A Nonconvex Cost Optimization Approach to Tracking Multiple Targets by a Parallel Computational Network

Kenneth Rose, Eitan Gurewitz and Geoffrey Fox

Caltech Concurrent Computation Program
California Institute of Technology
Mail Stop 206-49
Pasadena, CA 91125

Abstract

The problem of tracking multiple targets in the presence of displacement noise and clutter is formulated as a nonconvex optimization problem. The form of the suggested cost function is shown to be suitable for the Graduated Non-Convexity algorithm, which can be viewed as deterministic annealing. The method is first derived for the two-dimensional (spatial/ temporal) case, and then generalized to the multi-dimensional case. The complexity grows linearly with the number of targets. Computer simulations show the performance with crossing trajectories.

Introduction

The problem of tracking is that of estimating the trajectories of moving (point) objects given a set of noisy measurements in time. Many approaches have been suggested for tracking, some of which will be briefly mentioned here so as to clarify the relationship between them and our new method.

Two types of noise are assumed present, namely, displacement noise and clutter. The displacement noise corresponds to errors in the location of returns with respect to the actual locations of targets. Clutter consists of noisy points which do not relate to an existing target. If only displacement noise were present, then the problem would reduce to that of curve fitting to minimize some appropriate measure. In particular, if the target dynamics could be modelled by state space equations driven by Gaussian white noise, then the Kalman filter recursive solution could be used to minimize the mean squared error criterion.

The presence of clutter, however, adds a *data association* aspect to the problem, i.e. which of the observed returns corresponds to the target. The Probabilistic Data Association method [2] overcomes this difficulty by considering only the most likely associations and assigning an association probability to each hypothesis. The method outputs as state estimate the corresponding average of the conditional state estimates. Another difficulty arises when dealing with multiple targets. Unlike clutter, the presence of another target produces points with a structured distribution. Thus in the case of crossing targets, these points may be assigned high association probabilities and mislead the estimator. This gave rise to the Joint Probabilistic Data Association method [3], which assigns joint association probabilities to sets of hypotheses. The complexity of this method clearly grows combinatorially. A neural network method for approximating the joint association probabilities has recently been proposed [4].

An interesting approach to tracking is by using Dynamic Programming [5]. Here the space is discretized, and a full search through all possible states is efficiently performed by exploiting special properties of the problem. The advantage of the method is that for a single target, it will always find the optimal trajectory (within the resolution of the grid). On the other hand, the ability to resolve crossing targets is determined by the resolution since two trajectories passing through the same state will be merged by the search procedure. Refining the resolution clearly affects the complexity.

Hough Transform methods have also been suggested for tracking [6]. The Hough Transform detects trajectories belonging to a specified family of parametrized curves, by a voting procedure. It is relatively insensitive to clutter, but quite sensitive to displacement noise. Much work has been devoted to reduce the complexity of the multi-dimensional Hough Transform. It can naturally be used as a track initiator for another tracking method, by detecting possible trajectories within small windows in the raw data.

0-8186-2113-3/90/0000/0078$01.00 © 1990 IEEE

In this paper a new approach is proposed. First, the tracking problem is reformulated as a non-convex optimization problem, i.e., the minimization of an appropriate energy function. The form of this energy function is then shown to be suitable for an algorithm based on the principle of Graduated Non-Convexity (GNC) which was proposed for visual reconstruction [1]. We propose a deterministic algorithm which enables avoiding local minima. In fact the energy function is replaced by a sequence of energy functions which converges to the original energy function. The sequence starts with a convex energy function and gradually introduces non-convexity as it approaches the final energy function.

The method is first developed for the two-dimensional (space/time) case, and then generalized to deal with the n-dimensional case. Simulation results are shown to demonstrate the performance. Finally, issues of possible parallel implementation, notably in terms of cellular automata, are discussed.

The two dimensional (time/space) derivation

As stated in the introduction, the problem is made hard by its data association aspect, i.e., which point is associated with which target. In fact, if we knew the correct data association we could easily compute the optimal trajectory since the energy function would be convex. Moreover, the analytic solution could be given in terms of Green functions. The approach in this study will be to implicitly look for the set of points to associate with a target so as to minimize the energy. From such a viewpoint, if one considers all possible trajectories for a target, one should compute its energy after assigning to it the nearest returns.

The proposed energy function for two-dimensional (spatial-temporal) data is

$$E = \sum_i \min_j \{(u_i - d_i^{(j)})^2\} + \mu \sum_i (\ddot{u}_i)^2 + \alpha \sum_i \left(\frac{u_i - \hat{u}_i}{\sigma_i}\right)^2 , \quad (1)$$

where u_i is the trajectory location at time i, $d_i^{(j)}$ is the j'th data point at time i, and $\hat{u}$ is some prediction of the trajectory given past data or other external information such as other sensors etc., which may be nonuniformly weighted in time (σ_i). The first term of the energy function measures the trajectory's distance from the observed data. The second term penalizes non-smooth trajectories. The third term takes into account predictions and allows adding external information.

This energy function has many local minima because of the first term which is not convex, and indeed, the first term contains the data association problem. Reconsider the first term,

$$E_1 = \sum_i g_i(u_i), \quad (2)$$

where

$$g_i(x) = \min_j \{(x - d_i^{(j)})^2\}. \quad (3)$$

We wish to find a *convex* approximation E^* to the energy function, and we shall do it by replacing the functions g_i by some g_i^*. The condition for convexity is that the Hessian be positive definite. The Hessian of E^* is given by

$$\begin{aligned} H_{ij}(u) &= \frac{\partial^2 E^*}{\partial u_i \partial u_j} \\ &= \frac{d^2 g_i^*}{dx^2}(u_i)\delta_{ij} + 2\mu(Q^2)_{ij} + \frac{2\alpha}{\sigma_i^2}\delta_{ij} \end{aligned} \quad (4)$$

where δ_{ij} is the Kronecker delta and Q is the matrix given by

$$Q_{ij} = \begin{cases} 2, & \text{if } i = j; \\ -1, & \text{if } |i - j| = 1; \\ 0, & \text{otherwise.} \end{cases} \quad (5)$$

Since the matrix Q^2 is positive semidefinite, then by requiring the diagonal matrix representing the first and third terms in (4) to be positive definite we ensure that so is H and therefore E^* is convex. Hence we require

$$\frac{d^2 g_i^*}{dx^2} \geq -\frac{2\alpha}{\sigma_i^2} = -c_i. \quad (6)$$

The functions g_i are piecewise parabolic as illustrated in Figure 1. The best approximating functions (from below) g_i^* which satisfy (6) are obtained by fitting inverted parabolas to the boundaries as shown in Figure 1. These functions are differentiable and their derivative is continuous. Between two detected points, $2d$ apart, we get (assuming the origin is at the midpoint)

$$g_i^* = \begin{cases} \frac{c}{1+c}d^2 - cx^2, & \text{if } |x| < \frac{d}{1+c}; \\ (d - x)^2, & \text{otherwise.} \end{cases} \quad (7)$$

Note that for $c = c_i$ we get the convex approximation we needed, while on the other hand for $c \to \infty$ we get $g_i^* \to g_i$ and therefore $E^* \to E$.

One may therefore choose c as a natural parameter for gradually introducing non-convexity into the energy function. This is not the only possibility, another choice of parameter which is closely related to multi-scale methods is currently under investigation.

The algorithm will therefore be in the following lines. Initialize $c = c_i$ so that the energy function is convex, and optimize using your favorite method (e.g. gradient descent). At each iteration increase c to introduce some non-convexity and re-optimize. An important issue is that of where to stop the iterations. Recall that

$$g_i^*(x) \leq g_i(x), \qquad \forall x \tag{8}$$

which implies that if the configuration u^* globally minimizes E^*, then

$$E^*(u^*) \leq E(u), \qquad \forall u. \tag{9}$$

Hence, u^* is the global minimum of E if and only if

$$E(u^*) = E^*(u^*). \tag{10}$$

In certain cases it turns out that the convex approximation is already a good enough approximation of the energy function (this depends on the choice of parameters) so that (10) holds and the global minimum is found. Moreover, if by choosing a careful schedule for updating c we can ensure that we are always at the global minimum of E^*, then whenever we reach a configuration which satisfies (10), we have found the global minimum of E. Note that each of the intervals over which $g_i \neq g_i^*$ is made smaller as c is increased.

Generalization to the n-dimensional space

The generalization will be given for the n-dimensional spatial case and illustrated for the two dimensional spatial case. The main issue here is to produce a convex approximation to the energy function. Once we have that, we shall immediately see how to introduce non-convexity.

Again let us consider the first term of the energy function. It is a set of paraboloids centered at the data points. Over a two-dimensional space, the energy function looks like an irregular egg tray. The boundaries within which each paraboloid is defined are given by the appropriate Voronoi diagram. This is a set of hyperplanes which encloses with each data point all the points in space which are nearest to this data point. In order to obtain the convex approximation we smooth the function over these boundaries to satisfy the second derivative requirements. Similarly to the one-dimensional case (7), the function is modified within a sleeve around the boundary hyperplanes.

The form of the approximating function at a given point will depend on the number of data points associated with it. For the case of two dimensional space, a point is associated with two data points if it is in a sleeve, and with three data points if it is in the intersection of two sleeves. In general each point may be associated with up to $n+1$ data points (excluding pathologies). Now suppose that we are in a zone that is associated with $k+1$ data points. These points are all in a k-dimensional subspace. Moreover, assuming they are "generally positioned", i.e., no (k-1)-dimensional subspace contains all of them, then they are on some k-dimensional *hypersphere* (which will be simply referred to as sphere).

The approximating function is defined as an inverted paraboloid over the k-dimensional space, centered at the center of the bounding sphere, and an upright paraboloid in the remaining orthogonal directions.

$$g_i^*(x_1, ..., x_n) = K - c\sum_{j=1}^{k} x_j^2 + \sum_{j=k+1}^{n} x_j^2 \tag{11}$$

where K is a constant to be determined, $k+1$ is the number of data points associated with $(x_1, ..., x_n)$. These data points are in fact the vertices of a hyperpolyhedron in the k-dimensional space. The intersection of the corresponding sleeves is a smaller polyhedron congruent to it whose bounding sphere has the same center (see Fig. 2 for the two-dimensional case). Let R be the radius of this hypersphere, then

$$K = cR^2 + g_i(v) \tag{12}$$

where v will stand for any of the vertices of the sleeve intersection. Note that g_i has the same value for all these vertices (equally distant from data points). Note also that for the one-dimensional spatial case we obtain (7) from (11) and (12) by substituting $R = d/(1+c)$ which is indeed the radius of the one-dimensional bounding sphere, i.e. half the distance.

We shall omit the details here but it is not difficult to show that the approximating functions g_i^* are continuous, differentiable and their

derivative is continuous everywhere. Such an approximating function is shown in Fig. 3.

We have generalized our convex approximation to multi-dimensional spaces, and the resulting energy can still be naturally parametrized by c to introduce non-convexity.

On parallel implementation of the algorithm

In the previous sections we have constructed the sequence of energy functions which starts with a convex approximation and converges to the original energy function. However, the actual computation in the algorithm does not involve evaluating these functions everywhere. All that is required at each iteration is to evaluate the derivative with respect to each variable at the current point. As the g_i^* functions are defined by cases, the main problem is to establish the case, i.e., with how many and which data points it is associated. By geometrical considerations, it can be shown that given the current point and the nearest data point, all that is required is to search a certain window for additional data points. The window is defined as the difference of two hyperballs $B-b$, where b is a ball centered at the current point and whose radius is the distance to the nearest data point

$$r = |x - d^{(j)}|. \tag{13}$$

The larger ball B is the interior of a sphere passing through the data point, whose center is on the line connecting the two points, and whose radius is

$$R = \frac{1+c}{c} r. \tag{14}$$

This is illustrated in Fig. 4. The data points found in the crescent $B - b$ will determine the form of g_i^*.

Let us reconsider the energy function given in (1). As there is no interaction between tracks, the complexity grows linearly with the number of tracks. In fact, all trajectories can be computed in parallel. We shall next discuss possible parallelization of the computation of a single trajectory. The second observation to make is that trajectories are temporally but not spatially discretized.

The fact that the trajectories are not discretized in space allows avoiding apriori limitations on resolving crossing targets. The discretization of space into a large number of mutually exclusive states is typical for neural-network formulations and dynamic programming methods.

On the other hand, the discretization in time which is assumed to be property of the input, enables a parallel implementation. This can be done by a network of processors, each in charge of a given time slice. The only inter-processor communication is within small neighborhoods, through the smoothness term of the energy function which contains a temporal derivative. It is therefore natural to visualize such a system in terms of cellular automata. For a given time window, each cell processes one time slice data while incorporating into the processing the output of its defined neighbors.

In order to eliminate the need to transfer input data between processors as the time window slides, a cyclic index rotation is used. The processors are connected in a circle, and the connection is severed between the last and the first time slices in the window. As the window slides by one time unit, all indices are rotated so that each processor still deals with the same input data, but advances within the window. The processor which was last now becomes first and receives fresh input. Note that the disconnected branch is also rotated to be between the current last and first time slice in the window.

Simulation

A simulated example is shown in Fig. 5. There is one spatial dimension and one temporal dimension. Five crossing targets are detected in the presence of clutter and displacement noise. The targets were generated by specifying initial positions and velocities, and applying small acceleration noise to them at each time unit.

Summary

A nonconvex cost optimization approach is suggested for multitarget tracking in the presence of displacement noise and clutter. The method is based on deriving a convex approximation to the energy function and gradually introducing nonconvexity. By this procedure we start with the global minimum of the approximated energy function, and perform "tracking" of the global minimum while varying the nonconvexity parameter. In this respect the method can be viewed as deterministic annealing. The convex approximation was derived for the two-dimensional (spatial/temporal) case and then generalized for multi-dimensional cases. The

computations can be performed in parallel per track and per time slice. A simulated example is presented to demonstrate the performance of the method.

References

[1] Blake, A., and A. Zisserman, *Visual Reconstruction*. Cambridge, MA: MIT Press, 1987.

[2] Bar-Shalom, Y., and E. Tse, "Tracking in a cluttered environment with probabilistic data association," Automatica, Vol. 11, pp. 451-460, Sept. 1975.

[3] Fortmann, T. E., Y. Bar-Shalom and M. Scheffe, "Sonar tracking of multiple targets using joint probabilistic data association," IEEE Journal of oceanic Eng., July 1983.

[4] Sengupta, D., and R. A. Iltis, "Neural solution to the multitarget tracking data association problem," IEEE Trans. on Aeros. and Electr. Systems, vol. 25, pp. 96-108, Jan. 1988.

[5] Barniv, Y., "Dynamic programming solution for detecting dim moving targets," IEEE Trans. on Aeros. and Elect. Systems, vol. 21, pp. 144-156, Jan. 1985.

[6] Krishnapuram, R., and D. P. Casasent, "Hough transform detection of 3-D curves and target trajectories," Applied Optics, vol. 28, pp. 3479-3486, 1989.

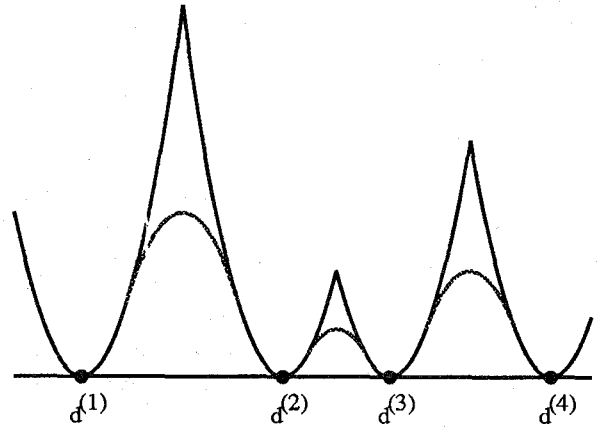

Figure 1. The function g and its approximation g^*

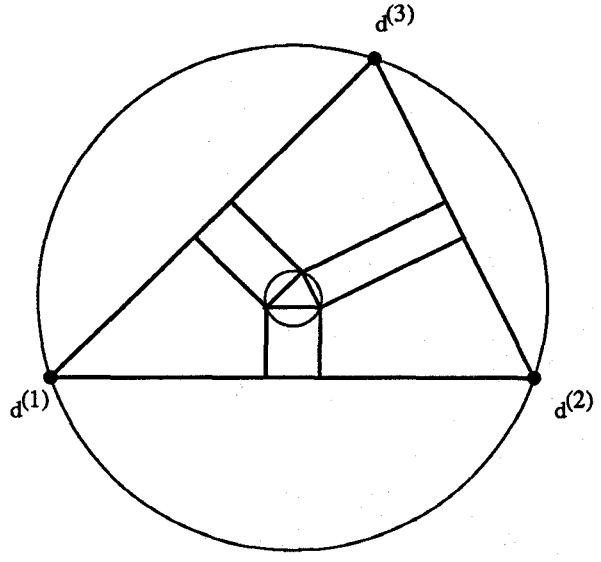

Figure 2. Partition of the domain of g^* according to its definition.

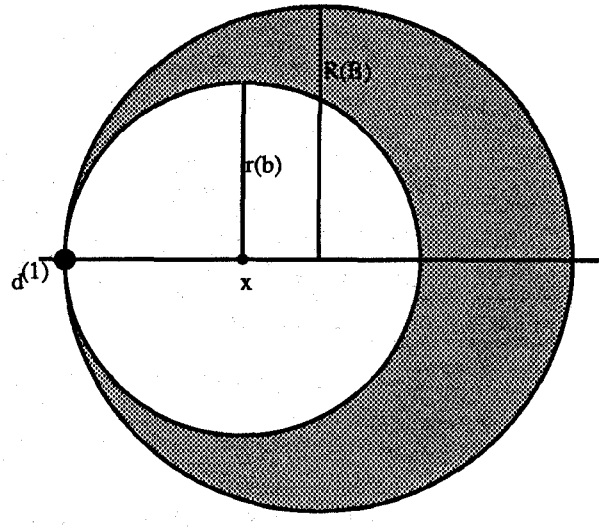

Figure 4. The search window is given by the shaded region $B - b$.

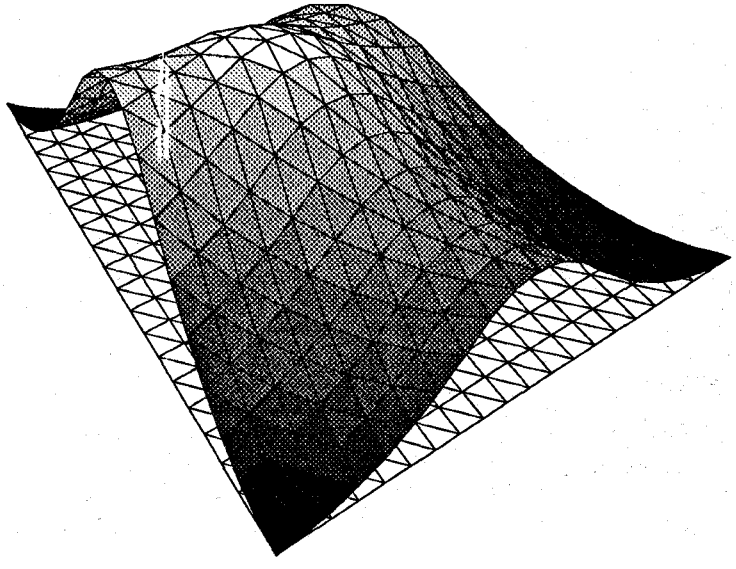

Figure 3. The approximation g^* between three data points

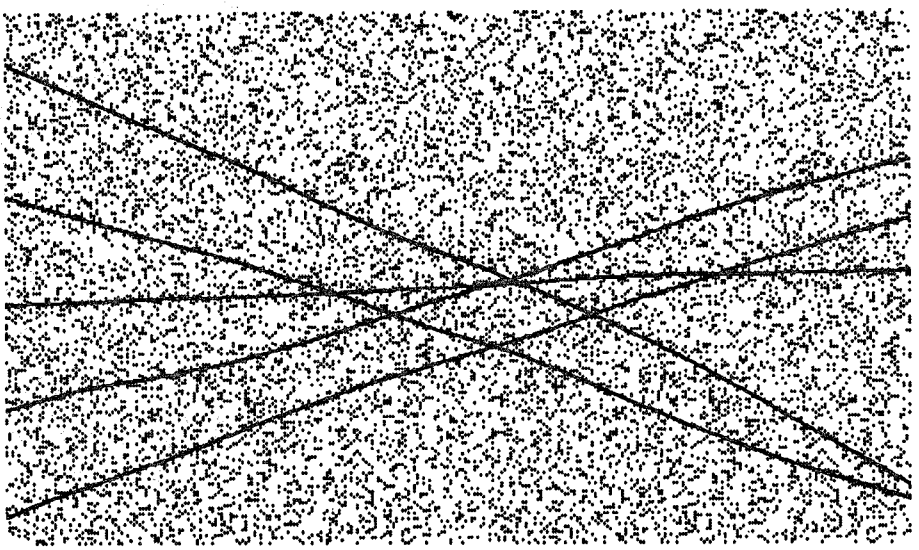

Figure 5(a). The original trajectories plus clutter.

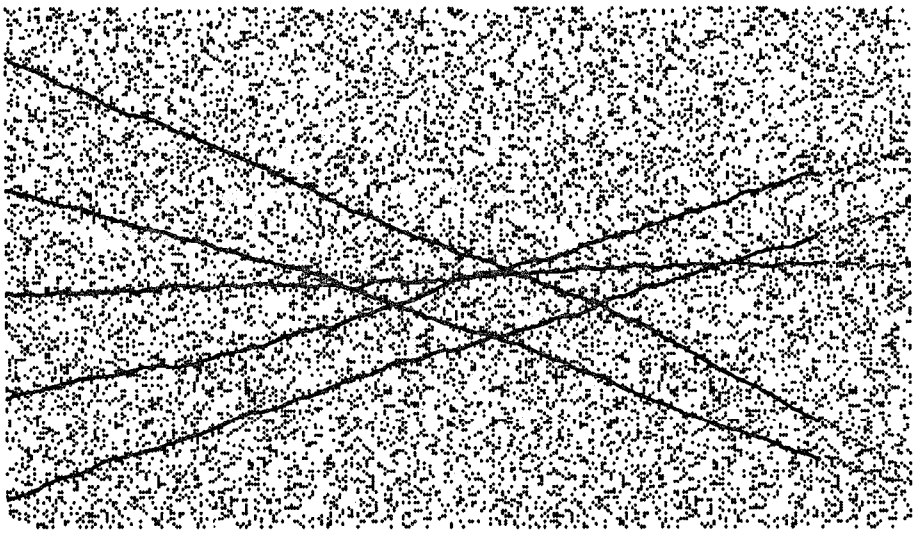

Figure 5(b). The computed trajectories.