Track-stitching using graphical models and message passing

L. J. van der Merwe* and J. P. de Villiers*†
*University of Pretoria, Pretoria, South Africa, loodvdmerwe@gmail.com pieter.devilliers@up.ac.za
†CSIR, Pretoria, South Africa, jdvilliers1@csir.co.za

***Abstract*—In order to stitch tracks together, two tasks are required, namely tracking and track stitching. In this study track stitching is performed using a graphical model and message passing (belief propagation) approach. Tracks are modelled as nodes in a track graph trellis (lattice) structure. This graph is then solved using a Viterbi data association algorithm. A Kalman filter is used to perform tracking, as well as in gating operations and in determining the track-to-track association probability. Multiple crossing targets, with fragmented tracks, are simulated. It is then shown, that the algorithm successfully stitches track fragments together, even in the presence of false tracks, caused by noisy observations.**

I. Introduction

Multiple target tracking (MTT) is a challenging problem that has been encountered in a variety of fields. The requirement of turning raw sensor data into long-term tracks of targets of interest, has application in numerous areas, including civilian surveillance, military target tracking and intent assessment applications.

The track stitching problem is inherent in many MTT environments. The problem is usually divided into two sections, namely tracking and track stitching [1]. During tracking, the track of a target may be broken for a variety of reasons, these include a sufficient number of missed updates, caused by a low detection probability, target occlusions, as well as cases where the update period might be long. This usually results in the target track being dropped, and restarted at a later time. This results in poor long term track maintenance and may in turn negatively influence target classification, threat assessment and resource assignment algorithms. The amount of track fragments and therefore, the number of possible track-to-track associations scales exponentially with the number of targets and time, making the problem more challenging.

The track based multiple hypothesis tracking (MHT) [2] algorithm can be adapted to solve the track stitching problem. This algorithm builds a tree of possible association hypotheses, while deferring the association to a later stage, with the intent of preserving other possible solutions before purging them. The association trees can be solved by a multitude of algorithms, including an integer programming algorithm [3] or by using an N-scan algorithm along with belief propagation [4]. Track fragments can also be modelled as nodes in a flow networks such as in [5] and [6]. The flow networks are then solved by associating the nodes with neighbouring nodes that are highly correlated. Further advancements include estimating the missing data between track fragments using rank minimisation of a Hankel matrix [7] where the missing data is estimated in such a way that it is maximally consistent with the known data.

In this study, a track stitching algorithm is developed, where certain track features are saved. These features are retained, even if the track has been dropped by the tracking algorithm. These features are then used to stitch the track fragments together by modelling the tracks as nodes in a lattice type track graph. The track graph is solved by using a Viterbi data association algorithm [8], [9], [10] to find the most likely paths through the graph, corresponding to the solution paths. In this study, it is assumed that measurements from the sensor arrives in sequence. It is also assumed that the targets are resolved, in that each target can only return one point observation at any given time.

In Section II, methods and algorithms that were developed are presented. In Section III, the results and findings are shown and discussed. Finally, in Section IV, some conclusions are provided, along with perspective work in this direction.

II. Methods and algorithms

In this section methods and algorithms used to perform multiple target tracking, as well as track stitching are discussed. First, it is discussed in this section how tracking is performed by using a track maintenance algorithm, as well a Kalman filter. A discussion follows, on how track stitching is performed through selectively stored information from the track fragments. The information from the track fragments are then used to represent nodes in a graphical model. Finally, a discussion follows on how the graphical model is solved.

A. Tracking and modelling

In order to perform track stitching from track fragments the following must first be performed to create track fragments:

1) Model linear noisy targets which can either be visible or occluded, producing bursts of true positions.
2) Model the true positions of the target as noisy sensor observations, in the presence of false alarms.
3) Run a a track manager and Kalman filter, using observation gating and data association to generate track fragments.

The resulting tracks will then be used to perform track stitching. Figure 1 shows an overview of the short term tracking algorithm, including target and clutter generation. As can be seen from the figure, the tracking environment is first set up, by generating bursts of true targets, and modelling these targets as radar observations. Clutter is then generated, and the sensor observations are inserted into the clutter. The clutter may also create false target tracks.

The sequential track managing algorithm then commences.

Tracks are first initiated based on the observations, next the observation is filtered, the observations in the following scans are also gated, and data association is performed with the observations within the gating area. It is worth noting here, selective information about the tracks are stored, in order to perform track stitching in real time.

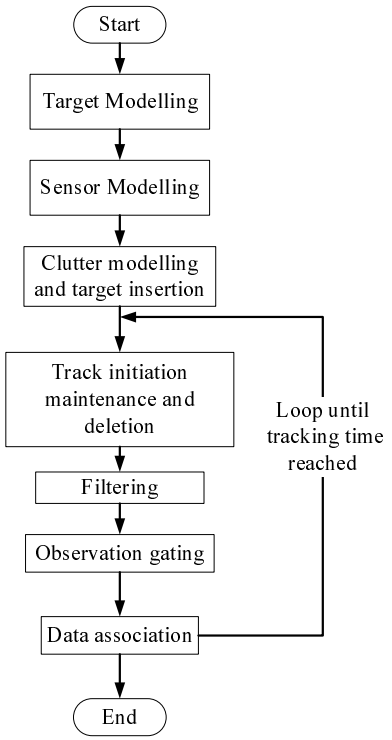

Fig. 1. An overview of the short term tracking algorithm, including target and clutter generation. Note that some track information is retained for use in the track stitching algorithm.

The details of the tracking algorithm used are as follows [14]:

1) True targets are simulated using a linear Markov model with added white Gaussian noise.
2) The sensor is modelled, by adding zero mean Gaussian noise.
3) Sensor observations are in polar coordinates.
4) The amount of clutter at each scan follows a Poison distribution.
5) The reliability of the sensor is modelled by including a probability of detection, P_D, which determines whether an observation was received by the sensor.
6) Track maintenance is performed by a M out of N algorithm.
7) Track initiation is performed using the single point initiation technique[15].
8) The ellipsoidal gating region, G_0, when no attribute data is contained in the measurement [14], is given by

$$d^2 \leq G_0 = 2\ln\left(\frac{P_D}{(1-P_D)(2\pi)^{\gamma/2}\beta_{FA}\sqrt{|S|}}\right). \quad (1)$$

Where β_{FA} is the false alarm rate, $|S|$ is the determinant filter residual covariance and γ is the dimension of the observation vector.

9) Data association is performed using a nearest neighbour approach.

1) Track fragment creation: Track fragments are created by modelling the target as being in one of two discrete states. Either visible or occluded. The Markov model in Figure 2 shows this behaviour. The transition probabilities in Figure 2 should be chosen in such a way that the model creates fragments or burst of tracks. This implies that the transition probabilities between the two nodes should be relatively low, when compared to the probabilities of remaining in the same state, as can be seen in the figure below.

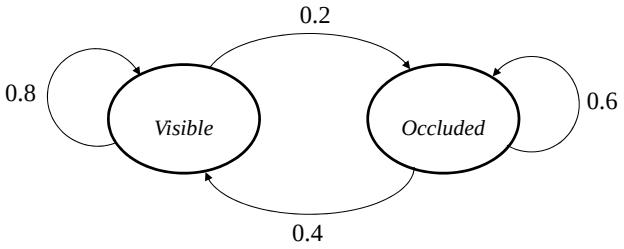

Fig. 2. The target can be either visible or occluded. By choosing the transition probabilities appropriately the model will create track fragments. These probabilities are shown on the transition arrows.

B. Track stitching

In order to perform track stitching, information of the confirmed tracks needs to be saved. This information includes the following:

1) Track start state.
2) Filter covariance at the start of the track.
3) Track end state.
4) Filter covariance at the end of the track.
5) The creation time of the track.
6) The termination time of the track.

Owing to the single point initiation technique used in the tracking algorithm, as mentioned earlier, the very first covariance of the track can not be used as the starting covariance, as this will be much larger than the consequent covariances after a following update has been received.

It is also worth noting here that the following is assumed:

1) The sensor measurements arrive in the correct chronological order, i.e. not out of sequence.
2) A target can generate exactly one measurement at any given time, i.e. the sensor measurements are resolved.
3) The target will not manoeuvre while occluded, and will continue in an almost linear fashion.

Given the above assumptions it can be inferred that a confirmed track can only be the result of exactly one target at any given time, t. Each track can be associated with exactly one target, thus multiple tracks, existing at the same time, must

be mutually exclusive to any target.In subsequent subsections this property is exploited to reduce the number of track-to-track association hypotheses.

The track stitching algorithm can be explained using a two target example. The targets are denoted as M_1 and M_2. Both targets are partially occluded, according to the model mentioned earlier, and produced track fragments $T_1, T_2 ... T_N$. These track fragments are represented by a vector containing the saved track information mentioned earlier in this section. The state diagram for the two targets, M_1 and M_2, is shown in Figure 3. It is assumed that the track fragment indices coincide with the chronological order of the track fragments for each target, although the two targets may have overlapped in time. In Figure 3 the nodes are referred to as states. A state represents a track fragment. A track fragment can either be stitched to a previous track fragment, from the same target or a track fragment can be the result of a new target (false or real). If a track fragment is not associated with an existing target, it represents a new target. A transition represents the possibility that a former track fragment and a later track fragment can result from the same target. Transitions to the same node are reserved for the *no association* hypothesis. In Figure 3 track fragment T_2 can be associated with either target M_1 or M_2. It is also important to note that because of the assumption that observations and therefore the tracks are in the correct sequence, these diagrams do not allow traversing from a later state to an earlier one, i.e. the diagrams take the form of directed graphs.

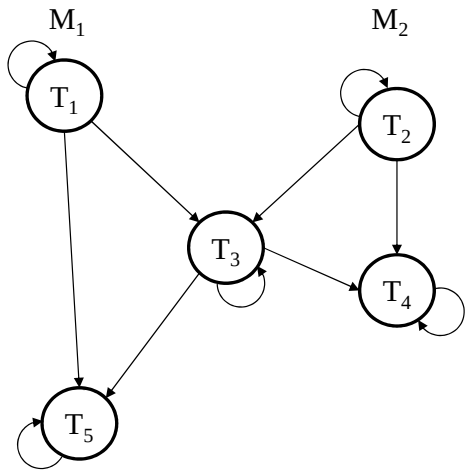

Fig. 3. Two targets M_1 (left) and M_2 (right) with their respective associations shown as arrows. Track fragment T_3 can be assigned to either target M_1 or M_2. If a later track fragment is not associated with a former track fragment, the appearance of new target is assumed (real or false).

The state diagram in Figure 3 can be restructured into a lattice or trellis like diagram. This configuration is referred to in this paper as a track graph trellis. For the sake of clarity, the invalid associations were removed. In implementation however, the invalid associations are set to have a likelihood of zero. The resulting graph can be seen in Figure 4. The columns of the graph represent the time when the track fragments were inserted, at irregular intervals. A column is only inserted in the graph when a new track fragment becomes available. This reduces the computational complexity, in that the graph only has to be solved once a new track fragment becomes available or when a track fragment is updated, instead of at every time step during tracking.

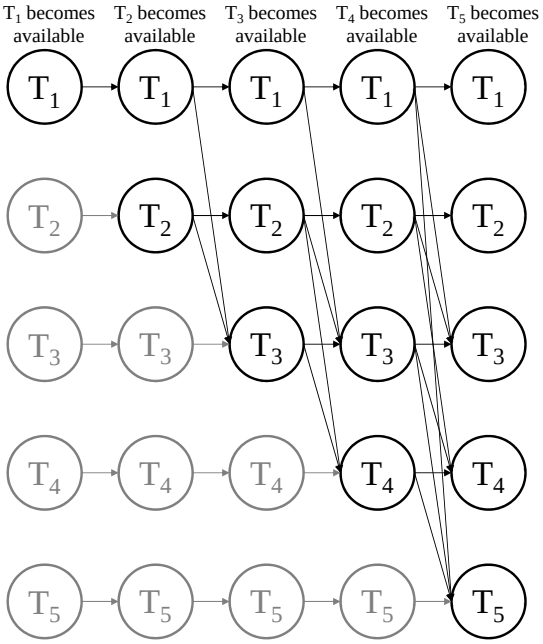

Fig. 4. The track graph trellis for the state diagram in Figure 3. The grey nodes represent states that are inserted to preserve the square structure of the graph.

The association probability of the the track fragment to itself is set equal to one, $p(T_n|T_n) = 1$, i.e. no other associations are possible, up until the column where the fragment was inserted into the graph (grey horizontal arrows). This is done to preserve the square structure of the graph, which makes implementation easier.

1) Solving the track graph trellis: In order to solve the track graph trellis, the Viterbi algorithm is briefly reviewed.

The Viterbi algorithm: The Viterbi [8] algorithm is a related algorithm of the sum-product belief propagation (message passing) algorithm and is often known as the min-sum, or max-product algorithm. The Viterbi algorithm solves the problem of maximising the probability of a sequence [9]. The goal is to find the sequence in the set of states, x, that maximises a global function, g, i.e. the sequence of most probable states. The most probable sequence in x can be defined by the term

$$\arg\max_x g(x). \tag{2}$$

The Viterbi algorithm can be summarised as follows. For a Hidden Markov Model (HMM) with state space S, initial probabilities π_i of being in state i and transition probabilities $a_{i,j}$ of transitioning from state i to state j, outputs $y_1, \ldots, y_T$ are observed. The most likely state sequence, $x_1, \ldots, x_T$, that produces the observations is given by the recurrence relations

in the equations [9]

$$p_{1,k} = p(y_1|k)\pi_k, \tag{3}$$
$$p_{n,k} = p(y_n|k) \max_{x \in S} (a_{x,k} p_{t-1,x}) . \tag{4}$$

Where $p_{n,k}$ is the probability of the most probable state sequence responsible for the first n observations, with the final state k. The Viterbi path can be retrieved by storing pointers of which state x was used in equation 4. Let $\mathrm{Pt}(k, n)$ be the function that returns the value of x used to compute $p_{n,k}$. Then:

$$x_T = \arg\max_{x \in S}(v_{T,x})$$
$$x_{n-1} = \mathrm{Pt}(x_n, n)$$

The algorithm can now backtrack to find the most probable state sequence (Viterbi path) using the pointer function, Pt. The probabilities can of course be implemented as costs (inverse probabilities), in this case the Viterbi algorithm aims to minimise the Viterbi cost.

The track graph can now be solved by maximising the association probability for the associations at each node, in each column of the track graph. This is akin to finding the most probable paths (lowest cost) through the track graph, where the paths are all mutually disjoint [10], i.e. the paths do not share any common nodes. Assuming that the track graph has columns, L, and rows, M. The path probability is given in equation 6, while the path cost is given in equation 5 for a path terminating at column l.

$$D_m = D_{l-1,m^*} + \min(\delta^G_{m,l-1}) \tag{5}$$
$$p(m) \propto \frac{1}{D_m} \tag{6}$$

Where D_{l-1,m^*} is the path cost in the previous column, and was found to terminate in row m^*, and D_m is the new path cost, terminating in row m. $\min(\delta^G_{m,l-1})$ is the minimum transition cost from column $l-1$ to the node in row m, column l. $p(m)$ is the probability associated with the path cost D_m. Again the probabilities can be normalized by ensuring the probabilities of the paths likelihoods sum to 1.

Any of these two equations can be used to determine the most likely paths. The solution to the track graph is therefore similar to a multiple Viterbi algorithm, where the paths are mutually disjoint.

Messages of the current path likelihood are propagated from a node in the current column, to nodes in next column in the graph. These path likelihoods are updated using equation 6.
Once the end of the graph is reached, the algorithm backtracks through the most likely nodes to obtain the set of most likely paths. These paths then represent the most likely track-to-track associations. For the example problem in the previous subsection two solution paths may found, and can be seen in Figure 5.

The number of solutions (stitched tracks) need not be known beforehand, as this is implicitly known. The algorithm only needs to keep solving for solution paths through the graph, until all the rows in the graph have been used at least once.

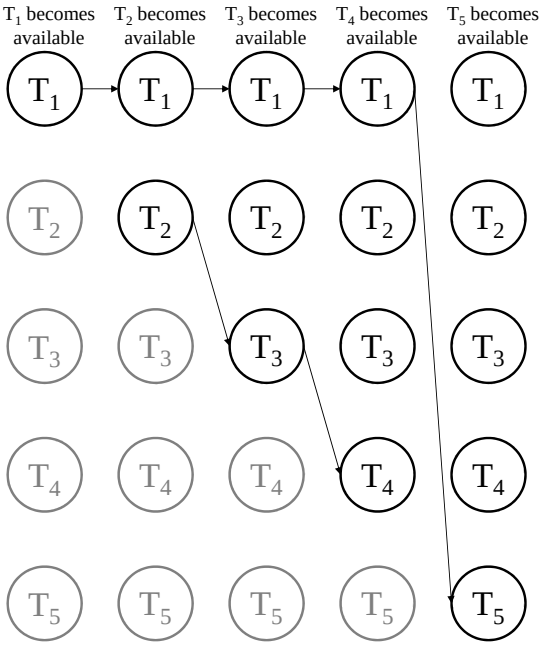

Fig. 5. The two most probable solution paths through the trellis. Track fragments T_1 and T_5 are associated with target M_1, while fragments T_2, T_3 and T_4 are associated with target M_2

2) Possible track-to-track associations: To reduce the number of associations further, ellipsoidal gating is used, as in the tracking algorithm. When a new track fragment is observed, the current position of each previous existing track fragment is predicted using a Kalman filter (i.e. the Kalman filter coasts up to the current point in time for each previous track fragment). The coasting operation of the Kalman filter causes the filter covariance to increase, this in turn increases the size of the ellipsoidal gate. Each track track fragment is evaluated in turn and if the estimated starting position of a the current track fragment falls within this gate of a previous track fragment, these two fragments may be stitched together, depending on the association probability (next subsection). This process is described in Figure 6. In the figure, T_1 is the previous track, while T_2 is the current track. A similar ellipsoidal gating region is used as before, given by

$$G_i = 2 \ln \left(\frac{cP_D}{(1 - cP_D)(2\pi)^{\gamma/2} \beta_{FA} \sqrt{|S|}} \right). \tag{7}$$

The addition of the constant c is to account for the added uncertainty added due to the occlusions and small variations in target bearing. This variable is chosen as close to unity, where $0 < c \leq 1$. Here, γ is the dimension of the estimated state vector of the tracks.

Given the possible associations that does not violate the gating condition, and the time restrictions imposed, as described earlier, a binary association matrix can be constructed. The dimensions of the matrix are equal to the number of track fragments.

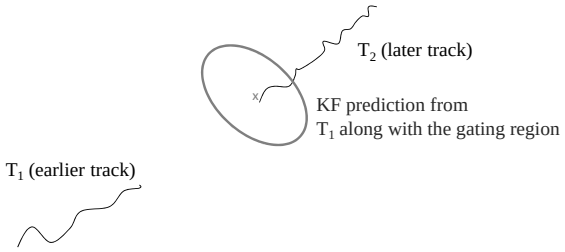

Fig. 6. The gating region and position prediction from previous track, T_1 is shown in grey. The starting position of the current track, T_2, falls within this gating region, and is therefore considered a possible association.

3) Track-to-track association probability: The track-to-track association probabilities will now be defined. These probabilities describe the viability of associating one track fragment to another.

Equations 8 to 11 describe the costs and likelihoods of associating an earlier track, T_i to a later track T_j. These equations hold only when it is assumed that the estimation errors, $\hat{x}^i - x^i$ and $\hat{x}^j - x^j$ are independent [16]. This assumption does not generally hold for closely spaced targets from the same target, but is a simplification. In equations 8 to 11, P_t^{ij} is the combined covariance of the two track fragments, $\hat{\Delta}_t^{ij}$ is the difference between the state estimations, of the two tracks, δ_t^{ij} represents an association cost between the two tracks, and $p(T_i|T_j)$ is the probability of assigning an earlier track, T_i, to a later track, T_j.

$$P_t^{ij} = P_t^i + P_t^j \tag{8}$$

$$\hat{\Delta}_t^{ij} = \hat{x}_t^i - \hat{x}_t^j \tag{9}$$

$$\delta_t^{ij} = \hat{\Delta}_t^{ij\top}[P_t^{ij}]^{-1}\hat{\Delta}_t^{ij} \tag{10}$$

$$p(T_i|T_j) \propto \frac{1}{\delta_t^{ij}} \tag{11}$$

All association probabilities, $p(T_i|T_j)$, entering a node in the graph can be normalised (see Figure 4) by ensuring that all probabilities sum to 1. In the above equations P_t^i is the covariance at the end of the earlier track, P_t^j is the covariance at the start of the later track and P_t^{ij} is the combined covariance at time t. $\hat{x}_t^i$ is the the estimated end position of the earlier track, while $\hat{x}_t^j$ is the estimated starting position of the later track. Association costs are converted to association likelihoods, by inverting the costs.

C. Track fusion and purging

At some point unused tracks need to be purged and associated tracks need to be permanently fused. This is performed by a track purging and fusion algorithm, using a sliding window approach.

Let the size of the window be K. The number of elapsed time steps since the track ended is is recorded. Once the amount of time steps elapsed, where no association was made to the particular track, is larger than the window size, K, the track is purged from memory. If an association was made to a particular track, the amount of time steps in which the association is confirmed is recorded. If this is greater than $K/2$, the two tracks are fused together according to the following rules:

1) The new track has the starting parameters of the earlier track.
2) The new track has the ending parameters of the later track.
3) The missing data is determined using linear interpolation.
4) The two separate tracks are removed from memory, and replaced with the new track.
5) All of the assumptions regarding time constraints, track independence and gating techniques mentioned earlier now applies to the new track.

III. Results and Findings

In this section the results and findings are provided. It is shown that track fragments can be stitched together using the methods discussed in the previous sections.

A. Track fragment generation

The result of two fragmented crossing true target tracks is shown in Figure 7.

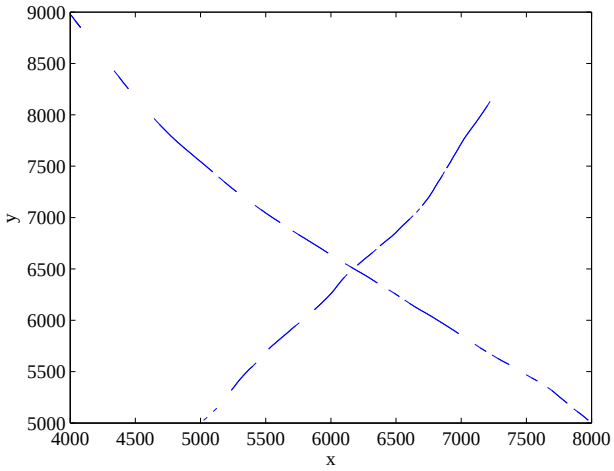

Fig. 7. The true track of a single target, with fragmentation. As can be seen the Markov model described in the previous section creates bursts of visible true tracks.

The following transition probabilities were used for each target:

$$p(Visible|Visible) = 0.8,$$
$$p(Visible|Occluded) = 0.2,$$
$$p(Occluded|Visible) = 0.4,$$
$$p(Occluded|Occluded) = 0.6$$

as shown in Figure 2 in the previous section.

As can be seen from Figure 7, this created bursts of the true target tracks. Tracking observations generated from such a target, created track fragments that is ordinarily dropped by a tracking algorithm.

B. *Tracking and track stitching results*

The short term tracking algorithm was applied to the scenario in Figure 8. The tracks were repeatedly dropped, and then reinitialised at a later stage, only to be dropped again. The estimated track fragments for the true target tracks in Figure 7 are shown in Figure 8. The original true tracks are also shown in Figure 8, along with false tracks and false observations (shown as dots).

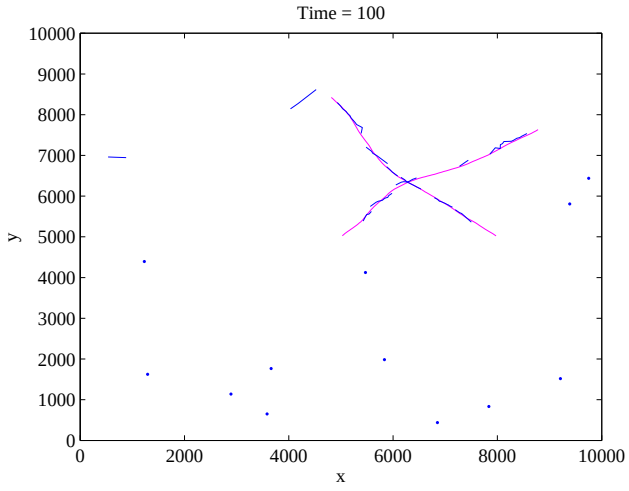

Fig. 8. The estimated track fragments caused by target occlusions for two crossing targets, as estimated by the tracking algorithm.

A gate from one track predicted forward into time is shown in Figure 9. The size of the gate increases as the time to which the prediction is made increases. This not only accounts the uncertainty of where the target will reappear, but also accounts for small deviations from the linear trajectory that was assumed the target would follow.

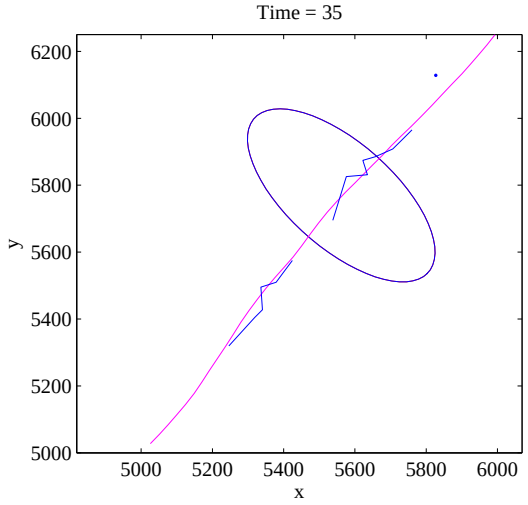

Fig. 9. A track fragment gate shown around a prediction from an earlier track to a later track.

As it is assumed that the observations arrive in sequence, of particular interest is the time intervals during which the tracks existed. For the scenario in Figure 8, the time intervals during which these tracks existed are shown in Figure 10. As can be seen overlapping occurs between the survival times of certain tracks, therefore these tracks could not have been generated by the same target. This fact is exploited to reduce the number of valid associations.

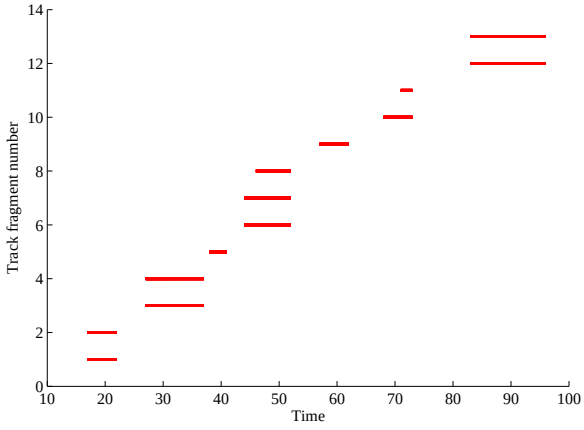

Fig. 10. The time intervals during which the track fragments existed.

The binary association matrix for the scenario in Figure 8 can be constructed using the gating and time information. This is shown in Figure 11. The matrix is symmetric around the diagonal, so only one half needs to be considered.

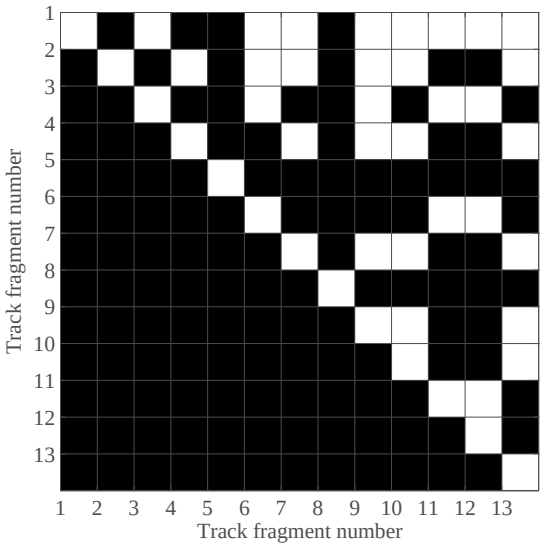

Fig. 11. The possible associations for the scenario in Figure 8. White blocks are possible associations.

Using the track graph model described in the previous section and the above information, the most probable sequence of track fragments can be determined, with the missing data linearly interpolated. The result of stitching the tracks in Figure 8 is shown in Figure 12.

The number of track-to-track associations in Figure 11 can be considerably reduced by using the purging and fusion algorithm described earlier. This is merely used to increase performance, and is not shown here.

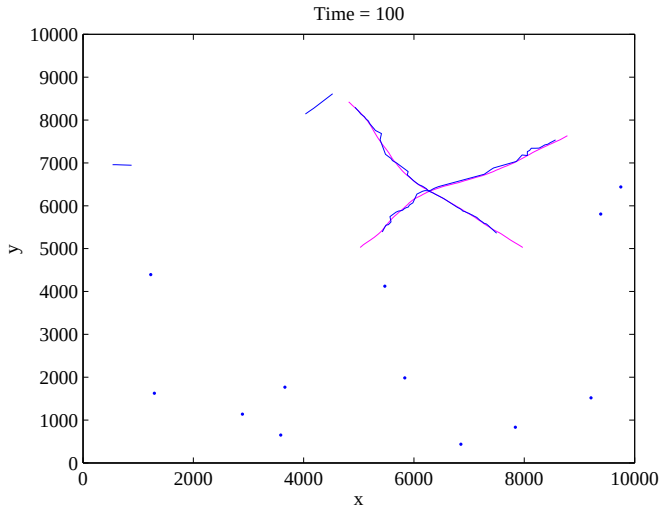

Fig. 12. The stitched tracks with the missing data linearly interpolated, for the scenario in Figure 8.

IV. Conclusion

In this paper it was shown that track stitching can be performed by using graphical models and message passing. The tracks were modelled as nodes in a lattice track graph structure. A method to solve the track graph was also provided. Finally the results were presented and discussed.

Future work includes simulating different target crossing scenarios, to evaluate the effectiveness of the algorithm. It also includes adapting a multiple hypothesis tracking algorithm to perform track stitching, with the aim to evaluate the algorithm against an MHT implementation using Monte Carlo simulations. Future work also includes extending this model to a more general graphical model. Lastly, it includes validating the developed algorithm with real world data, with an increased target density.

Acknowledgment

The authors would like to thank ARMSCOR and the South African Department of Defence for finance support during this study through the LEDGER initiative via the Council for Scientific and Industrial Research (CSIR).

References

[1] R. Ivey, J. Horn, and R. Merket, "Long-duration fused feature learning aided tracking," *Multisensor, Multisource Information Fusion: Architecture, Algorithms and Applications*, Apr. 2012.

[2] D. Reid, "An algorithm for tracking multiple targets," *Automatic Control, IEEE Transactions on*, vol. 24, no. 6, pp. 843 – 854, Dec. 1979.

[3] C. Morefield, "Application of 0-1 integer programming to multitarget tracking problems," *Automatic Control, IEEE Transactions on*, vol. 22, no. 3, pp. 302 – 312, Jun. 1977.

[4] Z. Chen, L. Chen, M. Cetin, and A. Willsky, "An efficient message passing algorithm for multi-target tracking," in *Information Fusion, 2009. FUSION '09. 12th International Conference on*, Jul. 2009, pp. 826 –833.

[5] G. Castanon and L. Finn, "Multi-target tracklet stitching through network flows," in *Aerospace Conference, 2011 IEEE*, Mar. 2011, pp. 1 –7.

[6] C.-Y. Chong, G. Castanon, N. Cooprider, S. Mori, R. Ravichandran, and R. Macior, "Efficient multiple hypothesis tracking by track segment graph," in *Information Fusion, 2009. FUSION '09. 12th International Conference on*, Jul. 2009, pp. 2177 –2184.

[7] T. Ding, M. Sznaier, and O. Camps, "Fast track matching and event detection," in *IEEE Conference on Computer Vision and Pattern Recognition*, vol. vol.2, Jun. 2008, pp. 1–8.

[8] A. Viterbi, "Error bounds for convolutional codes and an asymptotically optimum decoding algorithm," *Information Theory, IEEE Transactions on*, vol. 13, no. 2, pp. 260 –269, Apr. 1967.

[9] J. Forney, G.D., "The viterbi algorithm," *Proceedings of the IEEE*, vol. 61, no. 3, pp. 268 – 278, Mar. 1973.

[10] G. Pulford, "Multi-target viterbi data association," in *Information Fusion, 2006 9th International Conference on*, Jul. 2006, pp. 1 –8.

[11] C. M. Bishop, *Pattern Recognition and Machine Learning*. Secaucus, NJ, USA: Springer-Verlag New York, Inc., 2006.

[12] L. Rabiner, "A tutorial on hidden markov models and selected applications in speech recognition," *Proceedings of the IEEE*, vol. 77, no. 2, pp. 257 –286, Feb. 1989.

[13] L. Rabiner and B. Juang, "An introduction to hidden markov models," *ASSP Magazine, IEEE*, vol. 3, no. 1, pp. 4 –16, Jan. 1986.

[14] S. Blackman and R. Popoli, *Design and Analysis of Modern Tracking Systems*. Artech House, 1999.

[15] M. Mallick and B. L. Scala, "Comparison of single-point and two-point difference track initiation algorithms using position measurements," *Acta Automatica Sinica*, vol. 34, no. 3, pp. 258 – 265, 2008.

[16] Y. Bar-Shalom and X. Li, *Multitarget-Multisensor Tracking:Principles and Techniques*. YBS Publishing, 1995.