Multiple Target Tracking with Recursive-RANSAC and Machine Learning

Kyle Ingersoll

March 10, 2015

Abstract—**The Recursive-Random Sample Consensus (R-RANSAC) algorithm is a novel multiple target tracker designed to excel in tracking scenarios with high amounts of clutter measurements. R-RANSAC classifies each incoming measurement as an inlier or an outlier; inliers are used to update existing tracks whereas are used to generate new, hypothesis tracks using the standard RANSAC algorihtm. R-RANSAC is entirely autonomous in that it initiates, updates, and deletes tracks without user input. The tracking capabilities of R-RANSAC are extended by merging the algorithm with the Sequence Model (SM). The SM is a machine learner that learns sequences of identifiers. In the tracking context, the SM is used to learn sequences of target locations; in essence, it learns target trajectories and creates a probability distribution of future target locations. Simulation results demonstrate significant performance improvement when R-RANSAC is augmented with the SM, most noticeably in situation with high signal-to-noise ratio (SNR) and infrequent measurement updates.**

I. Introduction

In this paper, we present the Recursive-RANSAC (R-RANSAC) multiple target tracking (MTT) algorithm. The filtering, track initialization, and track management blocks of R-RANSAC are explained in Sections III-A, III-B, and III-C respectively. Section III-D discusses alternatives ways to view R-RANSAC in the context of other MTT algorithms. We then describe how tracking performance can be improved using machine learning. Section IV briefly describes the machine learner used here, the Sequence Model. Section V explains how the Sequence Model was incorporated into the R-RANSAC framework. Section VI describes the simulation that was developed to test the Sequence Model/R-RANSAC (SM/R-RANSAC) algorithm. Section VII presents simulation results and Section VIII discusses conclusions about the SM/R-RANSAC algorithm and future research directions[1].

II. Random Sample Consensus

The RANSAC algorithm [2] was designed to estimate the parameters of a signal in the presence of gross errors. When gross errors are present, traditional methods like least-squares regression often poorly model the signal of interest. RANSAC has been successfully applied to a wide range of computer vision problems. One of the most well-known applications of RANSAC is the computation of the homography, or geometrical transformation, between two images. When computing a homography, feature points along with their accompanying descriptors are found in each image. The feature points are then matched across the images by comparing their descriptors. While many features are correctly matched, incorrect matches are inevitable. Figure 1 demonstrates this situation[2]. Given this set of feature matches, RANSAC is used to compute the true homography; the resulting panoramic image can be seen in Figure 2.

The RANSAC algorithm proceeds as follows. First, an assumed signal model is selected. In the homography example, the chosen model is a 4 by 4 matrix encoding translation, rotation, and scaling information. Second, a random subset of data points is selected. The number of data points selected is the minimum number needed to estimate the model's parameters. In the homography case, four points are needed. Third, a model is constructed with this random subset of data points. Fourth, all remaining data points are classified as either inliers or outliers

[1]For further explanation on R-RANSAC, tracker-sensor feedback in the R-RANSAC framework, or machine learning with R-RANSAC, we refer the interested reader to [1].

[2]Images are from the windows dataset and can be accessed at https://canvas.instructure.com/courses/743674/assignments/1929377

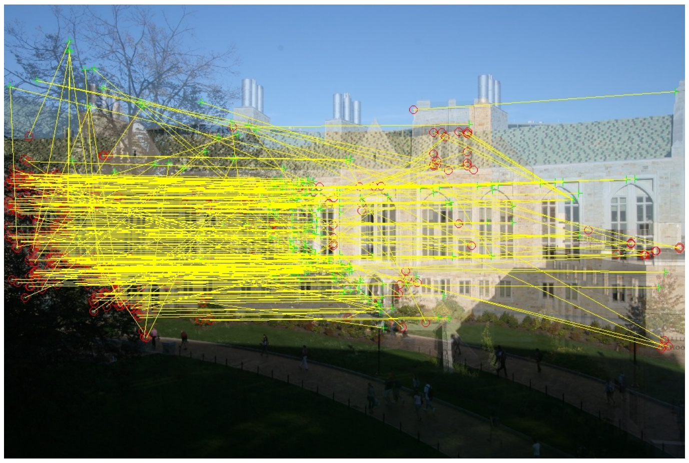
Fig. 1: Two overlapping images of a building are overlaid on top of each other. The transformation between these images is almost purely translational in the horizontal direction. Features in one image are labeled in red and features in the other image are labeled in green. Matching features are indicated by yellow lines. The majority of features are correctly matched, though some incorrect matches do exist.

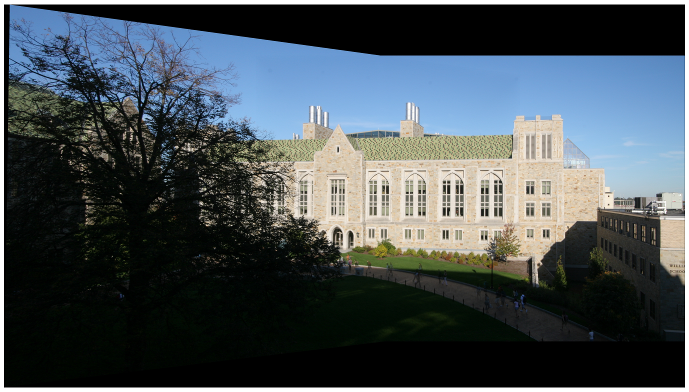
Fig. 2: The resulting panoramic image produced with the homography calculated from the correctly matched feature pairs identified in Figure 1.

to this new model. This inlier/outlier classification is performed by setting an inlier threshold; all points that fall within this threshold are denoted as inliers and are stored as the new model's consensus set. Steps two through four are performed iteratively. During the iterations, the model with the largest consensus set, or in other words, the model with the most support, is stored. At the end of the iterations, a smoothed model is produced by performing a least-squares regression on the consensus set of the model with the most support. To summarize, RANSAC generates several hypotheses of how to best model the available data and relies on the data to determine which hypothesis has the most support.

III. Recursive-RANSAC

Recursive-RANSAC was originally developed as a multiple object filter by Dr. Randal W. Beard and Dr. Peter C. Niedfeldt at Brigham Young University. The motivation behind R-RANSAC was to design a filter that inherited RANSAC's ability to robustly reject gross errors, but that could be used to estimate multiple signals. Reference [3] introduces R-RANSAC as a method of estimating multiple static signals that are updated via recursive-least squares. R-RANSAC is extended in [4] to estimate time-evolving signals.

A. Filtering

R-RANSAC maintains a model set, a set of hypothesis tracks described by their state estimate $\mathbf{x}$, error covariance $\mathbf{P}$, and consensus set χ. At every time step, each hypothesis track is propagated forward using the predict step of the Kalman filter, given by

$$\mathbf{x}_{t|t-1} = A\,\mathbf{x}_{t-1} \tag{1}$$

$$\mathbf{P}_{t|t-1} = A\,\mathbf{P}_{t-1}\,A^\top + Q \tag{2}$$

where A is the state transition matrix of the assumed motion model and Q is the covariance of the process noise.

Once all tracks have been propagated forward, an inlier threshold τ_R is set around each track; the ℓ_2 norm is used as a threshold. An inlier region is created when the inlier threshold is applied ($I_\text{R} = \{z : ||z - C\,\mathbf{x}_{t|t-1}||_1 < \tau_\text{R}$). For a given track, all measurements from the current scan are classified as inliers or outliers to that track according to whether or not the measurements fall within the track's inlier region. Each inlier is used to update the track using the update equations of the Kalman filter, given by

$$\mathbf{x}_t = \mathbf{x}_{t|t-1} + K\,(z - C\,\mathbf{x}_{t|t-1}) \tag{3}$$

$$\mathbf{P}_t = (I - K\,C)\,\mathbf{P}_{t|t-1} \tag{4}$$

where z is the measurement, C is the measurement observation matrix that relates the measurement to the target states, I is an identity matrix of the same dimensions as A, and K is the Kalman gain and is given by

$$K = \mathbf{P}_{t|t-1}\,C^\top\,(R + C\,\mathbf{P}_{t|t-1}\,C^\top)^{-1} \tag{5}$$

where R is the covariance of the sensor noise. The term *inlier region* is phraseology borrowed from RANSAC. In the tracking community, this region is more commonly referred to as a gate or a *measurement validation region* (see, for example, [5]). R-RANSAC uses an inlier region of fixed volume and bases the size of the inlier region on the assumed measurement noise covariance. Successful extensions of R-RANSAC have replaced the Kalman filter with a probabilistic data association (PDA) filter.

B. Track Initialization

Measurements that are outliers to all existing tracks are used to initialize new tracks using a RANSAC-based method. R-RANSAC stores all the measurements from the past $\mathrm{N_w}$ time steps in a measurement history window. When a measurement is found to be an outlier to all existing tracks, a random subset of measurements (which includes the outlier measurement under consideration) is selected. Using this subset of measurements, a model is computed. The measurement history window is then searched for other measurements that support this model, i.e. that meet the inlier threshold. This process of creating hypothesis models is repeated iteratively and the model with the largest consensus set is stored. At the end of ℓ iterations, the model with the most support is then propagated forward in time to the current time step. As it is propagated forward, it is updated by the measurements that compose its consensus set. These propagation and update steps are performed with a Kalman filter. This new model is then appended to the model set. The RANSAC iterations are also terminated if the cardinality of a model's consensus set exceeds $\gamma = \frac{\tau_\rho}{\mathrm{N_w}}$. The equations used to generate the hypothesis models are given in [6]. Due to the RANSAC-based initialization method, tracks are often referred to as models in the R-RANSAC context. The two terms are used interchangeably here.

Figure 3 provides a snapshot of a single time step of R-RANSAC. At this time step, there are three tracks in the model set, labeled by blue X's. There are several incoming measurements, drawn as circles. The inlier region of each track is indicated by a red square. Measurements classified as inliers are drawn in cyan whereas outlier measurements are drawn in orange. Each orange-colored measurement will be used to generate a new, hypothesis track which will then be appended to the model set.

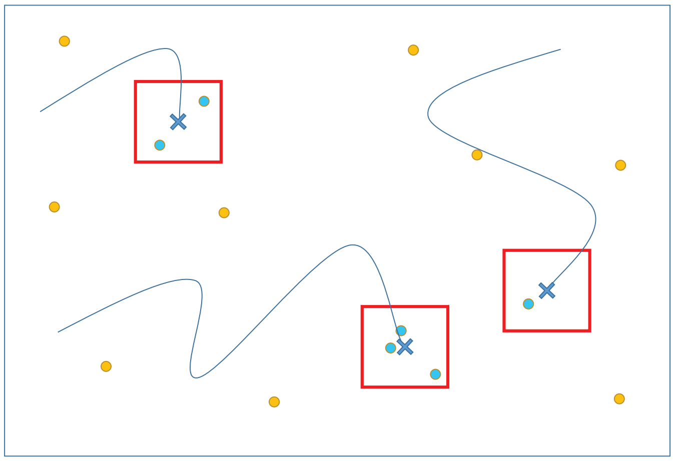

Fig. 3: A single time step of R-RANSAC. Current state estimates are indicated by blue X's. Inlier measurements are indicated by cyan circles. Outlier measurements are indicated by orange circles. The inlier regions are displayed as red squares.

C. Track Management

The remaining steps of R-RANSAC fall under the umbrella of track management: identifying valid tracks, merging redundant tracks, and pruning low-support tracks.

At the beginning of every time step, the consensus set of each track in the model set is updated by removing older measurements that have left the measurement history window. Each track's inlier ratio ρ is also calculated by $\rho = \frac{|\chi|}{Nw}$ where $|\chi|$ is the cardinality of the consensus set. Good, or valid, tracks are identified with a good model threshold and a timeline threshold. The good model threshold τ_ρ is the minimum inlier ratio for a model to be considered a good model. Likewise, the timeline threshold τ_{T} is the minimum number of time steps a model must have existed to be considered a good model. Models that meet both thresholds are outputted as good models for that time step.

In order to limit the number of false positive tracks, redundant tracks must be identified and removed from the model set. The Mahalanobis distance is used as a merging criteria. If two tracks meet the merging criteria, the track with the higher inlier ratio is retained and the other one is discarded. At the end of each time step, the tracks in the model set are ordered by their inlier ratio. The model set has a fixed size M and is truncated at each time step, thereby removing the least probable tracks.

D. R-RANSAC: Alternative Viewpoints

R-RANSAC was initially developed as a standalone multiple object filter, a fully-integrated package that was mutually exclusive with other tracking approaches such as joint probabilistic data association (JPDA) and multiple hypothesis tracking (MHT). However, in the process of further evolving R-RANSAC, it has been beneficial to view R-RANSAC in other ways.

R-RANSAC can be viewed purely as a track initialization algorithm, albeit one that imposes certain constraints on the filtering and data association blocks of the tracker. As a track initialization algorithm, R-RANSAC simply requires that each incoming measurement be classified as an inlier or an outlier. Given this requirement, any number of filters and data association techniques may be used with R-RANSAC. Consequently, R-RANSAC becomes a modular algorithm whose blocks can be substituted with application-specific replacements depending on the tracking scenario. This idea does not diminish the impact or utility of R-RANSAC; rather, it makes R-RANSAC highly adaptable, and applicable to an even greater number of situations. Figure 4 illustrates the modularity of R-RANSAC and includes several algorithms that can be used for data association and filtering.

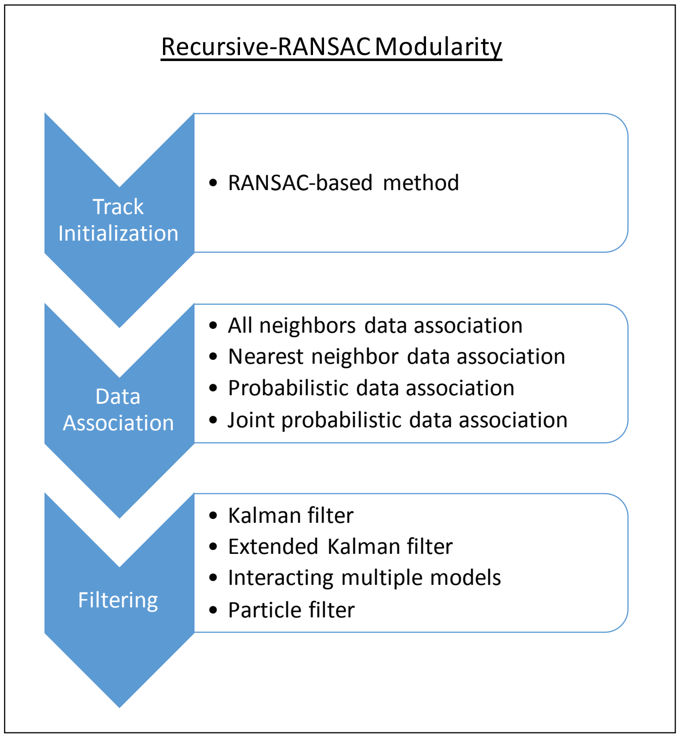
Recursive-RANSAC Modularity
Track Initialization
• RANSAC-based method
Data Association
• All neighbors data association
• Nearest neighbor data association
• Probabilistic data association
• Joint probabilistic data association
Filtering
• Kalman filter
• Extended Kalman filter
• Interacting multiple models
• Particle filter

Fig. 4: Recursive-RANAC as a modular algorithm.

IV. Sequence Model

The Sequence Model is an extension of the Sequence Memoizer, a machine learner first proposed by Wood, Gasthaus, et. al. in [7]. The Sequence Memoizer is a hierarchical Bayesian model designed to capture long-range dependencies in discrete data. The Sequence Memoizer is very appropriately named: it learns *sequences* of data. A classic application of such a learner would be in language prediction.

The SM was applied to a multiple agent, MTT problem in an urban environment in [8]. The SM, like the Sequence Memoizer, learns sequences of discrete data. Tracking usually takes place in a continuous field of view, so the first step in applying the SM to tracking is to discretize the field of view. In tracking, the SM learns sequences of target locations instead of continuous target paths. The grid size chosen to discretize the field of view is very important; an overly fine discretization makes it more difficult for the SM to learn sequences whereas an overly coarse discretization reduces the amount of usable information in the SM's prediction.

The SM creates a belief model of the field of view. For each grid location in the belief model, the probability that a target occupies that location in the future is estimated. The SM constructs this belief model in a Monte Carlo-like way by propagating forward several probable trajectories into the future. Propagating more possible trajectories forward creates a belief model that better estimates the true distribution, but also results in longer execution times. The number of possible trajectories was set at 100 in our simulations. The value of the belief model at a given grid location is the proportion of probable trajectories that passed through that location. An example belief model from the simulations presented in Section VI can be seen in Figure 5.

Conceptually, the SM can be a powerful tool in tracking. It has the natural ability to learn road networks, including details like the locations of stoplights, one-way streets, and common U-turn locations. The SM, because it requires a discretization of the environment, can also easily incorporate prior

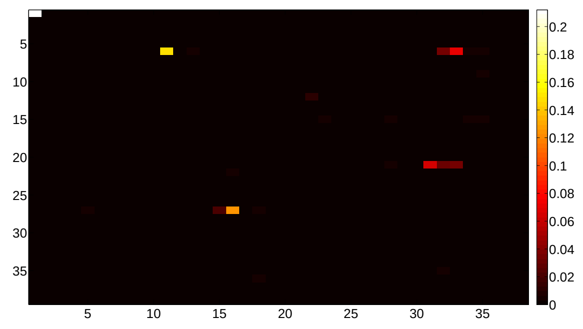

Fig. 5: An example belief model constructed by the Sequence Model. The probability of a target occupying a given grid location at the next time step is indicated by the color scale.

knowledge one might have of the tracking environment. Grid locations corresponding to no travel areas (buildings instead of streets, for example), can have their probability mass zeroed out or greatly reduced. Incorporating prior knowledge about the environment into a Kalman filter estimate is much more difficult.

The SM does, however, present some difficulties when applying it to tracking. First, the SM, like all machine learning algorithms, requires a training period before it begins to make accurate predictions. Care must also be taken to only train the SM with tracks that have a high probability of representing true targets to prevent spurious tracks from corrupting the training data. The most significant shortcoming of the SM in its current state is that it produces a single belief model describing the distribution of future locations for *all* targets. In [8], the SM is used to perform path planning for a cooperative team of UAVs with the goal of maximizing the number of targets seen. The shared belief model does not conflict with this objective. The authors also assume perfect data association; unfortunately, data association is one of the most difficult problems in real tracking scenarios. Because of this shared belief model, R-RANSAC cannot leverage the SM to associate measurements of closely spaced targets; the SM can only be used to distinguish between clutter measurements and target-associated measurements of widely spaced targets.

A. *Other Learning Approaches*

Machine learning is often applied to video-based target tracking in order to learn a visual appearance model of the targets. Reference [9] provides a representative example of this class of algorithm; when targets are not interacting, they are tracked with individual trackers and an appearance model is constructed. This appearance model includes positive examples (templates of the correct target) and negative examples (templates of other targets). When targets interact, the appearance models are used to distinguish between the targets. This use of machine learning is fundamentally different to the use proposed here. We seek to learn target trajectories as opposed to target appearances.

In our literature review, we only came across one other example of using machine learning to learn target trajectories. In [10], a motion map is used to learn non-linear motion patterns in the scene. This motion model helps to connect small sequences of associated measurements known as tracklets into larger sequences known as tracks. An affinity score is calculated between tracklets and learned motion patterns using the head and tail positions and velocities of the different segments. The motion pattern that receives the highest affinity score is used to connect the tracklets. This is an online learning algorithm; the motion model is constructed during tracking with high confidence tracklets. Reference [11] extends this method by using a Conditional Random Field to model the track affinities and dependencies, and by calculating the affinity scores globally. This method differs from our method in that it is a post-processing algorithm.

V. Incorporating the Sequence Model with R-RANSAC

When incorporating the SM into the R-RANSAC framework, we primarily use it as a tool to improve data association. We expect using the SM to be most advantageous in situations with infrequent measurement updates and a high proportion of clutter measurements. R-RANSAC relies entirely on its assumed dynamic model to perform data association. In the case of infrequent measurement updates, a target can deviate significantly from its assumed model, thereby making it much less probable that R-RANSAC correctly associates measurements. The SM does not share this inherent weakness of R-RANSAC; its belief model is constructed entirely from past observations of the target. Consequently, provided the SM has been sufficiently trained and the target does not deviate from its previous paths, the SM should be able to correctly associate measurements independent of the time between measurement updates. The simple example of a road with a right-hand turn clearly illustrates this point. Just prior to the turn, R-RANSAC propagates the state estimate forward and off the road. The SM, on the other hand, has never observed a target continuing straight on this section of road. Instead, it has observed targets making a right hand turn at that location and thus it assigns probability mass to the right.

R-RANSAC measurement association probabilities are calculated using the covariance of the innovation. The SM association probabilities are calculated by interpolating between grid locations in the belief model. Both sets of association probabilities are normalized. A weighting between the R-RANSAC and SM association probabilities is computed. Several approaches have been suggested on how to best weight these probabilities including using mean-based, entropy-based, or random weightings. Ideally, the weighting would be based on the certainty of each prediction. We attempt to approximate this ideal weighting by allowing the SM weight to grow linearly from zero to an upper bound $P(\text{SM})_{\text{max}}$. The SM weight reaches $P(\text{SM})_{\text{max}}$ when the track's age reaches ℓ_{SM} time steps; after ℓ_{SM} time steps we are confident the SM has been sufficiently trained. In the simulations presented here, $P(\text{SM})_{\text{max}} = 0.9$ and $\ell_{\text{SM}} = 300$.

When used with R-RANSAC, the SM is initialized with an arbitrarily large number of targets. All valid tracks outputted by R-RANSAC are used to update the SM. The R-RANSAC track with good model number i updates the belief model for the i^{th} target in the SM. A high τ_{T} value of 15 is used to prevent spurious tracks from corrupting the SM. The SM is updated with the highest probability measurement associated with a track instead of the track's state estimate. Valid tracks that were not updated at the current time step are not used to update the SM. The inlier region of R-RANSAC is expanded to account for possibly extreme intra-time step target maneuvers ($\tau_{\text{R}} = 70$). A grid size of 15 is used to discretize the environment.

Because the SM/R-RANSAC algorithm is designed to excel in tracking situations with infrequent measurement updates, the nearly constant velocity (CV) model is used in place of a higher-order linear model such as the nearly constant jerk (CJ) model. The CJ model performs poorly in these situations because its sensitive higher-order terms result in inaccurate predictions of future target locations far into the future. The CV model, although it does not model turning behavior, is less sensitive to errors in the state estimate and oftentimes more accurately predicts future target locations. The CV model also behaves in a more stable fashion when a target has maneuvered between time steps. In the case of a right hand turn, the CJ model requires several post-turn measurements to converge to the true path. Conversely, the CV model snaps to the true path after receiving a single post-turn measurement.

VI. Learning Simulation Environment

As described in Section IV, the SM is still under active development and can only be reasonably expected to improve tracking performance in certain situations. To fairly evaluate the performance of the SM/R-RANSAC combination, the simulated MTT scenario was designed to showcase the expected strengths of the new algorithm.

A MTT scenario was simulated in Matlab in order to test the SM/R-RANSAC algorithm. The simulation environment consists of a 550 x 550 field of view with four targets. Each target travels in a rectangular path, with each path being located almost wholly in one of the quadrants of the field of view. The target measurements are corrupted by normally-distributed noise with a standard deviation of 0.5. The simulation begins with a 600 time step learning period; this learning period contains no clutter measurements. After the learning period, alternating 400-time step periods with clutter and 200-time step periods without clutter occur until time step 2800 (the end of the simulation). The time periods with clutter are referred to as the "jamming" periods throughout this discussion. The periods without clutter measurements are designed to allow R-RANSAC to reacquire the targets and establish context for the learner. Figure 6 displays the number of clutter measurements over the entire simulation. The number of clutter measurements during each "jamming" period builds to a peak and then falls back to zero. Each successive "jamming" period rises to a higher maximum number of clutter measurements resulting in more difficult tracking as the simulation progresses. Figure 7 shows a snapshot of the simulation environment during a period of clutter measurements.

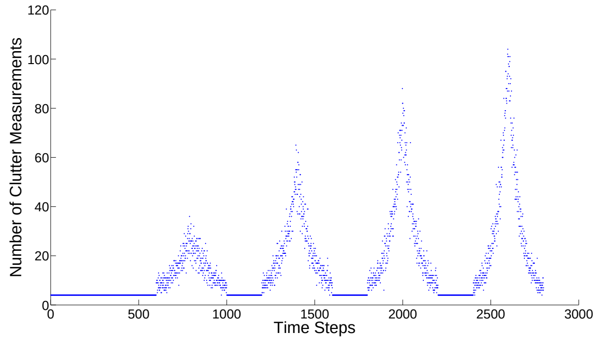

Fig. 6: The number of clutter measurements at each time step in the learning simulations.

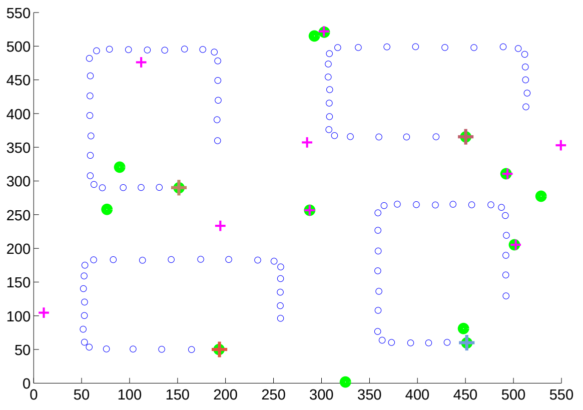

Fig. 7: A snapshot of the simulation environment during a period of clutter measurements. The measurements of the current time step are shown as green circles. The most recent 25 measurements generated by the true targets are shown as smaller, blue circles. The good R-RANSAC tracks are displayed as randomly colored, larger crosses. The hypothesis tracks are displayed as smaller, magenta crosses.

To make R-RANSAC compatible with this particular simulation, two modifications are necessary. First, tracks' consensus sets are updated with *all* of the inlier measurements instead of only with the nearest neighbor measurement. Second, an adaptive τ_ρ is used. A baseline value of 0.8 is assumed for τ_ρ; this baseline value is added to the average number of expected inlier measurements based on the average size of the measurement scans in the measurement history window, the area of the field of view, and the inlier region area. This adaptive τ_ρ appears to work well during all stages of the simulation. These two modifications are necessary to distinguish valid tracks from spurious tracks during the "jamming" stages of the simulation.

TABLE I: MOT Results - R-RANSAC with the Sequence Model

Simulation	MOTP	MOTA	MD	FP	MM
SM/RR - 9	0.4960	0.9793	0.0000	0.0203	0.0004
RR - 9	0.4452	0.9428	0.0207	0.0281	0.0083
SM/RR - 13	1.1883	0.9818	0.0000	0.0174	0.0007
RR - 13	1.7101	0.7896	0.1500	0.0088	0.0515
SM/RR - 17	11.2009	0.8690	0.0085	0.0911	0.0314
RR - 17	15.3591	0.5766	0.2718	0.0233	0.1284

TABLE II: Average OSPA Scores - R-RANSAC with the Sequence Model

Simulation	OSPA	OSPA-T	Std. Dev. (OSPA-T)
SM/RR - 9	4.1499	4.1786	0.4883
RR - 9	12.8371	13.0207	1.0472
SM/RR - 13	4.3613	4.3839	0.5872
RR - 13	35.8561	36.4852	1.2125
SM/RR - 17	20.3627	20.5861	0.7797
RR - 17	60.5101	61.4912	0.3926

VII. Learning Results

Three sets of simulations were run in order to compare R-RANSAC with the SM/R-RANSAC algorithm. The simulations are differentiated by the length of their time steps: 9, 13, and 17 seconds, respectively. In Tables I and II, the simulations are labeled as "SM/RR - *dt*" and "RR - *dt*" where *dt* indicates the time step length. Five trials were run with each simulation. Results were only extracted from the post-learning period interval of the simulation (i.e. the last 2200 time steps). The optimal subpattern assignment (OSPA) [12], [13] and multiple object tracking (MOT) [14] metrics were used. The R-RANSAC parameters were kept constant across all of the trials, except for the measurement noise covariance which was increased by one order of magnitude for the 17 second time step trials.

For all time step lengths, the SM/R-RANSAC algorithm outperformed R-RANSAC. As expected, the disparity in performance increases as the time step grows larger. This is because the Kalman filter estimate of future target location used by R-RANSAC becomes increasingly less trustworthy with longer time steps, whereas the SM estimate is less affected by longer time steps. One interesting observation is that SM/R-RANSAC's performance remained relatively constant between the 9 and 13 second simulations. This result is especially relevant to UAVs which often have strict limits on computational power: the same level of tracking performance can be achieved even when receiving measurements

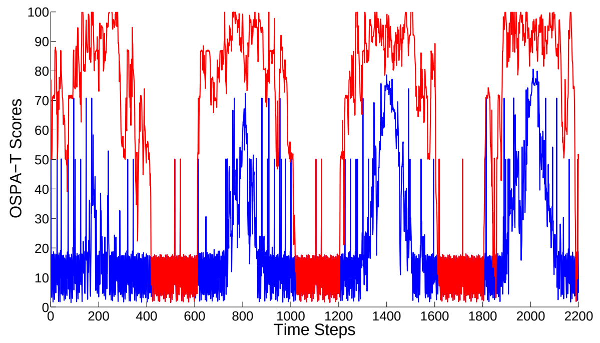

Fig. 8: The OSPA-T scores for a single trial of "SM/RR - 17 (blue)" and "RR - 17" (red).

and updating the model set much less frequently.

Figure 8 plots the average OSPA-T scores at each time step for the five trials of the "SM/RR - 17" and "RR - 17" simulations. During the periods without clutter measurements, R-RANSAC tracks very well and SM-R-RANSAC tracks perfectly. During the periods with clutter measurements, SM/R-RANSAC performs noticeably better than R-RANSAC. Figure 8 also shows that R-RANSAC's performance degrades as soon as the clutter measurements begin and its performance only improves after the clutter measurements have ended. SM/R-RANSAC, on the other hand, maintains excellent performance through the early stages of the "jamming" periods and strongly recovers before the "jamming" periods are over.

There is a significant time penalty associated with using the SM. R-RANSAC averaged 2.6111×10^{-6} seconds per time step whereas the SM/R-RANSAC algorithm averaged 2.3889×10^{-5} seconds per time step.

VIII. Conclusions

The ideas and results presented here are more proof-of-concept than fully developed. The simulation results showed that the SM significantly improved data association in situations with infrequent measurement updates and high amounts of clutter. However, this simulation was very simplistic: the targets were constrained to easily-learned and widely spaced paths. Future research should include simulations with interacting targets that deviate occasionally from their nominal paths. Eventually, experiments should be run on video where the objects of interest do not follow constrained paths. Future work will also need to look at determining the optimal discretization of the environment and the optimal way of combining the SM and R-RANSAC belief models. All of this future work, though, hinges on the continued development of the SM. The SM needs to be able to maintain separate belief models for individual targets for it to be successfully applied to more realistic tracking scenarios.

References

[1] James Kyle Ingersoll. *Vision-Based Multiple Target Tracking Using Recursive-RANSAC*. PhD thesis, Brigham Young University, 2015.

[2] Martin A. Fischler and Robert C. Bolles. Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography. *Communications of the ACM*, 24(6):381–395, June 1981.

[3] Peter Niedfeldt and Randy Beard. Recursive RANSAC: Multiple Signal Estimation with Outliers. In *Nonlinear Control Systems*, volume 9, pages 430–435, September 2013.

[4] Peter C Niedfeldt. *A Novel Multiple Target Tracking Algorithm in Clutter: Recursive-RANSAC*. PhD thesis, Brigham Young University, 2014.

[5] Yaakov Bar-Shalom, Fred Daum, and Jim Huang. The Probabilistic Data Association Filter. *IEEE Control Systems Magazine*, 29(6):82–100, December 2009.

[6] Peter C Niedfeldt and Randal W Beard. Multiple Target Tracking using Recursive RANSAC.

[7] Frank Wood, Jan Gasthaus, Cédric Archambeau, Lancelot James, and Yee Whye Teh. The Sequence Memoizer. *Communications of the ACM*, 54(2):91, February 2011.

[8] Kevin Cook, Everett Bryan, Huili Yu, He Bai, Kevin Seppi, and Randal Beard. Intelligent Cooperative Control for Urban Tracking. *Journal of Intelligent & Robotic Systems*, 74(1-2):251–267, September 2013.

[9] Xuan Song, Jinshi Cui, Hongbin Zha, and Huijing Zhao. *Vision-Based Multiple Interacting Targets Tracking via On-Line Supervised Learning - Computer Vision ECCV 2008*, volume 5304 of *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, Berlin, Heidelberg, October 2008.

[10] Bo Yang, Chang Huang, and Ram Nevatia. Learning Affinities and Dependencies for Multi-Target Tracking Using a CRF Model. In *CVPR 2011*, pages 1233–1240. IEEE, June 2011.

[11] R. Nevatia. Multi-Target Tracking by Online Learning of Non-Linear Motion Patterns and Robust Appearance Models. In *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1918–1925. IEEE, June 2012.

[12] Dominic Schuhmacher, Ba-Tuong Vo, and Ba-Ngu Vo. A Consistent Metric for Performance Evaluation of Multi-Object Filters. *IEEE Transactions on Signal Processing*, 56(8):3447–3457, August 2008.

[13] B Ristic and D Clark. A Metric for Performance Evaluation of Multi-Target Tracking Algorithms. *IEEE Transactions on Signal Processing*, 59(7):3452–3457, July 2011.

[14] Keni Bernardin and Rainer Stiefelhagen. Evaluating Multiple Object Tracking Performance: The CLEAR MOT Metrics. *EURASIP Journal on Image and Video Processing*, 2008:1–10, February 2008.