsmodellen en passen bestaande methodieken in dit raamwerk. Verder presenteren we een nieuw efficiënt puntvolgalgoritme dat deze modellen op een 'greedy' manier optimaliseert (eventueel met additionele constraints) inclusief een effectieve manier om bestand te zijn tegen detectiefouten en occlusie. Uit de evaluatie blijkt dat dit algoritme beter is dan bestaande 'greedy' methodieken. Tenslotte beschrijven we een uitbreiding van het algoritme dat automatische, gelijktijdige initialisatie van de puntsporen mogelijk maakt. Verscheidene experimenten tonen aan dat dit algoritme efficiënt is en nauwlijks gevoelig voor variaties in zijn parameters. Bovendien zijn de resultaten kwalitatief beter dan die van de andere algoritmen inclusief de verondersteld optimale 'multiple hypothesis tracker.'

In Hoofdstuk 3 breiden we het in Hoofdstuk 2 geïntroduceerde 'greedy' optimaliseringsalgoritme uit. We presenteren een methode waarbij de puntcorrespondentiebeslissingen worden uitgesteld om zo de uiteindelijke resultaten te verbeteren. Dat wil zeggen, we testen of de consequenties van een aantal kandidaatcorrespondenties leiden to andere en betere puntsporen. Het probleem waar we dientengevolge mee geconfronteerd worden is een multidimensioneel toekenningsprobleem, een bekend NP-compleet probleem. Om de zoekruimte te beperken, worden de kandidaatoplossingen geordend en additionele constraints opgelegd. Experimenten tonen middels verbeterde prestaties aan dat de voorgestelde uitbreiding inderdaad een verbetering is.

In Hoofdstuk 4 passen we het bewegingscorrespondentiemodel verder aan zodat het aan-

tal te volgen punten in de tijd mag variëren. Omdat deze optie principiëel conflicteert met het omgaan met tijdelijk missende puntmetingen, zijn extra aanpassingen nodig. Ten eerste voegen we een optimaliseringscriterium toe dat ernaar streeft het aantal vastgestelde puntsporen te minimaliseren. Ten tweede voegen we twee continuïteitscriteria toe om puntsporen met meer dan a_{max} achtereenvolgende missende puntmetingen of minder dan p_{min} achtereenvolgende puntmetingen te diskwalificeren. Het bijbehorende optimaliseringsalgoritme dat we presenteren is een sequentiëel heuristisch zoekalgoritme. Bij het vergelijken van dit algoritme met andere bekende algoritmen blijkt het voorgestelde algoritme eenvoudig in te stellen en is het in het algemeen efficiënter en nauwkeuriger.

In Hoofdstuk 5 bestuderen we het cluster- en verwante beeldsegmentatieprobleem ofwel het spatiëel clusterprobleem. Deze problemen zijn inherent moeilijk, zowel met betrekking tot het definiëren van adequate modellen als het optimaliseren van de modellen. We presenteren een model voor het clusterprobleem waarvoor het aantal clusters niet a priori bekend hoeft te zijn. In plaats daarvan streeft het model naar minimalisering van de som der kwadratische fouten ten opzichte van de clustercentra, terwijl een harde constraint op de clustervariantie wordt opgelegd. De afwezigheid van het aantal clusters als parameter van het model is onder meer van belang in het beeldsegmentatiedomein, waarvoor we het model specialiseren. Verder beschrijven we een cellulair coëvolutionair algoritme voor de optimalisering van het model in het beeldsegmentatie domein. In dit schema wordt een aantal agenten in een regelmatig 2-D grid geplaatst. Dit grid, dat het beeld representeert, definiëert onder andere buurrelaties tussen de agenten. Vervolgens overwegen de agenten pixelmigratie van de ene agent naar de ander om zo de homogeniteit van de beeldgebieden die ze representeren te verhogen. Als de beeldgebieden van naburige agenten gezamenlijk homogeen zijn vormen de agenten allianties. Als aan de andere kant een agent een afwijkend pixel signaleert, zal hij deze isoleren. In de experimenten tonen we de effectiviteit van de voorgestelde methode aan en vergelijken we hem met andere beeldsegmentatiealgoritmen. De efficiëntie kan eenvoudig verhoogd worden door het intrinsieke parallellisme van de voorgestelde methode uit te buiten.

In Hoofdstuk 6 presenteren we een partitioneel clusteralgoritme gebaseerd op het clustermodel dat we in Hoofdstuk 5 hebben beschreven. Conceptueel werken alle clusters samen met hun buurclusters om de kwadratische fout te minimaliseren en te voldoen aan de variantie constraint, vergelijkbaar met het algoritme in Hoofdstuk 5. Om de omgeving van een cluster vast te stellen zonder cruciale parameters introduceren we het begrip 'foreign' clusterelement. Tenslotte stellen we een nieuwe cluster tendens methode voor die gebaseerd is op het variëren van de variantie constraint parameter.

In het inleidende hoofdstuk verkennen we aspecten die het bewegingscorrespondentieprobleem, het beeldsegmentatieprobleem en het clusterprobleem gemeen hebben. Omdat er geen voorbeelden gegeven zijn omtrent hoe de structuur zich in de data manifesteert, kunnen al deze problemen als 'unsupervised' patroonherkenningsproblemen beschouwd worden. In deze thesis hebben we deze problemen als optimaliseringsproblemen gedefiniëerd, die onvermijdelijk een aantal conflicterende generalisatie- en specialisatiecriteria bevatten. Echter, zowel directe optimalisering van generalisatiecriteria als specialisatiecriteria leidt tot triviale oplossingen.

Omdat 'unsupervised' patroonherkenningsproblemen inherent conflicterende criteria bevatten, richten we ons in het inleidende hoofdstuk op multiple criteria gerelateerde modellerings- en optimaliseringsaspecten. Het blijkt dat deze problemen als zodanig onbeslisbaar zijn, omdat een groot aantal optimale, mathematisch equivalente, oplossingen bestaat.

Samenvatting van het proefschrift: *Bewegingscorrespondentie, Beeldsegmentatie, en Clustering: Modellerings- en Optimaliseringsaspecten".*

C.J. Veenman, Delft, November 2001.

Curriculum Vitae

Cornelis Johannes (Cor) Veenman was born in Ankeveen, on February 21, 1964. He obtained his VWO diploma at the Alberdingk Thijm College in Hilversum in 1982. Then, he started studying at the HTS in Hilversum and he received his B.Eng. degree in electrical engineering in 1987. For his graduation project he designed and implemented a software system that monitors electrical and pressure signals from patients in the operating room. In 1987 he started to study computer science at the Vrije Universiteit, Amsterdam. For his Master's project he investigated the genetic programming method and developed a novel representation scheme. He received his M.Sc. degree in 1996. Between 1991 and 1997 he worked as a software developer in multi-media applications and taught programming courses at a software company.

In 1997, he started as a research assistant at the Information and Communication Theory Group, Department of Mediamatics, at the Delft University of Technology. Initially he worked on the combination of facial feature detectors in a multi-agent framework in order to analyze facial expressions. Later he focused on combinatorial and multi-criterion optimization issues in feature point tracking, image segmentation, and clustering.

Since July 2001 Cor Veenman works as a researcher of Bioinformatics at the Information and Communication Theory Group, Department of Mediamatics, at the Delft University of Technology. He investigates dynamical gene expression behavior with pattern recognition and optimization techniques.

Acknowledgments

Scientific research is something you cannot do alone. You need people around with similar interests and motivation to obtain knowledge, to stimulate creativity, to test ideas, to share the joy of success, and to put failures into perspective. Luckily there have been many people during the period of my Ph.D. research supporting me in that sense. To start with I thank my supervisor Marcel Reinders with whom I have had many heated discussions and who gave me the freedom to explore our research field in all its aspects. Further I thank my Professor Eric Backer for his valuable opinion and unlimited enthusiasm.

Especially in the beginning of my research I experienced that several experts when asked for help in becoming familiar with their fields were very willing to help me. I like to mention Peter Lucas and Cees Witteveen for discussions about uncertainty calculi, Prof. Cees Roos for discussions about linear optimization theory, and Prof. Geert-Jan Olsder and Theo Driessen with whom I discussed game theory issues. When I started publishing I discovered that even those persons engaged in so many things can be prompt and helpful. Especially I would like to thank Prof. Ishwar Sethi for proofreading and giving submission advice.

Part of my Ph.D. research took place within the STW FASE project. The project was a cooperation between the Delft University of Technology and the National Research Institute for Mathematics and Computer Science (CWI). I thank our partners at CWI for the many constructive and at times critical meetings. I also want to thank Erik Mouw for helping me with numerous experiments with both hardware and software issues.

Brainstorming and scientific discussion are among my favourite activities, as some of the ICT-group members must have noticed. I especially want to thank Lodewijk Wessels, Eugene van Someren, and former group member John Schavemaker for being sparring-partners in this respect. Further I thank Alan Hanjalic for making our office more than just a place to work in and all members of the ICT-group for creating a pleasant working environment.

Finally I thank Leontien for witnessing my struggles during these years and forgiving me my absent-minded moments and I thank my parents for giving me the drive and the optimism that is indispensable, not in the least to doing research.

isbn 90-9015459-0