2014-07-02

Recursive-RANSAC: A Novel Algorithm for Tracking Multiple Targets in Clutter

Peter C. Niedfeldt
Brigham Young University - Provo

Recursive-RANSAC: A Novel Algorithm for Tracking Multiple Targets in Clutter

Peter C. Niedfeldt

A dissertation submitted to the faculty of
Brigham Young University
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

Randal W. Beard, Chair
Timothy W. McLain
Dah J. Lee
Brian Mazzeo
Bryan S. Morse

Department of Electrical and Computer Engineering

Brigham Young University

July 2014

ABSTRACT

Recursive-RANSAC: A Novel Algorithm for
Tracking Multiple Targets in Clutter

Peter C. Niedfeldt
Department of Electrical and Computer Engineering, BYU
Doctor of Philosophy

Multiple target tracking (MTT) is the process of identifying the number of targets present in a surveillance region and the state estimates, or track, of each target. MTT remains a challenging problem due to the NP-hard data association step, where unlabeled measurements are identified as either a measurement of an existing target, a new target, or a spurious measurement called clutter. Existing techniques suffer from at least one of the following drawbacks: divergence in clutter, underlying assumptions on the number of targets, high computational complexity, time-consuming implementation, poor performance at low detection rates, and/or poor track continuity. Our goal is to develop an efficient MTT algorithm that is simple yet effective and that maintains track continuity enabling persistent tracking of an unknown number of targets.

A related field to tracking is regression analysis, where the parameters of static signals are estimated from a batch or a sequence of data. The random sample consensus (RANSAC) algorithm was developed to mitigate the effects of spurious measurements, and has since found wide application within the computer vision community due to its robustness and efficiency. The main concept of RANSAC is to form numerous simple hypotheses from a batch of data and identify the hypothesis with the most supporting measurements. Unfortunately, RANSAC is not designed to track multiple targets using sequential measurements.

To this end, we have developed the recursive-RANSAC (R-RANSAC) algorithm, which tracks multiple signals in clutter without requiring prior knowledge of the number of existing signals. The basic premise of the R-RANSAC algorithm is to store a set of RANSAC hypotheses between time steps. New measurements are used to either update existing hypotheses or generate new hypotheses using RANSAC. Storing multiple hypotheses enables R-RANSAC to track multiple targets. Good tracks are identified when a sufficient number of measurements support a hypothesis track. The complexity of R-RANSAC is shown to be squared in the number of measurements and stored tracks, and under moderate assumptions R-RANSAC converges in mean to the true states. We apply R-RANSAC to a variety of simulation, camera, and radar tracking examples.

Keywords: multiple target tracking, data association, Kalman filter, RANSAC, IMM, JPDA, MHT, PHD, MCMC data association, video surveillance, SAR

ACKNOWLEDGMENTS

I am grateful for the support of the U.S. government through the Department of Defense, Air Force Office of Scientific Research, National Defense Science and Engineering Graduate (NDSEG) Fellowship, 32 CFR 168a, which allowed me the opportunity to study target tracking. I am also grateful for the support of Dr. Randal Beard, whose tutelage and guidance enabled the development of new algorithms and, more importantly, my development as a student and researcher. He helped me push the envelope of my own understanding and experience, and I am better for it. Various professors helped through their teachings, including Dr. Michael Rice who derived the maximum likelihood equations in Appendix B.

Kyle Ingersoll deserves special recognition for his implementation and write-up of the R-RANSAC and IMM algorithms for use with the experiments regarding video-based tracking; he also provided key insights that helped enlarge our vision of the modularity of R-RANSAC. Also, a thank you to Eric Quist for providing the underlying simulated and real SAR measurement data for SAR feature tracking. Finally, thank you to all the other students with whom I have directly worked with on various PhD research projects: Patrick DeFranco, Justin MacKay, Jared Wikle, Matt Duffield, Josh Sakamaki, Brandon Carroll, Joel Howard, Dallin Briggs, Matt Argyle, Bryce Pincock, Jacob Bishop, and many others.

Many members of my family have been unsung heroes supporting me through these past several years. I am grateful for the examples of my parents, who taught me the importance of dedicated and efficient work ethic, and who believe that my siblings and I can do anything we set our minds to. I am grateful for the love and support of my wife Kristen, who was the most directly effected by this undertaking. She is always supportive of the efforts and time required for my research, she is always interested in my work—even the details, and she somehow found time to proofread my dissertation while being an excellent mother to our twin boys. I have been blessed through my faith in Christ, and know that through His influence I have been inspired with new ideas and have overcome obstacles in my path.

Table of Contents

List of Tables

List of Figures

Nomenclature

$\lvert\cdot\rvert$	Absolute value or set cardinality
$f()$	Parameter or state evolution function
$h()$	Measurement function
i	Index over the number of true signals and measurements
j	Index over the R-RANSAC hypotheses
k	Measurement scan index
k_N	Measurement scan index at the beginning of a window of length N
ℓ	Number of RANSAC iterations per time step
p	Probability of randomly selecting a true measurement
p_{D}	Probability of detection
p_{G}	Probability of a measurement inside a measurement gate
p_{S}	Probability of track survival
$\mathbf{q}$	Set of the s indices of measurements of within a minimum subset
$\mathbf{u}$	Control inputs
$\mathbf{v}$	Measurement noise
$\mathbf{w}$	Process noise
$\mathbf{x}$	Parameters or states of true signal
$\hat{x}$	Estimate of x
$\mathbf{y}$	Measurement of true signal
$\mathbf{z}$	Unlabeled measurement
$\mathrm{E}[x]$	Expected value of x
$\mathcal{K}_k$	Set of clutter measurements at time k
$\mathcal{L}$	R-RANSAC track label
M_k	Number of true signals at scan k
$\mathcal{M}$	Number of stored R-RANSAC tracks
N	Measurement window length
P	Parameter or state covariance
Q	Covariance of process noise
R	Covariance of measurement noise
$\mathcal{R}$	Surveillance region
$\mathrm{StD}[x]$	Standard deviation of x
$\mathcal{S}_{\mathbf{q}}$	Minimum set of measurements with indices $\mathbf{q}$
$\mathbb{S}^s_{k_N:k}$	Set of all possible minimum subsets of s measurements from scan k_N to k
V	Volume of surveillance region
$\mathrm{Var}[x]$	Variance of x
$\mathcal{Y}_k$	Set of all true measurements at time k
$\mathcal{Z}_k$	Set of all measurements at time k
χ	Consensus set
δ_k	Time step interval
γ	RANSAC early termination threshold
λ_{c}	Average number of clutter measurements per time step
$\mu()$	Measure of a compact set
ρ	Inlier ratio

τ_{R}	RANSAC error threshold
τ_{RR}	R-RANSAC error threshold
$\tau_{\mathbf{x}}$	Similar track threshold
τ_{ρ}	Good track threshold
τ_{T}	Minimum lifetime threshold
ψ_k	Number of measurements at time k
$\Psi_{k_N:k}$	Number of measurements from scans k_N to k

Abbreviations

CPHD	Cardenalized probability hypothesis density
EM	Expectation-maximization
ER	Execution rate
GM-PHD	Gaussian mixture probability hypothesis density
GNN	Global nearest neighbor
GPS	Global positioning system
HT	Hough transform
IE	Input estimation
IMM	Interacting multiple model
JPDA	Joint probabilistic data association
KF	Kalman filter
LS	Least-squares
MAGICC	Multiple agent intelligent coordination and control
MAP	Maximum a posteriori
MCMC	Markov chain Monte Carlo
MCMCDA	Markov chain Monte Carlo data association
MHT	Multiple hypothesis tracking
MLE	Maximum likelihood estimate
MTT	Multiple target tracking
NCA	Nearly-constant acceleration
NCJ	Nearly-constant jerk
NCV	Nearly-constant velocity
NN	Nearest neighbor
NNKF	Nearest neighbor Kalman filter
NP-hard	Non-deterministic polynomial-time hard
PDA	Probabilistic data association
PFR	Path fragmentation rate
PHD	Probability hypothesis density
PMHT	Probabilistic multiple hypothesis tracking
R-RANSAC	Recursive-random sample consensus
RANSAC	Random sample consensus
RFS	Random finite set
RLS	Recursive least-squares
RMSE	Root mean-squared error
SAR	Synthetic aperture radar
SMC-PHD	Sequential Monte Carlo probability hypothesis density
TFAR	Track false alarm rate
TFD	Track fragmentation duration
TFR	Track fragmentation rate
TPD	Track probability of detection
UAS	Unmanned air system
WSS	Wide-sense stationary

Chapter 1. Introduction

Since at least the mid-1960s, there has been continued and increasing interest in autonomous multiple target tracking (MTT) algorithms [146]. Some of the more obvious MTT applications are tactical in nature: surveillance [128], tracking pedestrians [5,134,159], missile defense [35,83], radar tracking [11], and sonar tracking [52,81]. However, an increasing number of diverse civilian MTT problems are now considered:

- Air traffic control [95]
- Collision avoidance [22, 29]
- Tracking audio signals [3, 31, 148]
- Tracking space debris [73]
- Sports tracking [71, 172]
- Simultaneous localization and mapping (SLAM) [108, 111]
- Wildlife and molecular tracking [15, 27, 171]
- Cloud detection and tracking [2]
- Traffic and vehicle tracking [118, 153, 162]

MTT generally consists of three main subsystems: sensor processing to identify the observations within the raw sensor data, data association or the process of determining the correspondence of measurements to target tracks, and filtering to remove the measurement noise inherent to sensors. The data association step is a challenging NP-hard assignment problem [68], where each measurement can be a measurement of an existing track, a new track, or a clutter measurement. A common, sub-optimal approximation is to split the data association into two steps, as denoted in Fig. 1.1, where the number of tracks are identified in a track management algorithm, and a measurement weighting is computed based on the estimated tracks.

Over the years, many algorithms have been developed to solve the MTT problem, which will be enumerated and described in detail in Section 1.1. We note that some of these algorithms are even optimal in theory, provided infinite computational resources. In practice, however, approximations must be included to maintain feasibility—especially for

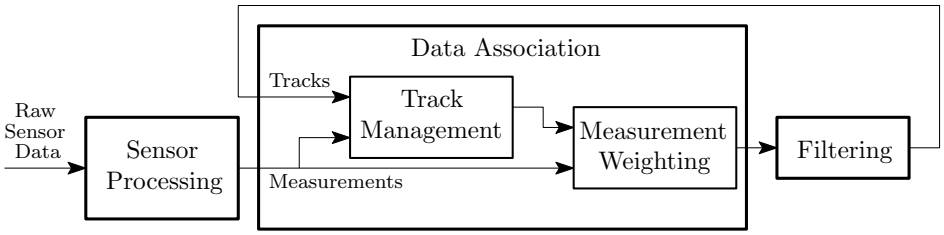

Figure 1.1: High-level architecture of an autonomous MTT system.

real-time tracking applications. Some of the design decisions and trade-offs that should be considered are how to manage the computational complexity, the complexity of the coding process, and tracker performance.

At Brigham Young University's Multiple AGent Intelligent Coordination and Control (MAGICC) Lab, we frequently encounter scenarios where robust, autonomous tracking of static or maneuvering targets is essential for mission success. One particular example that motivated our exploration of MTT algorithms was when the lab visually tracked a moving ground vehicle from an aerial platform. The project scope was specifically to design a controller that maneuvered a fixed-wing aircraft with dynamic constraints to maintain an operator-selected target near the center of the field of view. However, the underlying control algorithms required perfect data association. A video obtained during the testing phase of the project demonstrates the need of robust, autonomous, and real-time tracking algorithms. A tracking algorithm (e.g. a computer vision template matching algorithm) was used to track the target of interest within the image, in this case, a truck; track initiation was performed by an operator clicking on the truck in the real-time video feed in the user interface. The tracker performance was very good until the white truck drove over a white manhole in the middle of the street. The tracker, confused from two sources of measurements, chose one of the two signals and incorrectly locked onto the manhole, requiring the operator to reinitialize the track by again clicking on the truck.

Most importantly, we realized that an ideal tracking algorithm should not require an operator to have to babysit the system in the presence of clutter and other spurious measurements. In this case, there was actually a second signal, the white manhole, present in the data. In such cases, an ideal tracker should track both signals and distinguish between crossing and/or distracter signals to maintain track continuity. The ability to track

unexpected signals in addition to user-selected targets also requires some method of track initiation.

From our experience in computer vision, we recalled that the random sample consensus (RANSAC) algorithm [53] is a robust method to estimate the parameters of a signal in the presence of large amounts of clutter. The premise of RANSAC is to form numerous simple hypotheses to estimate the true parameters, where the best hypothesis is selected according to some cost function, such as maximizing the number of supporting measurements. RANSAC has found wide support, especially in the computer vision community where it is used to identify matching features between stereo images [158], identify shapes in images [140], and many other applications.

Unfortunately, RANSAC is a batch algorithm and can only estimate the parameters of a single signal. The fundamental contribution of this dissertation is to extend RANSAC by storing a set of RANSAC hypotheses between time steps and recursively updating them. This algorithm is called the recursive-RANSAC (R-RANSAC) algorithm. In the case of data with only a single signal, the hypothesis with the most inliers is identified as the correct hypothesis. This is essentially a maximum a posteriori (MAP) approximation of the true data association. In addition, by storing multiple hypotheses, R-RANSAC is inherently capable of estimating the parameters of multiple signals. Good signals can be identified when the hypothesis support achieves a particular threshold.

The R-RANSAC algorithm joins a large body of work within the MTT community. Some of the advantages of the R-RANSAC algorithm over existing trackers include: ease of implementation; reasonable computational complexity proportional to the number of measurements and stored models, squared; good track continuity while rejecting clutter; and a modular framework to incorporate a variety of existing techniques such as nonlinear filtering, multiple model filtering [127], and the method of assigning data association measurement weights. The full extent of contributions with respect to the R-RANSAC algorithm are summarized in Section 1.2.

1.1 Literature Review

We divide our survey of the literature into two main sections. First, we review the RANSAC algorithm and a few of the variants that address multiple signals in the data set. Second, we summarize many of the MTT algorithms and discuss their strengths, weaknesses, and relation to R-RANSAC.

1.1.1 Random Sample Consensus (RANSAC)

RANSAC was developed in the early 1980s as a new paradigm to regression analysis given a batch of data [53]. Traditional regression techniques estimate the parameters using all or most of the available data set, and then remove observations inconsistent with the hypothesis before producing a final estimate. Alternatively, RANSAC forms numerous simple hypotheses, using a minimum subset of the full data set, and computes the support of each hypothesis, where the hypothesis support refers to the number of measurements, or *inliers*, whose residual is less than a specified threshold. The hypothesis with the greatest support is selected and smoothed using the inlier set, allowing RANSAC to quickly and accurately estimate the parameters of an underlying signal in clutter.

While we have described the standard implementation of RANSAC, we note that there are more advanced techniques to improve the accuracy, efficiency, or robustness of RANSAC. For a recent survey of the many variations of RANSAC, see [30]. One technique that could be especially useful to improve execution rates is guided sampling, where statistical information of the measurements is used to influence the otherwise random selection of data points within the minimum subsets that are used to generate hypotheses [156, 157].

The RANSAC algorithm was originally designed to estimate the parameters of a single signal. When there is more than one signal within the batch of data, RANSAC returns only one parameter set. The estimation process is also more difficult since additional signals act as clustered outliers. An intuitive approach when using RANSAC to estimate multiple signals in a batch of data is to find one RANSAC estimate, remove the inliers of that hypothesis from the data set, and repeat until no more hypotheses with high support can be found; this technique is known as sequential RANSAC [82, 164]. However, sequential

RANSAC is not robust, as an incorrect estimate of the first (or any subsequent) hypothesis causes the remaining hypotheses to also be inaccurate [179]. We note that some papers in the literature, e.g. [4,16,46], rename sequential-RANSAC as recursive RANSAC, an unfortunate misnomer since sequential-RANSAC relies on iteration within a time step rather than a temporal Bayesian recursion. This is an important distinction of the R-RANSAC algorithm discussed in this dissertation.

The multiRANSAC algorithm was developed to robustly estimate multiple signals in a data set, though the number of signals must be known a priori [179]. An alternative technique was developed in [178], where the residual of each measurement to each hypothesis is used to infer the number of underlying signals. Unfortunately, the multiple signal estimation algorithm in [178] is a batch algorithm, and does not appear to naturally extend to a recursive framework.

While not based on RANSAC, per se, other algorithms have been developed to estimate the parameters of multiple signals using randomized sampling. One algorithm is the randomized Hough transform (RHT) [173, 174], which is an extension of the Hough transform (HT) [72]. RHT uses a random selection of measurements to build the Hough histogram matrix in order to reduce the computational burden of exhaustively accounting for all measurements. While it is more efficient than the HT, RHT still suffers from inaccuracies due to operating in a discrete state space. RHT also remains computationally intensive and performs poorly in a high percentage of outliers [154]. Instead, the authors of [154] propose a technique called J-Linkage, which identifies the clusters of data within a measurement-hypothesis matrix using the Jaccard distance. Finally, the statistical learning algorithm proposed in [28] removes the need to provide parameters to J-Linkage by defining a Mercer kernel, which can be used with statistical learning concepts to estimate both the number of signals in the data and their parameters. These are all batch algorithms, and an extension to a recursive algorithm is not readily apparent. However, future studies should be conducted to examine efficient recursive estimation using these techniques.

Thus far, we have only discussed variations of RANSAC that have been used to estimate the parameters of one or more static signals. The remainder of this section briefly de-

scribes two RANSAC-based algorithms designed specifically for target tracking: RAMOSAC and KALMANSAC.

Random Model and Sample Consensus (RAMOSAC)

The random model and sample consensus (RAMOSAC) algorithm was developed to address deficiencies in visual tracking [149]. Specifically, when tracking targets via matched features, there are instances when the number of features is so low that accurate feature matching is unlikely. RAMOSAC extends the original RANSAC algorithm by randomly selecting a motion, or dynamic, model before selecting a random minimum subset to generate a new hypothesis [149]. By allowing for multiple motion models of different complexities, RAMOSAC can accommodate a varying number of matched features. RAMOSAC is purely a vision-based tracking algorithm that does not incorporate a dynamic model. While multiple objects are tracked, methods for track initiation are not discussed and it is assumed that manual selection is required.

KALMANSAC

KALMANSAC is a RANSAC-based algorithm that robustly tracks a moving target with linear-Gaussian dynamics [161]. The premise of KALMANSAC is to use the RANSAC paradigm to find the data association resulting in the MAP solution. KALMANSAC first selects a random minimum subset of measurements to estimate the most likely state estimate, which is subsequently used to estimate the most likely inlier/outlier measurement labeling over the measurement set. These steps are iterated until a stopping criterion is satisfied. The resulting measurement labeling is used to seed the iteration of subsequent time steps. Since KALMANSAC computes the MAP estimate, the underlying measurement distribution, excluding the outliers, must be unimodal. However, this assumption does not hold when there exist multiple targets since there is no mechanism for data association between targets. When tracking a single target, we expect R-RANSAC and KALMANSAC to converge to the same solution.

1.1.2 Multiple Target Tracking

Many of the existing MTT algorithms can be categorized into three groups based on the underlying optimization of the data association objective function [129]. The first is the use of heuristics such as the nearest neighbor (NN) filter [12]. The remaining two classifications are both Bayesian techniques: Bayes estimator approaches such as joint probabilistic data association (JPDA) [56], and MAP approaches such as multiple hypothesis tracking (MHT) [132]. The NN, MHT, and JPDA algorithms are three of the original algorithms developed during the early stages of the autonomous tracking research. The remainder of this section reviews several of the existing MTT algorithms and their relation to the recently developed R-RANSAC algorithm. Given the breadth of research, and the many variations of specific algorithms, the following survey is by no means exhaustive. For an extended review of these and other MTT algorithms, the interested reader is referred to the surveys in [10, 38, 129, 175].

Before describing specific algorithms, we briefly discuss some general MTT terms and techniques. First, a target track is defined as a sequence of estimated states describing a single target over several time steps. The second is a method to reduce the computational complexity of the NP-hard data association, which is to reduce the possible measurement-to-track combinations by assuming that distant measurements are so unlikely that they can be ignored. This technique is called gating. Typically, gating is performed using a statistical gate, where all measurements outside of a certain Mahalanobis distance of a track are assumed to be unassociated with that track [19].

Nearest Neighbor Algorithms

The simplest of the MTT algorithms rely on heuristics to estimate the unknown data association. At each scan NN algorithms rely on finding a greedy assignment of measurements to existing tracks. The NN Kalman filter (NNKF) [12] propagates each target track using the measurement with nearest statistical distance. Note that in this case it is possible for a single measurement to be used to update two nearby tracks, which would lead to track coalescence. The global NN (GNN) algorithm [19] is also a greedy assignment,

but enforces the constraint that one measurement can only be assigned to one target by solving a 2D-assignment problem, e.g. the Munkres or the Hungarian algorithm [112]. Both NNKF and GNN require prior knowledge of the number of targets, requiring a separate track management algorithm.

It is known that the NN algorithm is not robust to clutter. One method to reduce, but not eliminate, the effects of clutter is to consider the assignment over multiple scans. Oh et al. developed a greedy batch-mode algorithm called the multi-scan NN filter to not only estimate the correspondence between measurements and tracks, but also to perform track management by detecting new tracks [120]. The basic idea of the multi-scan NN filter is that all measurements are initially marked as false alarms. Using two unmarked measurements, a constant velocity model is used to estimate the initial states of a tentative track. The NN algorithm is used to propagate the states forward in time, and the track is accepted if the support of the track exceeds a threshold. The associated measurements are marked, and the process is repeated until no more tracks can be found. In essence, this algorithm is related to the sequential-RANSAC algorithm [82, 164] and is expected to suffer similar problems of robustness as described in [179], which is confirmed in [122].

Probabilistic Data Association Algorithms

One of the first Bayesian approaches to MTT is the probabilistic data association (PDA) filter [13]. The PDA filter weights nearby measurements based on their statistical distance and the probability of false alarms and new targets. The weighted measurements are then combined into a single measurement which is used in a modified Kalman filter, where the only modification is an additional term in the covariance propagation to account for the uncertain data association [13, 19]. The PDA (and variants of the PDA) filter are sub-optimal approximations of the optimal Bayesian estimator. Under the assumption that the previous prior distribution is exact, then only the current set of measurements are needed to update the distribution [11].

The PDA filter is well-suited for tracking a single target in clutter. However, when there are multiple targets that are not well-separated, the performance of the PDA filter degrades. The joint PDA (JPDA) filter was developed to explicitly account for shared measure-

ments. The only algorithmic difference between JPDA and PDA is that JPDA computes the measurement weighting differently, accounting for the joint probability that a measurement could be associated with multiple tracks [56]. Unfortunately, computing the joint association probabilities between multiple shared tracks in JPDA is more computationally complex than PDA. Several variations of the JPDA filter that reduce the computational complexity through approximations include: cheap JPDA [54], suboptimal JPDA [137], near-optimal JPDA [136], single-scan algorithm with leave-one-out heuristic [74], and maximum-likelihood PDA [20]. One important assumption of both the PDA and JPDA algorithms and most of their variants is that they assume that the number of targets is known [19], and require a separate track management subsystem to perform track initiation and deletion when that assumption is not true.

Multiple Hypothesis Tracking Algorithms

The last of what we term classical MTT algorithms is the multiple hypothesis tracking (MHT) filter, originally proposed in [132]. An excellent tutorial of the MHT filter can be found in [17]. In summary, MHT estimates the correspondence between measurements and tracks by exhaustively enumerating all possible correspondences and identifying the MAP hypothesis over multiple measurement scans. One distinct advantage of the MHT filter over the JPDA and NN filters is that data association is delayed for several measurements. This allows subsequent scans to influence the hard data association for previous data steps. Another advantage over the previous methods is that MHT naturally accounts for track initiation and deletion, without requiring a separate track management algorithm. This allows MHT to be optimal in theory with infinite computational resources [17]. Unfortunately, the computational complexity of MHT is orders of magnitude larger than JPDA, and coding MHT is known to be particularly daunting [17]. There exist two versions of MHT that differ in the manner of hypothesis generation: the measurement-oriented approach [132] and the track-oriented [44, 96] approach.

In the measurement-oriented approach, a set of hypotheses are maintained that enumerate all possible measurement associations. In other words, at the reception of a new scan of measurements, each measurement can be labeled as a measurement of an existing target, a

new target or a false alarm. Each hypothesis defines a set of tracks, where no measurements are shared between tracks. The exponential number of hypotheses required to exhaustively describe the possible data associations requires a pruning and merging step to reduce the number of hypotheses, which reduces the optimality of the algorithm.

In the track-oriented approach, instead of propagating hypotheses with all possible measurement-to-track association combinations, only a set of tracks are propagated between time steps. At the beginning of each time step, the existing tracks are reformed into hypotheses and updated with new measurements. Kurien describes these structures as target trees, where the root of the tree represents the previous estimate of the track, and subsequent leaves denote the different measurement assignment possibilities, where each level of the tree represents a different time step [96]. Again, a sub-optimal pruning step is used to reduce the number of track hypotheses. It has been observed that the measurement-oriented approach typically results in several thousand hypotheses, while the track-oriented approach results in a few hundred possible tracks [17]. We note that real-time performance of MHT for modestly complicated scenarios is possible [11, 18]. A common approach to track-oriented MHT, developed by Cox et al. [39, 40], is to apply Murty's algorithm [113] to identify the most-likely assignments without explicitly generating each hypothesis. Using this approach, the average computational complexity can be reduced to $\mathcal{O}(N^3)$ on average, where N is the number of measurements or tracks [40]. The worst-case computational complexity is $\mathcal{O}(N^4)$. A variant of the track-oriented approach utilizes the cheap-JPDA method [54] to eliminate some of the less-likely tracks before forming new hypotheses based on the new measurements [130].

We note that R-RANSAC is related to the track-oriented MHT algorithm in that it also finds the MAP hypothesis. However, instead of exhaustively enumerating over all assignments, only the most likely tracks are stored.

A variant of the MHT algorithm is the probabilistic MHT (PMHT) filter [150, 151], which is a Bayesian estimator. The PMHT filter is a batch algorithm that uses all measurements from a window of recent measurement scans to update each track, where measurements are individually weighted and subsequently combined into a single pseudo-measurement for a Kalman update; note that multiple measurements are allowed to be assigned to a single target. The PMHT filter removes the need for a pruning algorithm while reducing the com-

binatorial complexity to be linearly dependent on the number of measurements, tracks, and clutter measurements. The trade-off is that PMHT requires prior knowledge of the number of tracks and also utilizes the expectation-maximization (EM) algorithm [45], which requires multiple passes to converge to the solution and can be slow; we note that in one study [50] PMHT was slower than the MHT implementation in [130]. Originally, PMHT did not model clutter; however, several subsequent ad hoc techniques and a general formulation to account for clutter are summarized in [41]. An excellent review of the PMHT algorithm and several of its variants can be found in [41, 170]. The application of the EM algorithm is reminiscent of the RANSAC algorithm, but we note that EM utilizes the full data set, whereas RANSAC uses minimum subsets to estimate data association and is much faster.

Markov Chain Monte Carlo Data Association Algorithm

The Markov chain Monte Carlo data association (MCMCDA) algorithm, introduced by Oh et al., is an approximation scheme for the optimal Bayesian filter [121, 122], and is shown to converge to the Bayesian solution given unlimited resources. MCMCDA extends the work of other Markov chain Monte Carlo (MCMC) tracking techniques [33, 125] by incorporating missed detections, clutter, and track management [122]. The basic idea of the online-MCMCDA algorithm is to store a window of measurements and the prior association from the previous time step. For each measurement in the current scan, MCMCDA randomly selects one of several possible association events over multiple iterations, where the most likely event is stored. Some of the possible association events include track birth and death, updating tracks, splitting and merging tracks, etc. The authors state that the random sampling of the data association is related to simulated annealing at a constant temperature [122]. In order to make the MCMC sampling more efficient, MCMCDA assumes that targets have a known maximum velocity and that targets will not have more than a specified number of missed detections. In [121, 122], it was shown for a dense tracking scenario with 380 targets, probability of detection of 0.7, and an average clutter rate of 5 false alarms per time step that MCMCDA is more accurate than MHT while also running 20 times faster.

This algorithm is the most similar to R-RANSAC, since R-RANSAC stochastically samples trajectories from the same association space. The main difference is R-RANSAC's

application of the RANSAC minimum subset paradigm, where only a few points are used to generate a hypothesis track. As a result, MCMCDA is order $O(N^2 \log N)$ in the number of measurements [122], whereas R-RANSAC is of order $O(N^2)$ and is also simpler to code.

Particle Filter Algorithms

Another MTT algorithm that has seen increased attention is the particle filter. Initially, particle filters were developed for single target tracking of nonlinear, non-Gaussian systems in clutter [7]. Subsequent implementations account for the unknown data association of measurements to a fixed, known number of tracks [77, 142]. In [76], the authors extend their previous work to robustly identify track deletions, but acknowledge that the track initiation problem remained difficult. Another drawback is the large number of particles required to estimate the states of multiple targets. The Rao-Blackwellized particle filter helps to reduce the number of particles by splitting the problem into two parts: the data association problem, and the filtering of each target conditioned on the data association [139]; once again, the number of targets must be known a priori. In [114], the authors present a technique that deterministically detects the existence of a new target within a sliding window of measurements, reminiscent of a standard M/N detector [19], and increases the number of tracked targets based on a probability that depends on the number of consecutive detections of the tentative target. In [79], the authors propose using two layers of particle filtering, where the parent layer estimates the number of targets and the child layer estimates the target states conditioned on the number of targets. Unanswered questions about particle filters include how many particles are required for a given level of accuracy, and the dependence of the number of particles to the dimension of the state vector [43]. Two papers with an excellent survey of particle filtering techniques are [114, 163].

Probabilistic Hypothesis Density Algorithms

The probability hypothesis density (PHD) filter was developed to address several difficult challenges of MTT, such as tracking in high target density environments, where group or cluster tracking is required due to closely spaced targets [105]. General MTT

requires not only the estimation of the target states but also an estimation of how many targets are present in the data set, which is ill-suited for vector notation. To solve this problem, a Bayesian estimator called the PHD filter is proposed by Mahler that reformulates the MTT problem using random finite sets (RFS) [101, 102]. An RFS is a set of finite dimensional random variables; in other words, there is a discrete probability distribution on the cardinality of the set, and for each cardinality there is a distribution for each of the elements in the set [166]. An important property of an RFS is that the integral over a subset of the posterior is equal to the expected number of targets within that region [101]. Using RFS, a Bayesian recursion is developed using finite set statistics where the states of the unknown number of targets are treated as an RFS, and the variable number of measurements received per scan are also treated as an RFS [101, 102].

As with other general Bayesian filters, the RFS Bayesian recursion is intractable. In an effort to reduce some of the computational complexity, only the first order moment of the RFS, or intensity, is propagated to the next time step. This intensity is defined as the PHD. One underlying assumption of the PHD filter is that the underlying RFS are Poisson, which are completely characterized by their mean [166].

Unfortunately, the Bayes recursion on the RFS posterior intensity, or PHD, is still intractable. The first approach to develop a feasible estimate of the PHD filter was to use sequential Monte Carlo (SMC) techniques to sample the underlying PHD distribution. Various researchers independently proposed this technique, called SMC-PHD in [144, 167, 176]. The SMC remains useful because there are no restrictions on target dynamics or noise characteristics, which can be nonlinear and non-Gaussian, respectively. Later, Vo et al. developed a closed-form recursion for the PHD under the assumption of linear Gaussian systems, called the Gaussian mixture PHD (GM-PHD) [165, 166].

Some of the advantages of the PHD filter, listed in [105], include the following. First, the PHD filter employs an implicit data association accomplished via an all neighbors update, where every measurement updates every track according to a likelihood function. This first advantage leads to the second, which is that the PHD filter is computationally efficient. When prior information is known on the birth distribution of new targets, the PHD is linear in the number of measurements and stored tracks; when prior information is unknown, the

previous set of measurements can be used to seed the next set of measurements, resulting in a complexity that squares with the number of measurements. Finally, the PHD filter is able to directly estimate the number of targets present in the data.

Unfortunately, the PHD filter does not accurately estimate the number of targets when the probability of detecting true targets is low [51]. This reduces the ability of the PHD to maintain track continuity over long periods of time, since a single missed detection causes track loss, and multiple missed detections may cause the track to be pruned. In the presence of clutter, the PHD is also known to have a large variance on the estimated number of targets. For this reason, Mahler subsequently introduced the cardenalized PHD (CPHD) filter [104], which propagates a belief of the number of targets between time steps as well as the intensity distribution. This comes at the expense of computational complexity, which becomes cubic in the number of measurements, and increases the delay before detecting new targets. A survey of various PHD and CPHD algorithms can be found in [103], though many new applications can be found in the literature since its publication.

Other MTT algorithms

While we have attempted to cover a wide range of tracking algorithms, there still remain at least two classifications that we have not discussed. For example, one class of algorithms described in [129] estimates the data association by mapping the measurements onto a trellis and solving an optimization problem. Another class of algorithms that we have not explored are computer vision algorithms, such as feature tracking [100, 155], template matching [8], mean-shift [6, 26], and others [126, 175]. Instead of relying on a dynamic model, computer vision algorithms utilize visual properties such as edges, colors, etc. to track multiple targets. One advantage of computer vision algorithms is that by relying on visual cues, tracking can be more robust to unmodeled dynamics, such as turbulence while an aerial vehicle tracks ground targets. However, they are more susceptible to environmental changes (e.g. sunlight variation) that negatively affect the underlying detection algorithms.

1.2 Summary of Contributions

The main contribution in this dissertation is the development of a novel MTT algorithm called recursive-RANSAC (R-RANSAC). The R-RANSAC algorithm extends the traditional RANSAC algorithm to recursively estimate the parameters/states of multiple static/dynamic signals in clutter. The premise of R-RANSAC is to store multiple hypotheses between time steps and to identify the best hypotheses, which are used to estimate the underlying signals. New measurements are used to either update existing hypotheses or to generate new hypotheses from a sliding window of measurements. Specific contributions of this dissertation include the following:

- A general formulation of the R-RANSAC algorithm that is applicable to both static parameter estimation and dynamic multiple target tracking.
- The derivation showing the computational complexity of R-RANSAC to be proportional to the number of new measurements and stored tracks, squared.
- An analysis of the R-RANSAC parameter sensitivity with regards to standard MTT metrics.
- A theoretical analysis demonstrating that R-RANSAC converges in mean to the true states when tracking well-separated targets.
- The derivation of novel RANSAC techniques for trajectory estimation of both maneuvering and non-maneuvering targets.
- An extensive comparison of R-RANSAC with five existing MTT algorithms: the GNN, JPDA, MHT, MCMCDA, and GM-PHD filters. We find that R-RANSAC has excellent track continuity, low error estimates, and better than average execution rates compared to existing algorithms.

In short, we find that the R-RANSAC algorithm is well-suited for applications where robust autonomous tracking is desired to persistently track targets both spatially and temporally.

1.3 Current R-RANSAC Publications

Some of the research discussed in this dissertation has been previously published. The R-RANSAC algorithm for estimating the parameters of static signals in clutter was originally presented in [115]. R-RANSAC was used to estimate the range to static ground point scatterers using a synthetic aperture radar in [117]. The R-RANSAC algorithm was later extended in [116] to track multiple dynamic targets in clutter. We have also published a comparison of R-RANSAC to the CPHD [52,168] filter when tracking objects from a moving aerial platform in [64].

1.4 Dissertation Outline

The outline of the remainder of this dissertation is summarized as follows. Chapter 2 presents the general formulation of the R-RANSAC algorithm; we also analyze both the computational complexity and parameter sensitivity of R-RANSAC. In Chapter 3, we apply the R-RANSAC algorithm to estimate the parameters of multiple static signals, where we use the recursive least-squares filter [110] to update the estimated parameters. In Chapter 4, we apply the static version of R-RANSAC to numerous simulated and experimental scenarios. In Chapter 5, we formulate the R-RANSAC algorithm to estimate the states of multiple dynamic signals, where we use the Kalman filter to update the target states. Chapter 5 is divided into four main sections. First, we assume that the targets follow a known dynamic model with zero-mean process noise. Second, we account for cooperative targets and include the known control inputs in the R-RANSAC estimation scheme. Third, we discuss the tracking of maneuvering targets. Specifically, we develop a RANSAC-based maximum likelihood estimation technique to estimate the control inputs from the measurements in the sliding window and we also propose the interacting multiple model variant of R-RANSAC to track maneuvering targets. Fourth, we show that the R-RANSAC estimates converge in mean to the true states of well-separated targets and that the expected hypothesis support of the R-RANSAC hypotheses converges as the measurement window size increases. Chapter 6 discusses four simulated and real applications when applying R-RANSAC to MTT scenarios. Chapter 7 presents our conclusions and future avenues of research.

Chapter 2. Recursive-RANSAC Framework

One of the advantages of the recursive-RANSAC (R-RANSAC) algorithm is its modularity and applicability to a wide range of tracking scenarios. We devote this chapter to introducing the general R-RANSAC framework that can be tailored to suit specific problems, as will be done in subsequent chapters. First, we describe some notation and explicitly state the class of problems that we are interested in by listing assumptions. We then review the traditional RANSAC algorithm and its parameters. The duration of this chapter is then used to introduce the novel R-RANSAC algorithm for a general class of systems, discuss the resulting computational complexity of R-RANSAC, and conduct an R-RANSAC parameter sensitivity analysis with respect to standard tracking metrics.

2.1 General Notation and Assumptions

Define the surveillance region as a time-varying compact subset of the measurement space $\mathcal{R}_k \subset \mathbb{R}^m$ at time step k. The surveillance region has a known, finite volume given by $V_k = \mu(\mathcal{R}_k)$, where the function μ defines the measure of the set. Classical examples of a surveillance region include the field of view of camera or radar systems. At time step k, there are M_k true signals within the surveillance region. The parameters, or states, of each target are defined as $\mathbf{x}_k^i \in \mathbb{R}^n$, where the states evolve according to

$$\mathbf{x}_k^i = f(\mathbf{x}_{k-1}^i), \tag{2.1}$$

where $f : \mathbb{R}^n \mapsto \mathbb{R}^n$; note that this framework allows for both static and dynamic states. Our objective is to estimate the number of true signals and the current states $\mathbf{x}_k^i$ of each signal.

At each time step k, sensor processing results in observations that directly or indirectly allow estimation of each target's states. The probability that the i^{th} target is measured is modeled by the random process $\mathcal{P}_k^i(\omega)$, where $\omega \in \Omega = \{0, 1\}$ and $\omega = 1$ signifies a single, valid measurement of the true signal. Mathematically, the output $\mathbf{y}_k^i \in \mathbb{R}^m$ is described by

$$\mathbf{y}_k^i = \begin{cases} h(\mathbf{x}_k^i) + \mathbf{v}_k & \text{when } \omega = 1 \\ \emptyset & \text{when } \omega = 0, \end{cases} \tag{2.2}$$

where $h : \mathbb{R}^n \mapsto \mathbb{R}^m$ is the measurement function, the measurement noise $\mathbf{v}_k \in \mathbb{R}^m$ is a wide-sense stationary (WSS), zero-mean Gaussian random process with covariance matrix R, and $\emptyset$ denotes that no measurements of the target were received at that time step. The probability distribution on the events in $\mathcal{P}_k^i(\omega)$ is problem specific.

We model the probability of a target existing from time step to time step as a Bernoulli random variable with the parameter p_{S} defined as the probability of target survival. The probability of target survival accounts for targets that persist for a finite time, such as when they either leave the surveillance region or no longer become targets; e.g. a vehicle after parking or a pedestrian after entering a building.

Throughout this dissertation we use the notation $i : j$ to denote the set $\{i, i+1, i+2, \ldots, j-1, j\}$. The R-RANSAC algorithm operates over sliding window of N time steps. To simplify notation we define $k_N \triangleq k - N + 1$, and use the notation $k_N : k$ to denote the past N-length window of time samples. We allow for multiple measurements per time step to account for the multiple targets and occasional spurious measurements or clutter. Let $\mathcal{Y}_{k_N:k}^i = \bigcup_{\kappa=k_N}^{k} \{\mathbf{y}_\kappa^i\}$ denote the set of windowed measurements of the i^{th} target, and let $\mathcal{Y}_{k_N:k} = \bigcup_{i=1}^{M_k} \mathcal{Y}_{k_N:k}^i$ denote the windowed set of all true measurements. Let $\mathcal{K}_{k_N:k}$ be the windowed set of clutter measurements, where the distribution of the random set of clutter $\mathcal{K}_k$ is problem dependent [166]. Borrowing terminology from the radar community, a sensor *scan* $\mathcal{Z}_k$ is the set of measurements received after processing the sensor data; i.e. $\mathcal{Z}_k = \mathcal{Y}_k \bigcup \mathcal{K}_k$. Letting $|\cdot|$ denote the cardinality of a finite set, the number of measurements in scan k is $\psi_k \triangleq |\mathcal{Z}_k| = |\mathcal{Y}_k| + |\mathcal{K}_k|$. There is no labeling on the unordered measurements. The total number of measurements after k scans is $\Psi_{k_N:k} = |\mathcal{Z}_{k_N:k}|$.

An important step of multiple target tracking (MTT) is to identify the subset of true measurements $\mathcal{Y}^i_{1:k} \subset \mathcal{Z}_{1:k}$ associated with the i^{th} target, a process known as data association. In a Bayesian context, we estimate the states $\mathbf{x}^i_k$ of the i^{th} target by solving for the density $\mathcal{P}(\mathbf{x}^i_k \mid \mathbf{x}^i_{0:k-1}, \mathcal{Y}^i_{1:k})$ after estimating the correct association $\mathcal{Y}^i_{1:k}$. This dissertation focuses on finding an estimate of the true data association over a sliding window, $\hat{\mathcal{Y}}^i_{k_N:k} \approx \mathcal{Y}^i_{k_N:k}$, which we use to estimate the trajectory of the i^{th} target $\hat{\mathbf{x}}^i_{0:k} \approx \mathbf{x}^i_{0:k}$. Throughout the dissertation, the notation $\hat{x}$ denotes the estimate of x.

In this dissertation we examine a subset of the generalized MTT problem by making the following assumptions, which are common in the MTT literature [14, 19, 122, 166].

Assumption 2.1: Target states, target measurements, and clutter are statistically independent.

Assumption 2.2: At each time step, at most one measurement per target is detected, where $\mathbf{y}^i_k \in \mathcal{R}_k$.

Assumption 2.3: The random process describing the clutter $\mathcal{K}$ is modeled in two parts: the number of clutter measurements is Poisson with parameter λ_{c} characterizing the average number of clutter measurements per time step, and the distribution of clutter over $\mathcal{R}_k$ is uniform.

Assumption 2.4: The random process $\mathcal{P}^i_k(\omega)$ governing the measurements of the i^{th} target is modeled as a Bernoulli process that is state independent, where the probability of detection $p^i_{\text{D}} = p_{\text{D}} \in (0, 1],\ \forall i$.

Assumption 2.5: The system states are observable from the measurements.

The following lemma gives the probability of selecting a measurement of the i^{th} target at random from the measurement window $\mathcal{Z}_{k_N:k}$.

Lemma 1 *Under Assumptions 2.1–2.4, if the expected number of targets is $M = \text{E}[M_k]$, then the expected value of the probability of selecting a measurement of the i^{th} target, p^i, at random from the measurement window $\mathcal{Z}_{k_N:k}$ is*

$$p^i = \frac{p_{\text{D}}}{\lambda_{\text{c}} + M p_{\text{D}}}, \tag{2.3}$$

Proof: The probability of selecting a measurement of the i^{th} target at random from the measurement window is given by

$$p^i = \mathrm{E}\left[\frac{|\mathcal{Y}^i_{k_N:k}|}{|\mathcal{Z}_{k_N:k}|}\right].$$

By Assumption 2.1 we have

$$p^i = \frac{\mathrm{E}\big[|\mathcal{Y}^i_{k_N:k}|\big]}{\mathrm{E}[|\mathcal{K}_{k_N:k}|] + \mathrm{E}\big[|\mathcal{Y}_{k_N:k}|\big]}.$$

Using Assumptions 2.2–2.4 gives

$$p^i = \frac{N\, p_{\mathrm{D}}}{N\, \lambda_{\mathrm{c}} + N\, M_k\, p_{\mathrm{D}}}.$$

Equation (2.3) follows when $M = \mathrm{E}[M_k]$. ■

In Chapters 3 and 4, we consider specific problems when the signal states $\mathbf{x}$ are the static parameters that characterize a linear input-output mapping, such as in regression analysis. In Chapters 5 and 6, the states $\mathbf{x}$ are time varying, such as the position and velocity measurements of a maneuvering target. The specific measurement and state equations will be defined formally in subsequent chapters.

2.2 Random Sample Consensus (RANSAC) Algorithm

The traditional RANSAC algorithm is a regression technique that estimates the parameters of a single signal from a batch of measurements, where the measurements may be corrupted by a large number of spurious measurements. The RANSAC algorithm consists of two repeated steps: hypothesis generation and hypothesis validation. Essentially, RANSAC estimates the data association to find $\hat{\mathcal{Y}}^i_{k_N:k} \subset \mathcal{Z}_{k_N:k}$ by forming many hypotheses using *minimum subsets* of the N scans of measurements and identifying the best hypothesis. The final estimate can then be found by smoothing over the estimated data association.

The novelty of the traditional RANSAC algorithm is the use of minimum subsets to generate many simple hypotheses of the underlying signal. Define a minimum subset

of s measurements as $\mathcal{S} \in \mathbb{S}^s_{k_N:k}$, where $\mathbb{S}^s_{k_N:k}$ is the set of $\binom{\Psi_{k_N:k}}{s} = \frac{\Psi_{k_N:k}!}{s!(\Psi_{k_N:k}-s)!}$ possible combinations of s measurements in the measurement window $\mathcal{Z}_{k_N:k}$. The cardinality s of the minimum subsets of is given by the minimum number of measurements required to form an estimate of the true signal states based on observability principles. During the hypothesis validation step, RANSAC can quickly identify the hypothesis that best characterizes the full data set. In fact, RANSAC can successfully converge to the true parameters when the probability of successfully selecting a good measurement p is very low; some implementations report success even when $0.15 \leq p \leq 0.2$ [140, 161].

During hypothesis validation, a metric is needed to characterize the support of each hypothesis. While other metrics exist [30], a common metric is to count the number of measurements supporting each hypothesis. The set of supporting measurements is called the consensus set χ and approximates the true data association, $\chi = \hat{\mathcal{Y}}^i_{k_N:k} \approx \mathcal{Y}^i_{k_N:k}$. The consensus set is typically defined as the measurements whose residual is less than a user specified threshold τ_{R} [53]. We define a general function that identifies the set of inliers to the states $\hat{\mathbf{x}}$ in the set $\mathcal{Z}_{k_N:k}$ as

$$\chi = \mathsf{Inlier}\big(\mathcal{Z}_{k_N:k}, \hat{\mathbf{x}}, \tau_{\mathrm{R}}, \big). \tag{2.4}$$

For notational convenience, we do not explicitly list the parameters and variables that χ depends on.

The mathematical description of the RANSAC algorithm is given in Algorithm 1 and summarized as follows. During the hypothesis generation step, s measurements are randomly selected to form a minimum subset $\mathcal{S}_{\mathbf{q}}$, where $\mathbf{q}$ defines the indices of the randomly selected measurements from the set $\mathbb{S}^s_{k_N:k}$ (Line 2). Under Assumption 2.5, there exists a function $g : \mathbb{S} \mapsto \mathbb{R}^n$ such that a RANSAC hypothesis is formed according to $\hat{\mathbf{x}}' = g(\mathcal{S}_{\mathbf{q}})$ (Line 3). During the hypothesis validation step, the consensus set χ' is identified as the set of measurements whose residual is less than a user specified parameter τ_{R} to the hypothesis (Line 4). If this is the best hypothesis thus far, then it is stored in memory (Line 6). Typically, the metric to determine the best hypothesis is based on the size of its consensus set. An optional parameter $\gamma \leq 1$ allows for early termination for improved computational

Algorithm 1: Random Sample Consensus (RANSAC) Algorithm [53]

Input: Measurement set $\mathcal{Z}_{k_N:k}$, number of iterations ℓ,
error threshold τ_{R}, stopping criteria γ.

1: **for** ℓ iterations **do**
2: Randomly select a minimum subset $\mathcal{S}_{\mathbf{q}} \in \mathbb{S}^s_{k_N:k}$.
3: Generate a hypothesis $\hat{\mathbf{x}}' = g(\mathcal{S}_{\mathbf{q}})$.
4: Compute the hypothesis consensus set, $\chi' = \mathsf{Inlier}\big(\mathcal{Z}_{k_N:k}, \hat{\mathbf{x}}', \tau_{\mathrm{R}}, \big)$.
5: **if** new hypothesis has larger consensus set than previous hypothesis **then**
6: Store current hypothesis.
7: **end if**
8: **if** $|\chi'| > \gamma N$ **then**
9: break
10: **end if**
11: **end for**
12: Smooth best estimate using consensus set.

efficiency if the size of the consensus set is greater than a percentage of the total number of time steps N (Line 8). The final step of RANSAC is to smooth the best hypothesis over the consensus set (Line 12).

The three parameters in Algorithm 1 are the number of iterations ℓ, the inlier threshold τ_{R}, and the optional stopping criteria γ. Under the assumption that the noise characteristics of the underlying system are known, Fischler and Bolles suggest that the inlier threshold τ_{R} should be set to a few standard deviations above the expected error. They also suggest two methods to estimate the number of iterations ℓ needed to successfully find an estimate of the true signal [53]. First, they suggest setting the number of iterations to be within α_ℓ standard deviations of the number of expected iterations needed to find the signal,

$$\ell = \mathrm{E}[\ell] + \alpha_\ell \, \mathrm{StD}[\ell], \tag{2.5}$$

where

$$\mathrm{E}[\ell] = \frac{1}{p^s}, \tag{2.6}$$

$$\mathrm{StD}[\ell] = \frac{\sqrt{1-p^s}}{p^s}, \tag{2.7}$$

where p is calculated from (2.3). See [53] for the derivation. The probability ζ of selecting at least one minimum subset with all good measurements is equal to

$$\zeta = 1 - (1 - p^s)^\ell. \tag{2.8}$$

The second method is to solve (2.8) for ℓ such that a desired minimum success rate ζ^{d} is achieved, giving

$$\ell = \frac{\log 1 - \zeta^{\mathrm{d}}}{\log 1 - p^s}. \tag{2.9}$$

While it has been shown that the probability of success in (2.8) is optimistic [156], it is an upper bound characterizing the likelihood of RANSAC converging to the correct solution. One of the reasons (2.8) is optimistic is that it does not account for the conditioning of the minimum subset. This effect will be analyzed in Section 5.5.1.

The stopping criteria γ controls how good the hypothesis must be before early termination. When $\gamma = 1$, then a measurement from each time step must be an inlier to the track, resulting in most, if not all, of the ℓ hypotheses to be tested. We typically set γ within a few standard deviations of p_{D}, where the standard deviation of a sampled Bernoulli distribution is $\sqrt{\frac{p_{\mathrm{D}}^s(1-p_{\mathrm{D}}^s)}{N}}$.

2.3 Recursive-RANSAC Framework

The recursive-RANSAC (R-RANSAC) algorithm extends the RANSAC algorithm in two significant ways: to recursively update state estimates based on sequential measurements and to track multiple targets. The basic idea is to avoid recomputing hypotheses between time steps by storing up to $\mathcal{M}$ hypotheses, or tracks, in memory between time steps. For each new measurement scan, R-RANSAC tests each new measurement to see if it is an inlier to one or more existing hypotheses; if so, those hypotheses are updated. Instead of discarding a measurement if it is not an inlier to any existing hypotheses, it is used within a traditional RANSAC step to seed a new hypothesis that best describes that measurement

over a windowed set of measurement scans. Valid hypotheses are identified when they achieve a certain level of support.

The mathematical description of the generic R-RANSAC algorithm is given in Algorithm 2 and summarized as follows. First, all existing tracks are propagated to the current time step (Line 2); note that this step is not needed for static signals. Second, measurements are tested to see if they are inliers to any of the existing tracks (Line 3), where the inlier function finds measurements whose residual is less than the error threshold τ_{RR}. We note that the error could be calculated according to a number of different norms, including the Euclidean norm and the Mahalanobis distance. The adjacency matrix J, which summarizes the binary relation of the inlier status to the existing tracks, is then used to compute the measurement weighting w_{ij} for the j^{th} measurement to the i^{th} track along with the missed detection weight w_{i0} (Line 4). In hard association algorithms such as nearest neighbor (NN) filters, $w_{ij} \in \{0, 1\}$ such that the measurement either is an inlier or is not. In soft data association algorithms such as PDA, $0 \leq w_{ij} \leq 1$ such that the measurement may only partially contribute to the measurement update of the states. The missed detection weight w_{i0} is obviously equal to one when there are no measurements in the gate, but even with measurements in the gate, given $p_{\mathrm{D}} < 1$, the missed detection weight may be non-trivial. The calculation of the measurement weight is a design decision, where some of the options are discussed at the end of this section.

If the measurement $\mathbf{z}_k^i$ is an outlier to all existing tracks, then a variant of the standard RANSAC algorithm is used to generate a new track that best characterizes the measurement (Line 7). The modification to RANSAC is that only minimum subsets that contain the current measurement are used to generate new hypotheses, i.e. $\mathcal{S}_{\bar{\mathbf{q}}} \in \left\{\mathbb{S}_{k_N:k-1}^{s-1} \times \{\mathbf{z}_k^i\}\right\}$, where $\bar{\mathbf{q}}$ defines the indices of the randomly selected measurements from the set of $\binom{\Psi_{k_N:k-1}}{s-1}$ minimum subsets and the current measurement. If the measurement $\mathbf{z}_k^i$ is an inlier to one or more tracks, then it is used to update those tracks according to the weight w_{ij} (Line 9). An example of updating the states with weighted measurements using a modified Kalman filter is given in Appendix A. After processing the measurements, the consensus set and the inlier

ratio for each track are updated (Line 11), where the inlier ratio of the j^{th} track is given by,

$$\rho_k^j = \frac{\left|\chi_k^j\right|}{N}, \tag{2.10}$$

which is the ratio of the number of inliers to the length of the measurement window. Track deaths are modeled by removing tracks according to the probability $(1 - p_{\text{S}})^{\mathsf{MD}}$, where p_{S} is the survival probability and MD is the number of consecutive missed detections (Line 13); note that tracks with a measurement will always survive since $\mathsf{MD} = 0$. Tracks are merged when their states are determined to be similar to within a similarity threshold $\tau_{\mathbf{x}}$ (Line 14). A common similarity metric is the Mahalanobis distance; if $\tilde{\mathbf{x}} = \hat{\mathbf{x}}_k^j - \hat{\mathbf{x}}_k^{j'}$ is the residual between two track estimates and $P \in \mathbb{R}^{n \times n}$ is a covariance matrix, then the Mahalanobis distance is given by

$$\|\tilde{\mathbf{x}}\|_P = \sqrt{\tilde{\mathbf{x}}^\top P^{-1}\, \tilde{\mathbf{x}}}. \tag{2.11}$$

The remaining tracks are pruned to keep the best $\mathcal{M}$ tracks (Line 14). The final step of R-RANSAC is to identify valid tracks (Line 15); for example, valid tracks may be those tracks where both the inlier ratio is over the good track threshold and when the track lifetime is greater than the minimum lifetime threshold. Good tracks are assigned a unique label $\mathcal{L}^j$ for the duration of the track lifetime.

The parameters for R-RANSAC are the number of stored tracks $\mathcal{M}$, the number of stored measurement scans N, the inlier threshold τ_{RR}, the similarity threshold for track merging $\tau_{\mathbf{x}}$, the good track threshold τ_ρ, and the minimum track lifetime τ_{T}. The tuning of these parameters will be discussed in detail in Section 2.6. We refer to the j^{th} R-RANSAC track as the six-tuple $\mathcal{M}^j = (\hat{\mathbf{x}}^j,\, P^j,\, \chi^j,\, \rho^j,\, t^j,\, \mathcal{L}^j)$, such that each R-RANSAC track stores the following properties: estimated track states $\hat{\mathbf{x}}^j$, state covariance P^j, consensus set χ^j, inlier ratio ρ^j, track lifetime t^j, and track label $\mathcal{L}^j$. For convenience and clarity, the time index is implied.

We note that Algorithm 2 is very modular. While the focus of this dissertation is on linear time-invariant systems, the R-RANSAC framework can be extended to nonlinear systems by replacing Lines 2, 7, and 9 with the appropriate nonlinear propagation equation, hypothesis generation, and update equation, respectively. Many of the underlying functions

Algorithm 2: Recursive-RANSAC (R-RANSAC) Algorithm

Input: Previous measurement window $\mathcal{Z}_{k_{k_N}:k-1}$, current measurement scan $\mathcal{Z}_k$, measurement window length N, number of stored tracks $\mathcal{M}$, error threshold τ_{RR}, good track threshold τ_ρ, minimum lifetime threshold τ_{T}.

1: **for** each time step k **do**
2: Propagate state estimate of all active tracks.
3: Compute the association matrix J, where $J_{ij} = \begin{cases} 0 & \text{if } \mathsf{Inlier}\big(\mathbf{z}_k^i, \hat{\mathbf{x}}^j, \tau_{\mathrm{RR}},\big) = \emptyset \\ 1 & \text{otherwise.} \end{cases}$
4: Compute measurement weighting w_{ij} using the association matrix J.
5: **for** each $\mathbf{z}_k^i \in \mathcal{Z}_k$, **do**
6: **if** the measurement is an outlier to all tracks, $\sum_{\forall j}(J_{ij}) = 0$ **then**
7: Create new track using RANSAC, where the current measurement is in each randomly selected minimum subset, $\mathcal{S}_{\bar{\mathbf{q}}} \in \big\{\mathbb{S}^{s-1}_{k_N:k-1} \times \{\mathbf{z}_k^i\}\big\}$.
8: **else**
9: Update $\hat{\mathbf{x}}_k^j$ using w_{ij} and $\mathbf{z}_k^i$.
10: **end if**
11: Update the consensus set χ_k^j and the inlier ratio $\rho_k^j = \frac{|\chi_k^j|}{N}$.
12: **end for**
13: Kill tracks with probability $(1 - p_{\mathrm{S}})^{\mathsf{MD}}$.
14: Merge and prune to keep best $\mathcal{M}$ tracks.
15: Identify good tracks, according to $\rho_k^j \geq \tau_\rho$ and $t_k^j > \tau_{\mathrm{T}}$.
16: **end for**

are also modular; for example, the merging step in Line 14 could save the best of a set of similar tracks; an alternative would be to adapt the merging algorithm in [166, Table II] by weighting each of the similar tracks based on their inlier ratio and combining them into a single track. Additionally, the filtering steps in Lines 2 and 9 could be modified to model unknown target maneuvers using, for example, the interacting multiple model filter [21]. Finally, the measurement weights w_{ij} could be computed according to several existing techniques: nearest neighbor techniques where $w_{ij} \in \{0, 1\}$ describes a hard association, PDA techniques where $0 \leq w_{ij} \leq 1$ describes a soft association, or PMHT techniques that also apply a soft association weight $0 \leq w_{ij} \leq 1$ but that relax Assumption 2.2 to allow for multiple measurements per target per time step.

2.4 Computational Complexity

In this section, we analyze the computational complexity of R-RANSAC. The following theorem demonstrates that the complexity of R-RANSAC is at worst proportional to the number of measurements and stored tracks, squared.

Theorem 1 (R-RANSAC Computational Complexity) *Given the parameters $\mathcal{M}$, N, and ℓ, if the expected number of measurements is known, $\psi = \mathrm{E}[\psi_k] = \lambda_{\mathrm{c}} + M_k$, then the expected complexity of R-RANSAC is*

$$\mathcal{O}\Big(\lambda_{\mathrm{c}}\,\ell\,N\,\psi + M_k + \mathcal{M}^2\Big). \tag{2.12}$$

The worst case complexity, when all the measurements are outliers to all estimated tracks, is

$$\mathcal{O}\Big(\ell\,N\,{\psi_k}^2 + \mathcal{M}^2\Big). \tag{2.13}$$

Proof: The computational complexity of R-RANSAC is proportional to the sum of the computational cost of each step within the algorithm, and is given by

$$c_{\mathrm{tot}} = \underbrace{\mathcal{M}\,c_{\mathrm{P}}}_{\text{Line 2}} + \underbrace{\psi\,\mathcal{M}\,c_{\mathrm{fit}}\,c_{\mathrm{w}}}_{\text{Lines 3-4}} + \underbrace{\lambda_{\mathrm{c}}\,c_R}_{\text{Line 7}} + \underbrace{M_k\,c_{\mathrm{U}}}_{\text{Line 9}} + \underbrace{\mathcal{M}^2}_{\text{Line 14}}, \tag{2.14}$$

where c_{P} is the cost of a Kalman prediction, c_{fit} is the computational cost of computing the residual of an observation to a track, c_{w} is the computational cost of computing the measurement weighting, c_{U} is the cost of a Kalman update, and c_{R} is the computational cost of computing a track using RANSAC. Lines 11, 13, and 15 are of constant complexity and are not included. Cost terms c_{P}, c_{fit}, and c_{U} are all dependent on the size of the state n, and therefore assumed constant for a particular system. The cost of computing the measurement weighting c_{w} is dependent on the scheme used for computing the weights; all neighbors, NN, and PDA are all constant complexity; if full JPDA is used, then the computational complexity is at worst exponential in the number of tracks $\mathcal{M}$.

The computational cost of the RANSAC algorithm is given in [180] as

$$c_R = \ell(c_g + N\psi + N), \tag{2.15}$$

where the first term c_g is the cost to generate a hypothesis using the function g, the second term tests for inliers, and the final term smooths over the measurement window. The cost c_g is a function of the cardinality s of the minimum subset, which is constant for a particular system.

The computational complexity is proportional to the terms with the largest complexity for each parameter, resulting in (2.12). Note that in (2.12), the set of new measurements is partitioned into the set of inliers and outliers and processed accordingly. Since RANSAC tends to be more computationally complex than a Kalman update, we consider the worst case scenario when all new measurements are outliers to all existing tracks, such that (2.12) simplifies to (2.13). Even in the worst case, R-RANSAC is only quadratic in the number of measurements and stored tracks. ■

2.5 Metrics

In order to quantify the performance of MTT algorithms, metrics have been developed beyond the standard root mean-squared error (RMSE), which does not quantify the ability of a tracking algorithm to estimate the number of targets present in the data. Several existing tracking metrics that are widely accepted include track probability of detection, RMSE, execution rate, track false alarm rate, and path fragmentation rate [36]. In addition to these metrics we also introduce two new metrics that are useful to highlight some of the performance differences with existing algorithms: track fragmentation rate and average track fragmentation duration. Other possible metrics for MTT are discussed in [42,58,66,135,141]. This section defines each of these metrics and relates them to the simple scenario shown in Fig. 2.1.

The track probability of detection (TPD) defines the success of the tracker in identifying each target, and is approximated by dividing the number of *unique* detections of true targets by the total number of time steps that all targets exist. Figure 2.1 demonstrates why

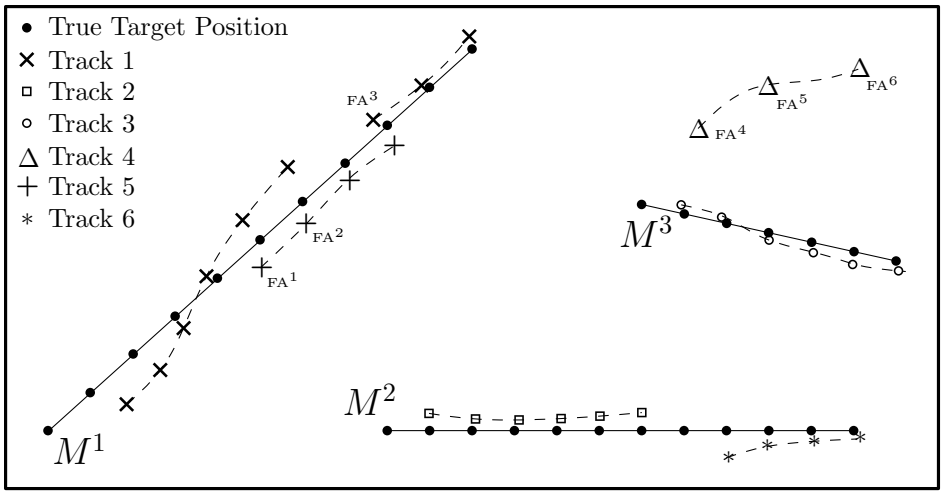

Figure 2.1: Example of true and estimated target tracks to demonstrate metric evaluation. The label FA_i identifies the i^{th} false alarm.

the word unique is emphasized: notice that both track 1 and track 5 estimate the same target, with several time steps estimating the same target position. In order to not overestimate the tracker's performance, a modified Munkres algorithm [112] is used to determine the best unique assignment problem between targets and tracks, where valid track estimates outside a maximum assignment threshold τ_{D} are assumed to be too far away to be a valid track. The minor modification to Munkres algorithm is that the most recent track assignment is given priority in the assignment over alternative assignments to improve the track continuity. In Fig. 2.1, the number of unique track detections is 26 and the total number of possible target detections is 30, resulting in $\mathsf{TPD} = 0.867$. With regards to occlusions, we define TPD such that perfect probability of track detection requires the tracker to track through occlusions within the surveillance region.

The RMSE is calculated as the square root of the mean-squared error, $\mathsf{RMSE} = \sqrt{\frac{\sum_{\kappa=1}^{k} \left\| \hat{\mathbf{x}}_{\kappa}^{j} - \mathbf{x}_{\kappa}^{i} \right\|^2}{N}}$. For clarity, we note that only the assigned tracks are used in the RMSE calculation. The execution rate (ER) is simply the number of seconds needed to run a tracker update.

The path fragmentation rate (PFR), which is in essence the track fragmentation rate defined in [36], is a metric quantifying the track continuity. In this case, PFR is the number of times that the tracker needs to switch between different track labels for a single

target trajectory divided by the total number of tracks. Consider track 1 and track 5 in Fig. 2.1. The metric tries to maximize track continuity by assigning track 1 to target M^1 until track 1 is temporarily no longer a valid track; at that point track 5 is assigned to M^1. Eventually track 1 reacquires the target and is reassigned to M^1 after track 5 dies. For the entire scenario, target M^1 has 3 track assignments, M^2 has 2 assignments, and M^3 has 1 assignment, resulting in $\mathsf{PFR} = \frac{6}{3} = 2$.

The track false alarm rate (TFAR) is equal to the total number of tracks that are not assigned to a target divided by the total number of time steps. Note that this includes both redundant tracks estimating the same target as an existing track as well as a track formed purely by clutter. In Fig. 2.1 there are a total of 6 false alarms denoted by FA^i, where three of the false alarms are redundant measurements of track M^1 as determined by the track assignment discussed in the previous paragraph. If the total number of time steps is $K = 20$, then $\mathsf{TFAR} = 0.3$ false alarms per time step.

Our definition of track fragmentation rate (TFR) is the total number of breaks within each individual track divided by the total number of tracks. The TFR metric captures the tracker's ability to track through missed detections. The track fragmentation duration (TFD) is the average number of time steps required to reacquire a target. For example, track 1 loses track of target M^1 once, but reacquires the target after one time step and continues tracking. In our contrived example, both $\mathsf{TFR} = \mathsf{TFD} = \frac{1}{3}$. Note that the gap between track 2 and track 6 is not included in either TFR or TFD, but is instead quantified as a lower TPD value.

2.6 Parameter Sensitivity Analysis

The parameters for R-RANSAC are the number of stored tracks $\mathcal{M}$, the number of stored measurement scans N, the inlier threshold τ_{RR}, the similarity threshold for track merging $\tau_{\mathbf{x}}$, the good track threshold τ_ρ, the minimum track lifetime τ_{T}, and the number of RANSAC iterations ℓ. One disadvantage of the R-RANSAC algorithm compared to other tracking algorithms is the number of parameters that need to be tuned for specific applications. However, in this section we provide some intuition on what trade-offs should be considered when tuning the parameters; Table 2.1 lists the nominal parameter settings

Table 2.1: Nominal parameter settings of the R-RANSAC algorithm. The constants represent tunable parameters.

Parameter	**Nominal Value**	**Parameter**	**Nominal Value**
τ_{R}	$3\sigma_R$	ℓ	$\frac{1}{\tau_{\mathrm{T}}}\left(\frac{1}{p^s}+2\frac{\sqrt{1-p^s}}{p^s}\right)$
γ	p_{D}	$\mathcal{M}$	$\geq \max M_k$
N	25-50	τ_{RR}	τ_{R}
$\tau_{\mathbf{x}}$	4-6	τ_ρ	$p_{\mathrm{D}}p_{\mathrm{G}}-2\sqrt{\frac{p_{\mathrm{D}}p_{\mathrm{G}}(1-p_{\mathrm{D}}p_{\mathrm{G}})}{N}}$
τ_{T}	10-20		

that work well in examples we have considered. Future work will consider adaptive parameter thresholds to account for varying environmental conditions.

When analyzing the sensitivity of each of the aforementioned metrics, only the parameter of interest is varied while the remaining parameters are constant. While this obviously masks any potential coupling between the parameters, sufficient intuition can be obtained by analyzing the parameters as if they were independent from each other. Thus far in the dissertation, we have described the general R-RANSAC algorithm; subsequent chapters apply R-RANSAC to both static parameter estimation and dynamic signal tracking. In order to characterize the effects of tuning the R-RANSAC parameters, we track dynamic signals using the R-RANSAC algorithm derived in Chapter 5. However, please note that the process of tuning the R-RANSAC parameters generalizes to a wide-range of problems.

Without loss of generality, the subsequent parameter sensitivity analysis relies on the following parameter settings; any changes to the parameters are explicitly stated. The simulation is run for 200 seconds with a time step of $\delta_k = \frac{1}{3}$ seconds. The surveillance region is $\mathcal{R}_k = [0\,1000] \times [0\,1000]$, where seven targets are born and later die during the simulation with a minimum lifetime of at least 25 seconds. The target follows a constant velocity dynamic model, where only the position is measured. The probability of detecting each target at each measurement scan is $p_{\mathrm{D}} = 0.8$, the probability of target survival is $p_{\mathrm{S}} = 0.999$, and an average of $\lambda_{\mathrm{c}} = 5$ clutter returns per time step are uniformly distributed over $\mathcal{R}_k$. The measurement noise for the sensor is set to $R = {\sigma_R}^2 I_m$, where $\sigma_R = 10$ and I_m is an

$m \times m$ identity matrix. The process noise for the Kalman filter is $Q = {\sigma_Q}^2 Q_{\text{NCV}}$, where $\sigma_Q = 1$ and

$$Q_{\text{NCV}} = \begin{bmatrix} \frac{{\delta_k}^4}{4} I_m & \frac{{\delta_k}^3}{2} I_m \\ \frac{{\delta_k}^3}{2} I_m & {\delta_k}^2 I_m \end{bmatrix} \tag{2.16}$$

is the process noise covariance for a nearly-constant velocity (NCV) dynamic model [14, 99]. The R-RANSAC parameters, unless otherwise specified, are the number of stored tracks $\mathcal{M} = 15$, the measurement window length $N = 25$ scans, the number of RANSAC iterations $\ell = 25$, the similar track threshold is a Mahalanobis distance of $\tau_{\mathbf{x}} = 4$, the good track threshold $\tau_\rho = 0.63$, the error threshold $\tau_{\text{R}} = \tau_{\text{RR}} = 3\sigma_R$, the early termination criteria $\gamma = \tau_\rho$, and the minimum lifetime threshold $\tau_{\text{T}} = 20$ time steps. Monte Carlo simulations of 50 iterations are run for each parameter setting, and all simulations were performed in MATLAB on a 3.16 GHz Core 2 Duo processor.

Measurement Window Length, N

To allow greater ability to study the effect of the window length N, for this analysis each target exists during the entire simulation. Figure 2.2 shows that the ER of R-RANSAC is linear with respect to N, as computed in Section 2.4. This is a major improvement over other sliding window based algorithms, such as MHT which is exponential in the length of the measurement window [132]. Larger measurement windows allow for improved data association when generating new tracks.

The remainder of the performance metrics are shown in Fig. 2.3. Both the RMSE and the number of tracks per target improve slightly as the window length increases. One potentially contradictory observation is that the TPD decreases slightly with the increasing N; this is explained by noting that the increased measurement window size increases the delay necessary to achieve an inlier ratio greater than τ_ρ. One possible method to reduce the delay would be to compute the inlier ratio starting from the oldest inlier to more quickly acquire new targets; however, this tends to result in an increase in the number of false tracks. As N increases, there are fewer false alarms caused by redundant measurements and improved probability of detection. Another important benefit of increasing N is that

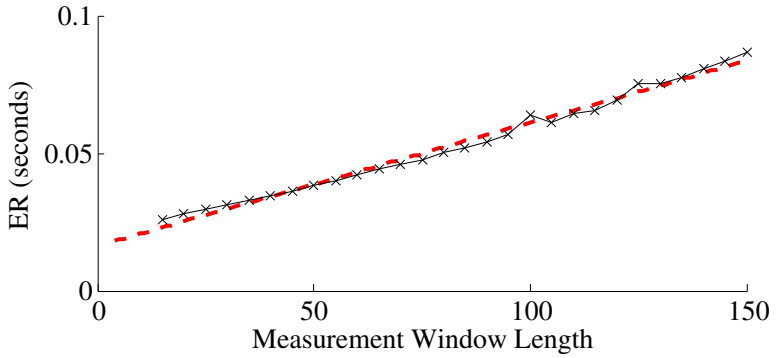

Figure 2.2: Variation in ER when varying the window length N. The dashed red line is a least-squares fit of the data to a line.

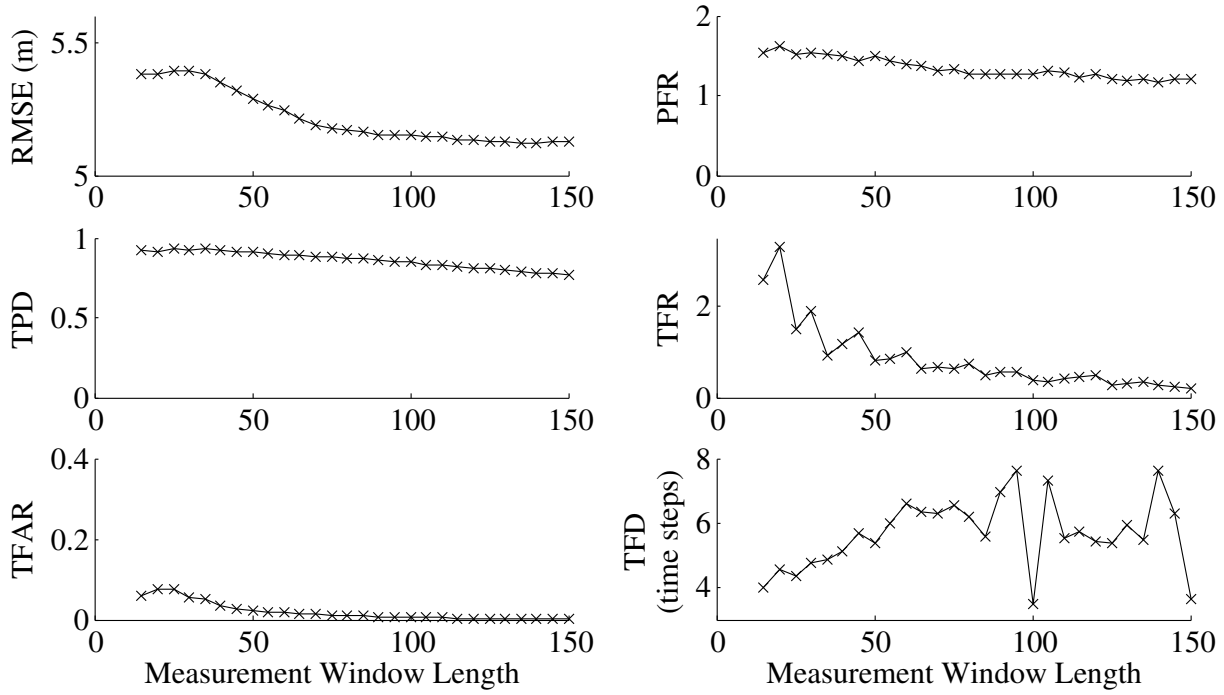

Figure 2.3: Variation in simulation results when varying the window length N.

it reduces the TFR. This is a result of the inlier ratio converging to the true probability of detection, as described later in Section 5.5.2. Fewer breaks per target result in an increased variance in the TFD.

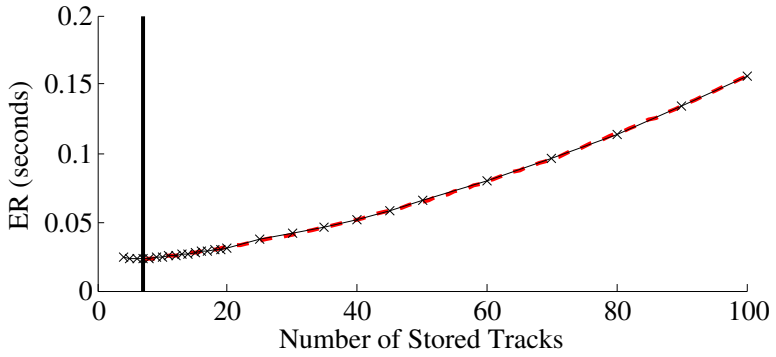

Figure 2.4: Variation in ER when varying number of stored tracks $\mathcal{M}$. The dashed red line is a least-squares fit of the data to a parabola, and the vertical line represents the maximum number of targets during the simulation.

Number of Stored Tracks, $\mathcal{M}$

The lower bound of the number of stored tracks required to track M targets is $\mathcal{M} > M$. When $\mathcal{M} < M$, then there are not enough tracks and the remaining targets will not be tracked. When $\mathcal{M} > M$, then the additional tracks formed by outliers will remain until the tracks die or are merged with an existing track. Fig. 2.4 confirms the analysis of the computational complexity in Section 2.4 that varying the number of stored tracks is proportional to $\mathcal{M}^2$ when $\mathcal{M} \geq M$. Additional computational complexity is required when $\mathcal{M} < M_k$ because there are more outliers than normal since at least one target is not included in the estimated track set.

Figure 2.5 shows the remaining metrics when varying the number of stored tracks. As expected, the detection of targets degrades significantly when $\mathcal{M} < M$. Also note that the TFAR decreases due to the fewer stored tracks. Similarly, the TFR also decreases, since if a track is loosing support then that track will likely be replaced since there are not enough to track all true targets. We find that the RMSE is largely unaffected by the number of stored tracks. For $\mathcal{M} \geq M$, most of the remaining metrics remain nearly constant for all values of $\mathcal{M}$; the only notable exception is the number of false alarms, which increases slightly since more tracks can be stored without being replaced.

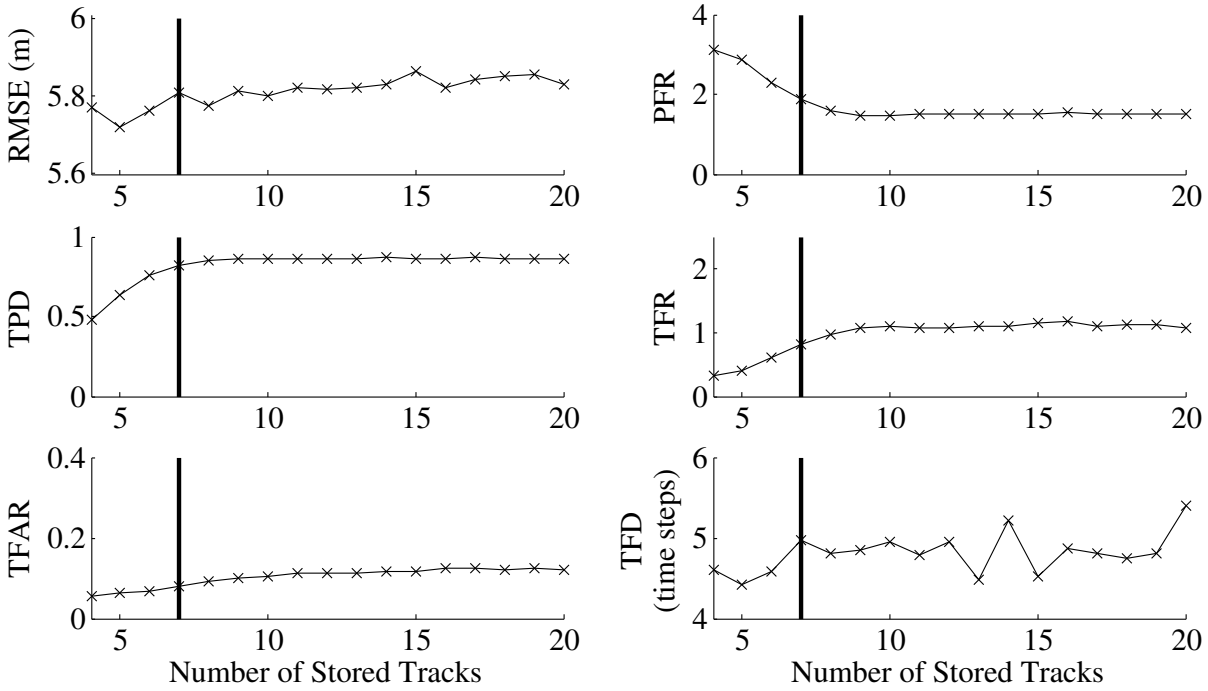

Figure 2.5: Variation in simulation results when varying number of stored tracks $\mathcal{M}$, where the vertical line represents the maximum number of targets during the simulation.

R-RANSAC Inlier threshold, τ_{RR}

One of the modular capabilities of the R-RANSAC algorithm is the specification of what defines the inlier function $\mathsf{Inlier}\big(\mathbf{z}, \hat{\mathbf{x}}, \tau_{\text{RR}},\big)$. While the inlier function was originally defined in the context of RANSAC, tracking techniques refer to the inlier function as the process of gating measurements. Two standard gating techniques are the rectangular gate and the ellipsoidal gate [19]. The rectangular gate corresponds to the gating mechanism used in the standard RANSAC algorithm, and is when the residual of the expected measurement $h(\hat{\mathbf{x}})$ and observed measurements $\mathbf{z}$ is less than a specified threshold. In other words, $\|\mathbf{z} - h(\hat{\mathbf{x}})\|_\infty < \tau_{\text{RR}}$, where the threshold τ_{RR} accounts for both the process and measurement noise. The ellipsoidal gate is when the observed measurement $\mathbf{z}$ of the measurement is less than τ_{RR} standard deviations from the expected measurement $h(\hat{\mathbf{x}})$; this error is also known as the Mahalanobis distance given in (2.11). One advantage of the Mahalanobis distance is that missed detections increase the gate area to accommodate the increased uncertainty.

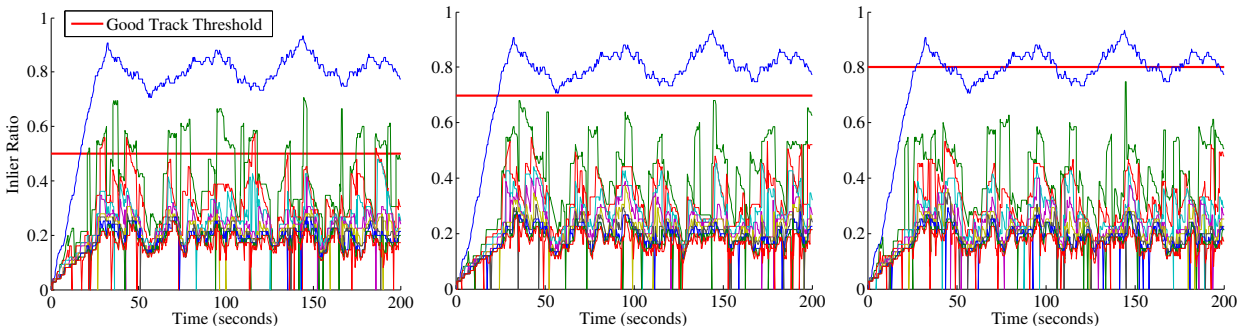

Figure 2.6: Example demonstrating the effect of τ_ρ (red horizontal line) on the probability of target detection and number of false alarms.

However, for simplicity we utilize the rectangular gate, unless stated otherwise, and set $\tau_{\text{RR}} = \tau_{\text{R}}$.

Based on the specified gating technique, we can compute the probability p_{G} that a measurement will be within the detection region. For rectangular gates, the p_{G} is the product of m Gaussian likelihood functions, which depend on the measurement noise level R and the size of the gate τ_{RR}. For ellipsoidal gates, the p_{G} is related to the χ^2 distribution, with specific values based on the dimensionality of the measurement vector m given in [19, Table 6.1].

Good Track Threshold, τ_ρ

The good track threshold has the greatest effect on the probability of detecting targets and the probability of false alarms. For example, consider the scenarios in Fig. 2.6, where the true probability of detecting the target is $p_{\text{D}} = 0.8$. When τ_ρ is set too low, as on the left, then more tracks generated by clutter or high variance measurements are classified as good target tracks. Conversely, when τ_ρ is set too high as on the right, then there are many instances when the good track is not classified as such, and there are many more missed detections.

The size of the measurement window should be considered when setting the good track threshold. As will be discussed in Section 5.5.2, the measurement window length is inversely proportional to the variance of the inlier ratio due to the random nature of the Bernoulli detection process. In our experience, we have found that a good trade-off between missed detections and false alarms is when we set the good track threshold to be a few

standard deviations below the true or expected probability of target detections within the inlier threshold

$$\tau_\rho = p_{\mathrm{D}} p_{\mathrm{G}} - \alpha_\rho \sqrt{\frac{p_{\mathrm{D}} p_{\mathrm{G}} (1 - p_{\mathrm{D}} p_{\mathrm{G}})}{N}}, \tag{2.17}$$

where $\alpha_\rho \geq 0$, and p_{G} is the likelihood of a measurement falling within the gate.

We verify (2.17) by varying τ_ρ in a Monte Carlo analysis, with the results presented in Fig. 2.7. Figure 2.7 shows that performance dramatically decreases when $\tau_\rho > p_{\mathrm{D}}$ (denoted by the solid vertical line) since very few targets will be detected, so we focus our attention when $\tau_\rho \leq p_{\mathrm{D}}$. While not included for brevity, we found that varying τ_ρ has no effect on the ER. Similarly, the RMSE and number of tracks per target are also minimally affected. The center row of plots clearly show that the probability of detection greatly decreases as τ_ρ increases near p_{D} and that there are more instances of missed detections as evidenced by the increased number (and duration) of breaks in the tracks. The trade-off, as shown in the lower left plot, is that the number of false alarms decreases nearly linearly with increasing τ_ρ. In this case, we suggest that the nominal setting given by (2.17) is when $\alpha_\rho = 2$, which is denoted by the dashed vertical line in the figure.

Similar Track Threshold, $\tau_{\mathbf{x}}$

When varying the similar track threshold $\tau_{\mathbf{x}}$, we did not observe noticeable variation in the ER, so it is not included for brevity. In general, we found that $\tau_{\mathbf{x}}$ has a minimal effect on most of the metrics. The one notable exception is that when $\tau_{\mathbf{x}}$ is small, then we see a significant increase in the number of false alarms per time step. Since tracks must be very similar to be merged, then redundant tracks have a higher likelihood of forming and converging to an existing track. Recall that redundant tracks are those tracks that are formed by measurements just outside of the inlier threshold, resulting in higher accuracy tracks. While not studied here, the trade-off is that larger $\tau_{\mathbf{x}}$ leads to increased coalescence of crossing or closely-spaced targets.

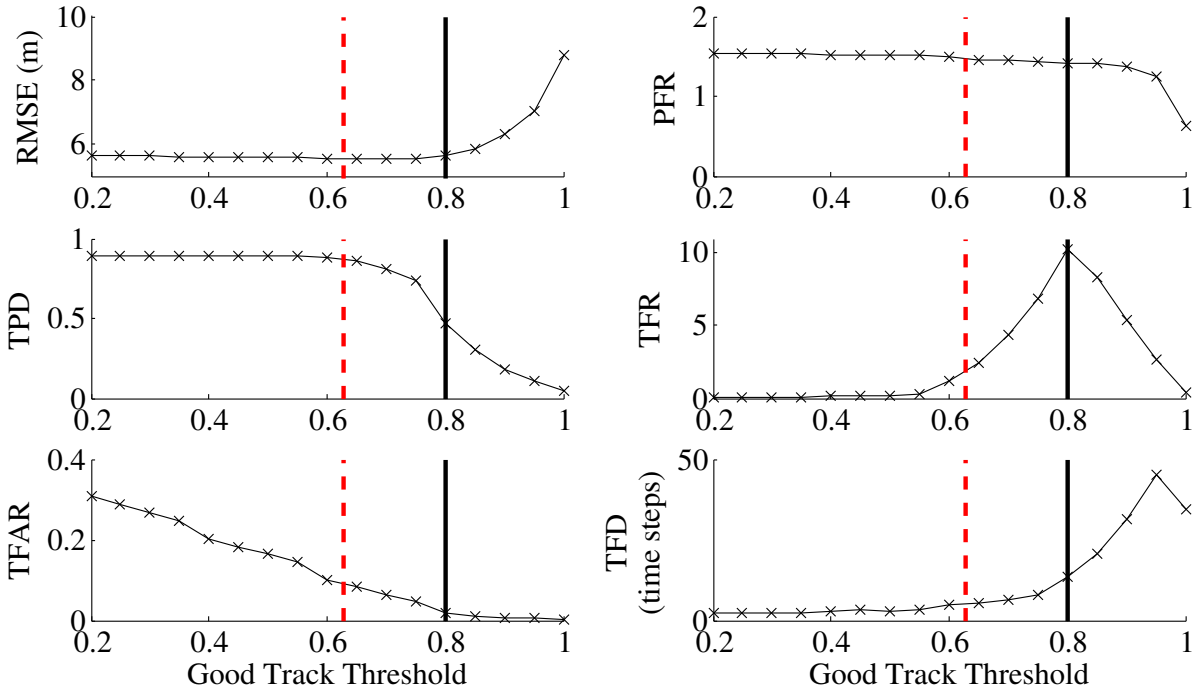

Figure 2.7: Variation in simulation results when varying the good track threshold τ_ρ, where the solid vertical line represents the true target probability of detection. The dashed vertical line represents the inlier threshold for specific simulation when $\alpha_\rho = 2$.

Minimum Lifetime Threshold, τ_{T}

Once again, in order to better observe the effects of varying the minimum lifetime threshold τ_{T}, the lifetime of each of the targets is such that they each persist during the entire simulation. The main trade-off when increasing τ_{T} is reduced TFAR at the expense of decreased TPD. We note that τ_{T} should have a minimal effect on TPD until the threshold is larger than the expected time to achieve the good track threshold $\frac{\tau_\rho N}{p_{\mathrm{D}}}$. When $\tau_{\mathrm{T}} > \frac{\tau_\rho N}{p_{\mathrm{D}}}$, then the minimum lifetime threshold increases the delay otherwise needed to initialize a valid track. This is verified in Fig. 2.9, where $\frac{\tau_\rho N}{p_{\mathrm{D}}} = 19.6$. Also, Fig. 2.9 shows that as a result of the increased number of false alarms, several of the other metrics show minor degradation as τ_{T} is decreased to zero. The ER is not effected as τ_{T} is varied, so it is not included for brevity.

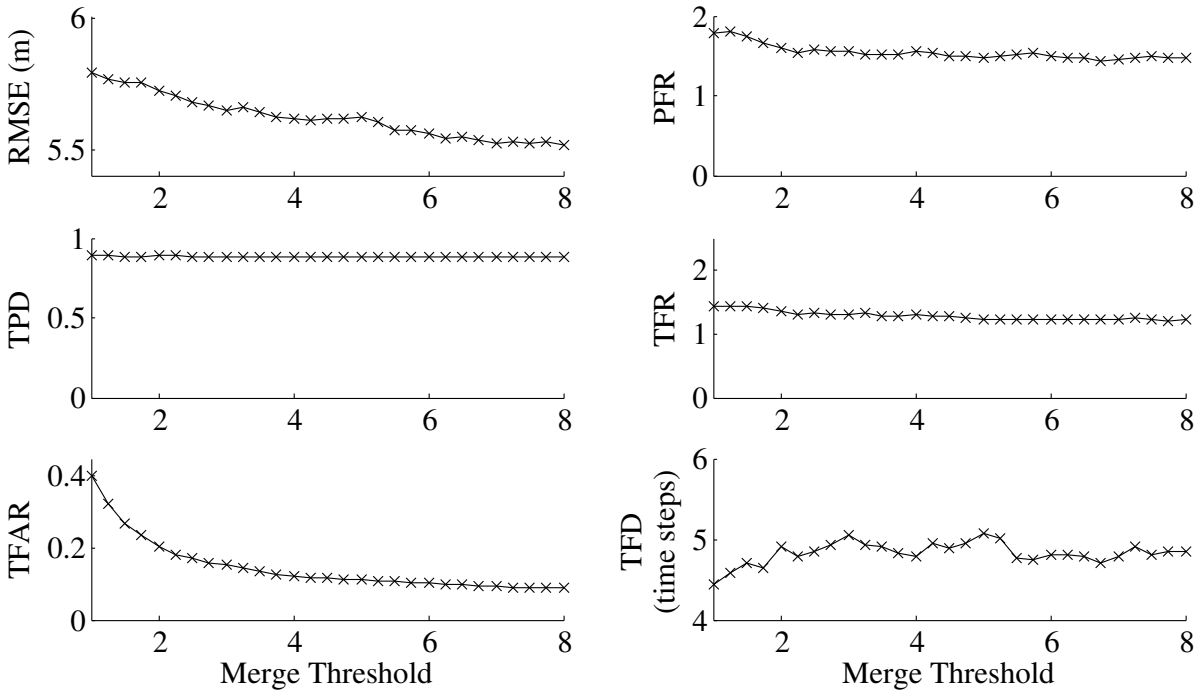

Figure 2.8: Variation in simulation results when varying the similar track threshold $\tau_{\mathbf{x}}$.

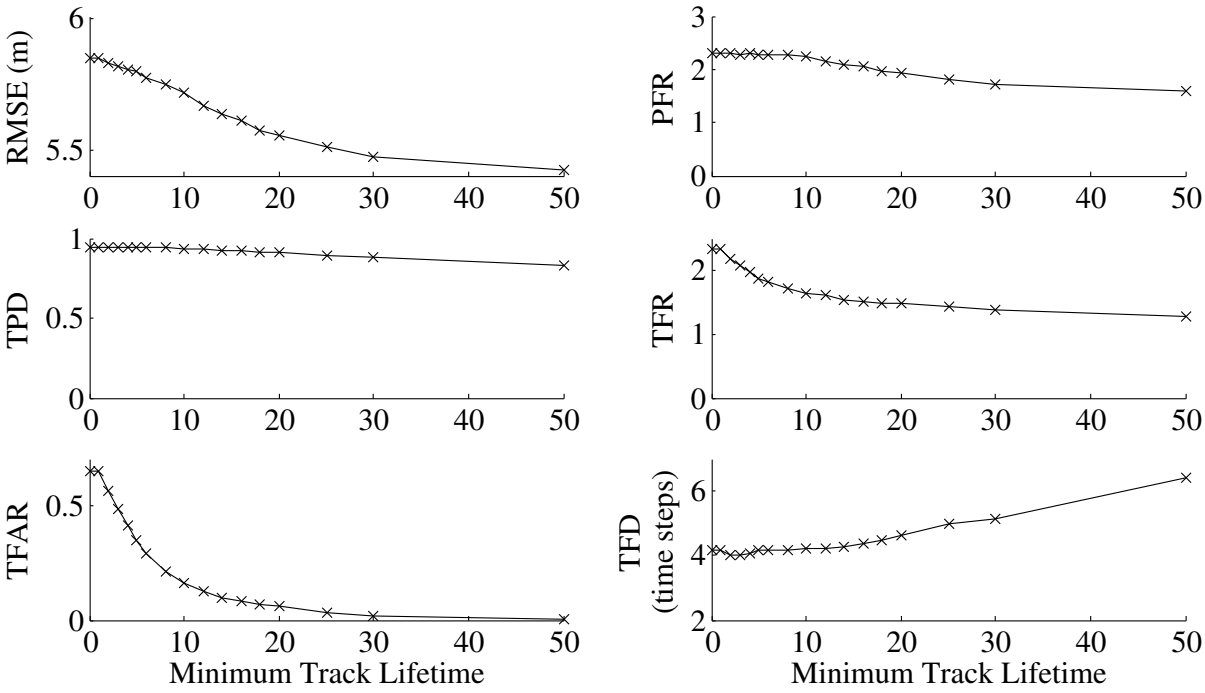

Figure 2.9: Variation in simulation results when varying the minimum lifetime threshold τ_{T}.

Number of RANSAC iterations, ℓ

Recall that in Section 2.2, we review the methods of computing the number of required RANSAC iterations suggested by Fischler and Bolles to ensure that a good minimum subset is selected. The R-RANSAC algorithm has the advantage of utilizing multiple time steps to find the target of interest. In other words, if a new target should be identified with probability ζ^{d} in k^{d} measurement scans, then (2.5) and (2.9) are modified to

$$\ell = \frac{1}{k^{\mathrm{d}}} \left(\frac{1}{{p_{\mathrm{D}}}^{s-1}} + \alpha_\ell \frac{\sqrt{1-{p_{\mathrm{D}}}^{s-1}}}{{p_{\mathrm{D}}}^{s-1}} \right), \tag{2.18}$$

$$\ell = \frac{1}{k^{\mathrm{d}}} \frac{\log 1 - \zeta^{\mathrm{d}}}{\log 1 - {p_{\mathrm{D}}}^{s-1}}, \tag{2.19}$$

respectively, where the terms $s-1$ reflect that the size of the minimum subsets are effectively reduced by one since we seed the RANSAC hypotheses with the current measurement (see Algorithm 2, Line 7). Additionally, the ability to keep a good track in memory helps R-RANSAC perform better than RANSAC in scenarios with many gross errors. To observe this effect, we run 100 Monte Carlo simulations for several values of p_{D}, where the number of RANSAC iterations for both R-RANSAC and RANSAC is $\ell = 20$. By selecting ℓ to be equal for both algorithms, the computational complexity should be similar and so we can more fairly compare their performance. Fig. 2.10 shows the average likelihood of successfully estimating the true signal per scan. Due to the memory element of R-RANSAC, once a good estimate is found, R-RANSAC is able to use that same estimate in the subsequent scan, whereas standard RANSAC must recompute the estimate all over again. Note, that the parameter ℓ could have been increased as p_{D} decreases to improve the likelihood of finding the true signals at the expense of computational complexity.

We return to our original scenario and analysis the effects of varying ℓ. As shown in Fig. 2.11, the ER is linear with respect to ℓ. Figure 2.12 shows that the remaining metrics are minimally affected by ℓ. We expected this for all metrics except for the TPD, which we supposed would increase as ℓ increased and then level off. When the probability of detection is high enough, this effect is hardly seen at all. We believe that this is because the underlying RANSAC is in a sense 'guided' to the correct solution by seeding the hypotheses with the

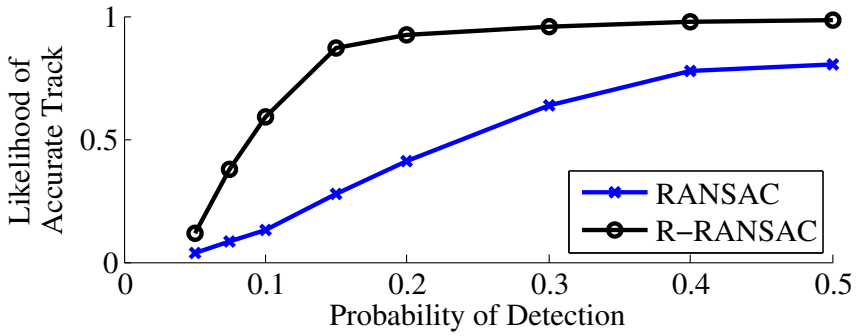

Figure 2.10: Likelihood at any scan that RANSAC and R-RANSAC will correctly estimate the true signal.

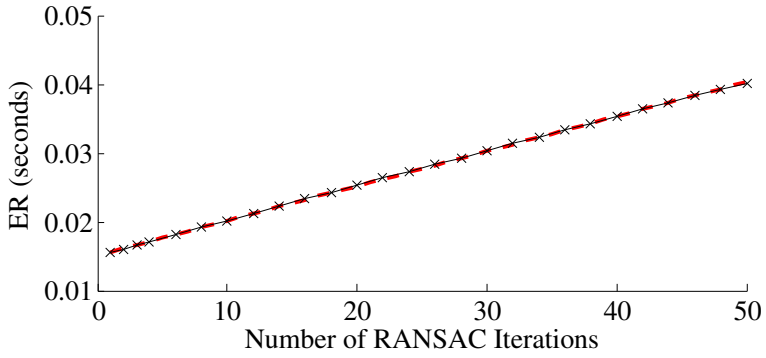

Figure 2.11: Variation in ER when varying the number of RANSAC iterations ℓ. The dashed red line is a least-squares fit of the data to a line.

outliers; also, the minimum lifetime threshold masks the ramp-up time needed for low values of ℓ. To show the underlying effects, we decrease the probability of detection to $p_{\mathrm{D}} = 0.4$ and include the resulting probability of track detection in Fig. 2.12. When the probability of detection is decreased, we see the ramp-up time that we originally expected, along with a 10.2% decrease in overall track detection due to lower p_{D}.

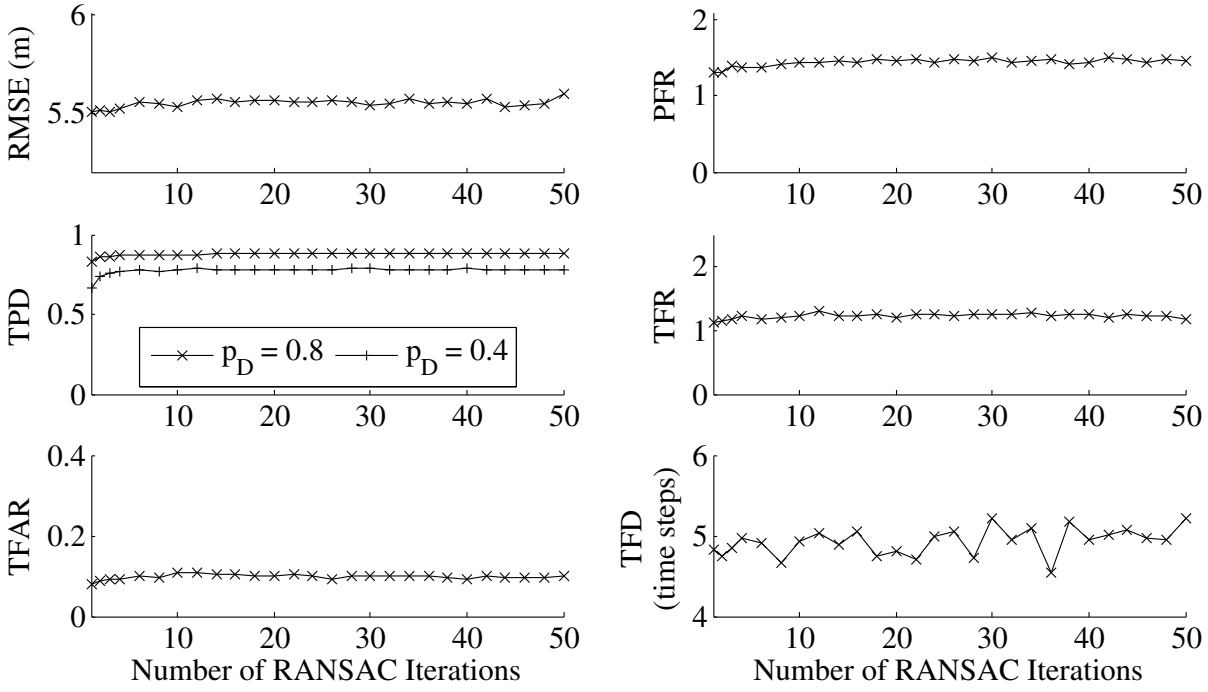

Figure 2.12: Variation in simulation results when varying the number of RANSAC iterations ℓ.

Chapter 3. Parameter Estimation of Multiple Static Signals

Regression analysis is a technique used to model the relationship between input variables and the output of a system. Typically, regression analysis assumes there is a single underlying system generating the outputs. In real data, there often exist multiple signals contributing to the observed data set, all of which we desire to estimate. One application that we are interested in is the geolocation of ground objects from an aerial vehicle using either an electro-optical camera or radar. However, the applications for regression analysis are wide ranging, including computer vision, system identification, sensor fault detection, and more.

When there is no clutter, least squares (LS) is an optimal algorithm to estimate the parameters that best fit a known input/output mapping to a data set by minimizing the mean-squared error [110]. The LS formulation requires that all the observations be distributed with zero-mean and finite variance about the true signal. If there is clutter in a data set, then the LS parameter estimate may diverge from the true signal parameters unless the data association algorithm is perfect [53].

When observations are received sequentially, it is known that LS, while optimal, is not computationally efficient. The inefficiency comes from the need to compute the inverse of a matrix with increasing dimensions. The recursive least-squares (RLS) algorithm is able to avoid this inefficiency by applying the Sherman-Morrison formula [143] to compute a rank one update to the covariance matrix [110]. RLS converges in mean to the LS solution [69]. Similar to LS, RLS requires perfect data association or the estimate can diverge. One heuristic used as a data association technique is to gate the observations by ignoring observations whose residuals or covariance-weighted residuals are larger than some threshold, as in [25]. In multiple target tracking, this technique is called nearest neighbor. In clutter, nearest neighbor techniques can diverge from the true solution [12].

A widely cited, comprehensive treatment on the subject of regression analysis is presented by Draper et al. in [47]. Of particular interest is the chapter devoted to robust regression, where the assumption of Gaussian measurement noise is relaxed to include distributions with heavy tails. Unfortunately, even this improvement does not generalize to clutter. Clutter can be generated through extreme measurement error or through misclassification of possible signal sources. We are interested in scenarios when there are multiple signal sources and when there is no labeling to identify the measurement sources. Data association is the process of associating observations to their respective source: one of the several underlying signals or clutter.

In the context of robust parameter estimation of a single signal, both Rousseeuw and Huber have both published books on the advances in robust regression when allowing clutter. Specifically, they discuss the popular maximum likelihood estimators as well as least median of squares [75, 138]. Unfortunately, both techniques are computationally expensive and require solving a nonlinear minimization problem.

An algorithm that is capable of tracking multiple signals is the Hough transform [72]. The Hough transform (HT) is an algorithm from the computer vision community and is typically used to find parametrized signals such as lines and circles in digital images [9,49,92]. The HT is a voting algorithm, where the parameter space is discretized and observations are used to identify all instantiations of the expected signals. While the HT is capable of detecting multiple signals and rejecting clutter, it is also a batch algorithm and suffers from exponential complexity as the number of parameters increases. Also, the accuracy and efficiency of the HT are inversely related to the discretization size of the parameter bins.

In 1981, Fischler and Bolles proposed the random sample consensus (RANSAC) algorithm to estimate the parameters of a single underlying signal from a batch of observations. RANSAC was specifically designed to mitigate the effects of clutter corrupting the data set. Existing techniques estimated the parameters by considering all data and removing data points inconsistent with the current estimate. RANSAC approaches the problem using the opposite paradigm: form numerous parameter hypotheses using the minimum number of observations, then use a scoring function to compute the strength or support of each parameter

hypothesis. The hypothesis with the largest support is selected as the best estimate of the underlying signal.

Over thirty years later, RANSAC is a standard technique within the computer vision community and is used when performing feature matching, plane estimation, and computing the fundamental matrix [152, 158, 180]. However, RANSAC is only designed to estimate the parameters of a single signal from a batch of data. In this chapter, we apply the recursive-RANSAC (R-RANSAC) framework discussed in Chapter 2 to estimate the parameters of a static signal when the observations are received sequentially. In other words, we extend RANSAC to recursively estimate the parameters of static signals in much the same way that the RLS filter extends LS. In addition, we seek to estimate multiple underlying signals and maintain the clutter rejection properties of RANSAC. This framework for estimating the parameters of static signals is the theoretical foundation of our work in [115, 117].

3.1 Problem Description

We desire to estimate the parameters of all underlying signals given a set of observations. Define a compact set $\mathcal{R}_{\mathbf{z}_x} \subset \mathbb{R}^{m_1}$ over the input space and a compact set over the output space $\mathcal{R}_{\mathbf{z}_y} \subset \mathbb{R}^{m_2}$. Let $\mathcal{R}_k = \mathcal{R}_{\mathbf{z}_x} \times \mathcal{R}_{\mathbf{z}_y}$ be the surveillance region with finite volume $V_k = \mu(\mathcal{R})$, where the function μ defines the measure of the set. Define the inputs to a system as $\mathbf{z}_{x,k} \in \mathcal{R}_{\mathbf{z}_x}$ and let $\mathbf{z}_{y,k} \in \mathcal{R}_{\mathbf{z}_y}$ be the outputs of the system. Referring to the notation developed in Chapter 2, measurements of the i^{th} true signal are defined as the input/output pairs, $\mathbf{y}_k^i = (\mathbf{y}_{x,k}^i, \mathbf{y}_{y,k}^i)$, and general measurements are defined as the input/output pairs, $\mathbf{z}_k = (\mathbf{z}_{x,k}, \mathbf{z}_{y,k})$ for the duration of the chapter. The inputs are related to the outputs according to (2.2), where the measurement function is

$$h(\mathbf{y}_{x,k}, \mathbf{x}) = \phi(\mathbf{y}_{x,k})\,\mathbf{x}, \tag{3.1}$$

where h is linear in the parameters and $\mathbf{x} \in \mathbb{R}^n$. We emphasize that since the states $\mathbf{x}$ are static, the state evolution function in (2.1) is the identity function, i.e. $\mathbf{x} = f(\mathbf{x})$. As specified in Assumption 2.3, the clutter process $\mathcal{K}_k$ is uniform over the surveillance region $\mathcal{R}_k$ and each clutter measurement results in an input/output pair $(\mathbf{z}_{x,k}, \mathbf{z}_{y,k})$.

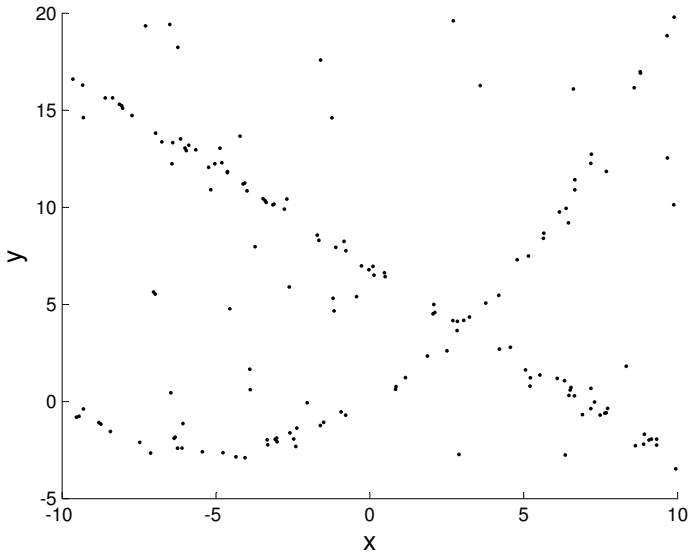

Figure 3.1: Example of a set of observations generated by two signals in clutter.

A simple example illustrating the problem is shown in Fig. 3.1. In this scenario $M = 2$, where ϕ takes the form $\phi(\mathbf{z}_x) = [{\mathbf{z}_x}^2 \ \mathbf{z}_x \ 1]$, where $\mathbf{z}_x$ is drawn from a uniform distribution $\mathcal{U}$(-10, 10). The true, unknown signal parameters are $\mathbf{x}^1 = [0 \ \ -1 \ \ 7]^\top$, and $\mathbf{x}^2 = [0.1 \ \ 1 \ \ 0]^\top$. With $p_{\mathrm{D}}^1 = 0.6$, $p_{\mathrm{D}}^2 = 0.5$ and $\lambda_{\mathrm{c}} = 0.5$. The covariance of the measurement noise is given by $R = {\sigma_R}^2 \, I_{m_2}$, where $\sigma_R = 0.25$ and I_{m_2} is an $m_2 \times m_2$ identity matrix. The total number of measurement scans is $K = 100$. Note that if the measurements were received sequentially, RLS would diverge due to the multiple signal sources and clutter; if RANSAC operated on the batch of measurements, then a single estimate would be generated that would most likely be an estimate of the most probable signal $\mathbf{x}^1$. In the following, we describe the R-RANSAC algorithm that can sequentially estimate the parameters of both signals in the presence of clutter

3.2 Multiple Signal Parameter Estimation using R-RANSAC

In this section, we develop the modular components of the R-RANSAC framework described in Algorithm 2 that are needed to estimate the parameters of multiple signals

in clutter. Specifically, the two main R-RANSAC components discussed in the following sections are the hypothesis generation step and the hypothesis update step.

3.2.1 Parameter Estimation using the Traditional RANSAC Algorithm

In this section, we derive the function that maps minimum subsets to a parameter estimate, $g : \left\{\mathbb{S}^{s-1}_{k_N:k-1} \times \{\mathbf{z}^i_k\}\right\} \mapsto \mathbb{R}^n$. First we consider the LS solution that estimates the parameters $\mathbf{x}$ using all windowed measurements of the true signal. We then use maximum likelihood to estimate the parameters $\mathbf{x}$ using only a minimum subset of data from a window of measurements.

If there were only one signal in the data with no gross errors, i.e. $M_k = 1$ and $\lambda_c = 0$, then LS is the optimal minimum mean-squared estimate of the parameters of the underlying signal. The LS solution is found by solving the following minimization problem [110]

$$\hat{\mathbf{x}}^* = \arg\min_{\mathbf{x}} \sum_{\kappa=k_N}^{k} \|\mathbf{y}_{y,\kappa} - \phi(\mathbf{y}_{x,\kappa})\mathbf{x}\|_\infty. \tag{3.2}$$

Written in matrix notation, where

$$\begin{aligned} Y_{x,k} &= \begin{bmatrix} \mathbf{y}_{x,k_N}, & \mathbf{y}_{x,k_N+1}, & \ldots, & \mathbf{y}_{x,k} \end{bmatrix}^\top, \\ Y_{y,k} &= \begin{bmatrix} \mathbf{y}_{y,k_N}, & \mathbf{y}_{y,k_N+1}, & \ldots, & \mathbf{y}_{y,k} \end{bmatrix}^\top, \\ \Phi(Y_{x,k}) &= \begin{bmatrix} \phi(\mathbf{y}_{x,k_N}), & \phi(\mathbf{y}_{x,k_N+1}), & \ldots, & \phi(\mathbf{y}_{x,k}) \end{bmatrix}^\top, \end{aligned}$$

then the LS solution is given by

$$\hat{\mathbf{x}}^* = \left(\Phi(Y_{x,k})^T \Phi(Y_{x,k})\right)^{-1} \Phi(Y_{x,k})^\top Y_{y,k}. \tag{3.3}$$

Returning to the multiple signal parameter estimation problem posed in Chapter 2, we now consider when there are multiple, unlabeled measurements received at each measurement

scan, such that $M_k \geq 1$ and $\lambda_c \geq 0$. Define the following vectors and matrices,

$$\begin{aligned} Z_{x,k} &= \left[\mathbf{z}_{x,k_N}^1, \quad \ldots, \quad \mathbf{z}_{x,k_N}^{\psi_{k_N}}, \quad \mathbf{z}_{x,k_N+1}^1, \quad \ldots, \quad \mathbf{z}_{x,k}^{\psi_k}\right]^\top, \\ Z_{y,k} &= \left[\mathbf{z}_{y,k_N}^1, \quad \ldots, \quad \mathbf{z}_{y,k_N}^{\psi_{k_N}}, \quad \mathbf{z}_{y,k_N+1}^1, \quad \ldots, \quad \mathbf{z}_{y,k}^{\psi_k}\right]^\top, \\ \Phi(Z_{x,k}) &= \left[\phi(\mathbf{z}_{x,k_N}^1), \quad \ldots, \quad \phi(\mathbf{z}_{x,k_N}^{\psi_{k_N}}), \quad \phi(\mathbf{z}_{x,k_N+1}^1), \quad \ldots, \quad \phi(\mathbf{z}_{x,k}^{\psi_k})\right]^\top. \end{aligned}$$

Since the vectors $Z_{x,k}$ and $Z_{y,k}$ contain measurements of both multiple signals and clutter, we utilize RANSAC to estimate the parameters of the underlying signals in the data set. RANSAC is a suboptimal approach of estimating the true parameters $\mathbf{x}$ that are formed by using a minimum number of measurements. To do so, we must define a binary indicator matrix $\mathcal{B}$ that allows us to select specific measurements pairs of interest. In general, define $\mathbf{d} = [\mathbf{d}_1, \ldots, \mathbf{d}_d]^\top$ to be a vector where each element is an index to a specific measurement in $\mathcal{Z}_{k_N:k}$. Let $\mathcal{B}_i(\mathbf{d})$ be defined as a binary indicator matrix of dimensions $m_i d \times m_i \Psi_{k_N:k}$, where

$$\mathcal{B}_i(\mathbf{d}) = \begin{bmatrix} \mathbf{0}_{m_i \times m_i(\mathbf{d}_1 - 1)} & I_{m_i} & \mathbf{0}_{m_i \times m_i(\Psi_{k_N:k} - \mathbf{d}_1)} \\ \vdots & \vdots & \vdots \\ \mathbf{0}_{m_i \times m_i(\mathbf{d}_d - 1)} & I_{m_i} & \mathbf{0}_{m_i \times m_i(\Psi_{k_N:k} - \mathbf{d}_d)} \end{bmatrix}, \tag{3.4}$$

and where $i = 1$ for the input binary matrix and $i = 2$ for the output binary matrix. Given the binary matrix $\mathcal{B}_i(\mathbf{q})$ identifying the measurements within the minimum subset, define the vectors $\bar{Z}_{x,k} = \mathcal{B}_1(\mathbf{q}) Z_{x,k}$ and $\bar{Z}_{y,k} = \mathcal{B}_2(\mathbf{q}) Z_{y,k}$. The function mapping minimum subsets to estimates of the parameter vector is equal to

$$\hat{\mathbf{x}}(\mathbf{q}) = g(\mathcal{S}_\mathbf{q}) \triangleq \left(\Phi\big(\bar{Z}_{x,k}\big)^\top \, \Phi\big(\bar{Z}_{x,k}\big)\right)^{-1} \Phi\big(\bar{Z}_{x,k}\big)^\top \, \bar{Z}_{y,k}. \tag{3.5}$$

During each call to RANSAC, up to ℓ hypotheses $\hat{\mathbf{x}}'$ are formed during the hypothesis generation step. During the hypothesis validation step, the consensus set χ' is computed for each hypothesis using $\mathsf{Inlier}\big(\mathcal{Z}_{k_N:k}, \hat{\mathbf{x}}', \tau_\mathrm{R}, \big)$, where

$$\chi' = \mathsf{Inlier}\big(\mathcal{Z}_{k_N:k}, \hat{\mathbf{x}}', \tau_\mathrm{R}, \big) \triangleq \left\{\mathbf{z}_\kappa^i \in \mathcal{Z}_{k_N:k} : \left\|\mathbf{z}_{y,\kappa}^i - \phi(\mathbf{z}_{x,\kappa}^i)\, \hat{\mathbf{x}}'\right\|_\infty < \tau_\mathrm{R}\right\}. \tag{3.6}$$

The final step and solution to RANSAC is to identify the hypothesis $\mathbf{x}'$ with the largest consensus set $\chi^* = \arg_{\chi'} \max \mu(\chi')$ and to perform LS over that consensus set. With a slight abuse of notation, where $\mathcal{B}_1(\chi^*)$ returns a binary matrix representing the measurements in the consensus set, we have

$$\hat{\mathbf{x}} = g(\chi^*), \tag{3.7}$$

$$P = \left(\Phi(Z_{x,k})^\top \mathcal{B}_1(\chi^*)^\top \mathcal{B}_1(\chi^*)\Phi(Z_{x,k})\right)^{-1}. \tag{3.8}$$

3.2.2 Parameter Estimation using R-RANSAC

We now specify the underlying functions of Algorithm 2 describing R-RANSAC. In this case, since the parameters $\mathbf{x}$ are time-invariant, we can skip the propagation step in Line 2. To compute the association matrix in Line 3, we use the function $\mathsf{Inlier}\big(\mathcal{Z}_k, \hat{\mathbf{x}}', \tau_{\mathrm{R}},\big)$ as defined in (3.6). New hypotheses are generated using (3.7) and (3.8), where the support of each new hypothesis is given by (3.6).

Finally, we describe the update step in Line 9. Since we are estimating the parameters of static signals, it is possible to utilize the LS algorithm to smooth over the current consensus set χ_k^j; however, this becomes costly as the dimensions of the covariance matrix P become large. Instead, we utilize the Sherman-Morrison formula [143] to recursively update the covariance using a rank-one update. This process is known as RLS filtering [110]. The RLS update step to the j^{th} parameter estimate using measurement $\mathbf{z}_k^i$ is given by

$$L_k^j = \frac{P_{k-1}^j \, \phi^\top(\mathbf{z}_{x,k}^i)}{\lambda_{\mathrm{f}} + \phi(\mathbf{z}_{x,k}^i) \, P_{k-1}^j \, \phi^\top(\mathbf{z}_{x,k}^i)},$$

$$\hat{\mathbf{x}}_k^j = \hat{\mathbf{x}}_{k-1}^j + L_k^j \left(\mathbf{z}_{y,k}^i - \phi(\mathbf{z}_{x,k}^i) \, \hat{\mathbf{x}}_{k-1}^j\right), \tag{3.9}$$

$$P_k^j = \lambda_{\mathrm{f}}^{-1} \, P_{k-1}^j - \lambda_{\mathrm{f}}^{-1} \, L_k^j \, \phi(\mathbf{z}_{x,k}^i) P_{k-1}^j, \tag{3.10}$$

where $0 < \lambda_{\mathrm{f}} \leq 1$ is a forgetting factor.

Chapter 4. Applications to Static Parameter Estimation

In this chapter, we include some of the results of our experiments when using the recursive-RANSAC (R-RANSAC) algorithm to estimate the parameters of static signals. In Section 4.1, we compare R-RANSAC to several existing single-signal regression techniques: gated-recursive least-squares (RLS), RANSAC, and the Hough transform (HT). Some of these results were previously published in [115]. In Section 4.2, we estimate the parameters of multiple Legendre polynomials when the number of underlying signals is both known and unknown. Section 4.3 is a case study simulating the geolocation of static and slowly drifting targets using an aerial vehicle; some of these results are also presented in [115]. Finally, in Section 4.4 we develop a RANSAC technique to estimate the parameters of the hyperbolas characterizing the range returns of ground reflectors detected by a synthetic aperture radar (SAR) [117]. We show that R-RANSAC has over a factor of 20 speed increase over the HT [72] using both simulated and real data for similar accuracy.

4.1 Comparison with Other Parameter Estimation Techniques

In this section, we compare R-RANSAC to gated-RLS, RANSAC, and the HT algorithms. A scenario where all four algorithms are applicable is when estimating the slope and intercept of a single line with noisy observations in clutter. In other words, the function $\phi(\mathbf{z}_x)$ in (3.1) is characterized by $\phi(\mathbf{z}_x) = [\mathbf{z}_x,\ 1]$. We define the total number of scans to be $K = 1000$, and define a static surveillance region $\mathcal{R} = \mathcal{R}_{\mathbf{z}_x} \times \mathcal{R}_{\mathbf{z}_y}$, where $\mathcal{R}_{\mathbf{z}_x} = [0,\ 500]$, and $\mathcal{R}_{\mathbf{z}_y} = [0,\ 500]$. The measurement noise is $R = {\sigma_R}^2$, where $\sigma_R = 2$. The parameters of the R-RANSAC and RANSAC algorithms are $\tau_{\mathrm{R}} = 3\sigma_R$, $\gamma = \frac{2}{3}p_{\mathrm{D}}$, $\mathcal{M} = 2$ and $\tau_{\mathbf{x}} = [0.05,\ 10]^\top$. The number of random minimum subsets selected within RANSAC is $\ell = 30$, and $\ell = 10$ for R-RANSAC. The HT parameters are designed to look for the single best signal. For an equal comparison with R-RANSAC, the gate size of the gated-RLS is fixed and equal to τ_{R}. We

see that one disadvantage of gated-RLS is that it requires an accurate prior estimate of the underlying signal to avoid gating good observations. For this scenario, we assume that the first two observations are noisy but correct; using these two measurements the gated-RLS algorithm forms an initial estimate.

We perform two Monte Carlo simulations. First, we vary the probability of detection

$$p_{\mathrm{D}} \in \Big\{0.2,\, 0.3,\, 0.4,\, 0.5,\, 0.6,\, 0.7,\, 0.8,\, 0.9,\, 1.0\Big\},$$

where the measurement window length $N = 100$ is constant between all simulations. We find that with the given parameter settings, for $p_{\mathrm{D}} < 0.2$ all algorithms are inaccurate for the stated parameters. Second, we vary the window size

$$N \in \Big\{50,\, 100,\, 150,\, 200,\, 250,\, 300,\, 350,\, 400,\, 450,\, 500\Big\},$$

where the probability of detection $p_{\mathrm{D}} = 0.7$ is constant between all simulations. For each parameter configuration we perform 100 Monte Carlo simulations to estimate the parameters of a randomly generated line, where the only constraint on the line parameters is that the slope and intercept are bounded so that $(\mathbf{z}_x, \mathbf{z}_y) \in \mathcal{R}$. Since there is only one signal, we modify Assumption 2.3 by assuming that up to one clutter measurement is observed according to a Bernoulli process. In other words, we allow one measurement per time step such that a good measurement is received with probability p_{D} and a gross error occurs with probability $1 - p_{\mathrm{D}}$.

Figure 4.1 displays the average root-mean-squared error (RMSE) for all algorithms when varying the probability of detection. The average RMSE for R-RANSAC is as accurate or more accurate than other algorithms in scenarios under consideration. Note that the performance of all algorithms decreases as the probability of detection decreases. As is shown in Fig. 4.1, gated-RLS occasionally fails which skews the mean, but the median RMS error is approximately equal to R-RANSAC. In other words, while the gated-RLS filter usually is accurate, if $p_{\mathrm{D}} < 1$ then it is not provably robust in clutter. The parameters ℓ and N in RANSAC and R-RANSAC can be increased to more accurately track signals in low detection environments at the expense of computational complexity.

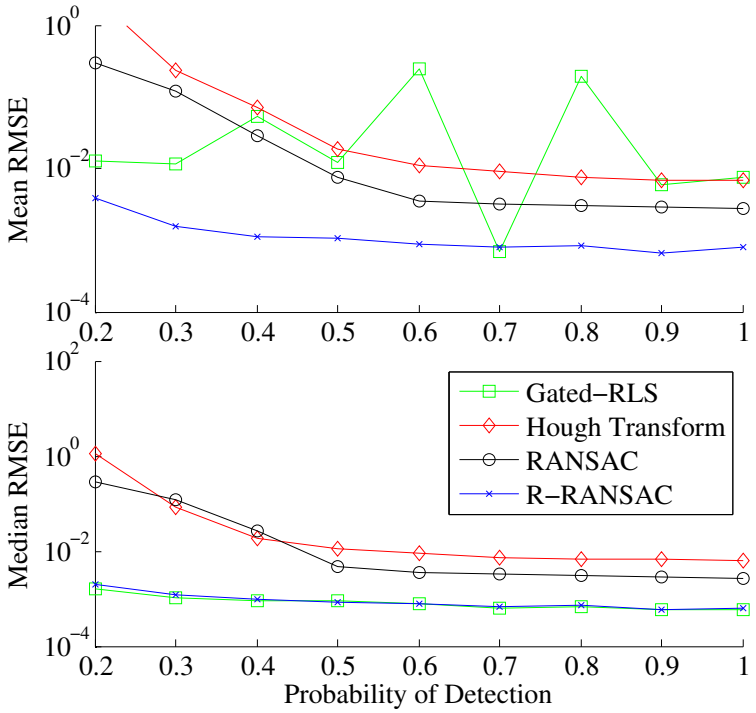

Figure 4.1: Comparison of the average RMSE of the slope for each algorithm when varying the probability of detection when $N = 100$.

Figure 4.2 displays the average execution rate (ER). Due to the brute force voting scheme of the HT, it is the slowest algorithm running at about 30 Hz. The gated-RLS filter is the fastest in all clutter densities. We also see that RANSAC is actually faster than R-RANSAC given the particular window size. This is due to the overhead needed in R-RANSAC to estimate multiple signals simultaneously, had they existed.

Figure 4.3 shows the average ER when the window size is varied and the probability of detection is fixed. While not shown, we find that when the probability of detection is sufficiently high, increasing the window size offers little improvement in accuracy. However, as the window size is increased, the time per scan for RANSAC also increases, eventually becoming slower than the R-RANSAC algorithm. The ER of R-RANSAC remains nearly constant, demonstrating the advantages of the underlying recursion employed by RLS to update the hypothesis parameters.

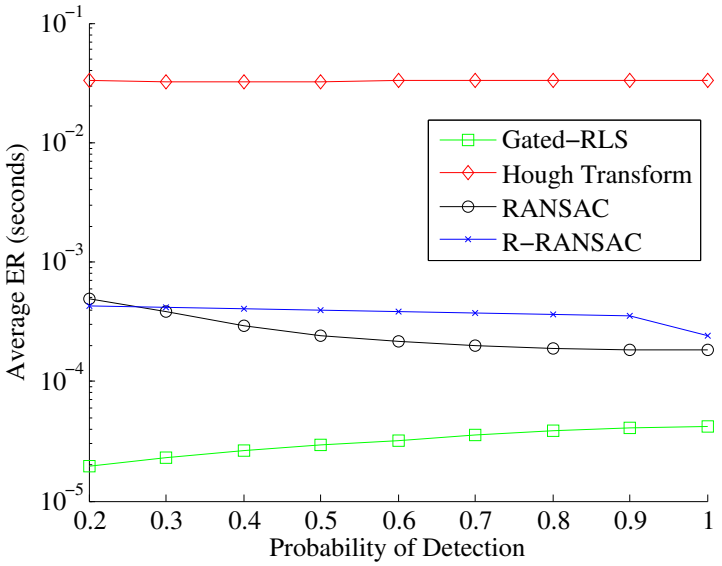

Figure 4.2: Comparison of the average ER required by each algorithm when varying the probability of detection when $N = 100$.

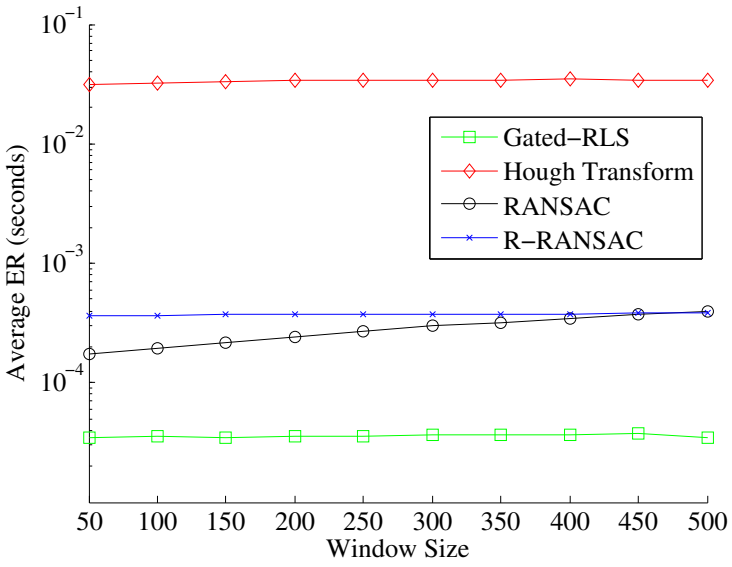

Figure 4.3: Comparison of the average ER required by each algorithm when varying the window size when $p_1 = 0.7$.

4.2 Legendre Polynomial Parameter Estimation

In the previous example, the full capability of R-RANSAC was under-utilized in order to compare it with existing algorithms. In this section, we estimate the parameters of multiple Legendre polynomials of the form $\phi(\mathbf{z}_x) = [1,\ \mathbf{z}_x,\ \frac{3{\mathbf{z}_x}^2-1}{2},\ \frac{5{\mathbf{z}_x}^3-3\mathbf{z}_x}{2}]$. Two cases are considered, depending on whether or not we have prior knowledge of the number of signals M_k. In both cases the R-RANSAC algorithm is able to successfully estimate the correct parameters; although, when the number of signals is unknown, the measurement window needs to be larger to differentiate the good signals from other false signals with high inlier ratios.

Number of Targets is Known

First, we look at the case when the number of signals M_k is known. We track $M = 5$ randomly generated signals that persist during the entire simulation, whose parameters are given by

$$\begin{aligned}
\mathbf{x}^1 &= [\ -0.227,\ \ -0.352,\ \ 0.171,\ \ 0\ ]^\top,\\
\mathbf{x}^2 &= [\ 0.140,\ \ -0.077,\ \ 0,\ \ 0\ ]^\top,\\
\mathbf{x}^3 &= [\ -0.160,\ \ 0.698,\ \ 1.041,\ \ -0.563\ ]^\top,\\
\mathbf{x}^4 &= [\ 0.303,\ \ 1.289,\ \ -0.754,\ \ -0.950\ ]^\top,\\
\mathbf{x}^5 &= [\ -0.631,\ \ 0.331,\ \ 0.777,\ \ -0.237\ ]^\top.
\end{aligned}$$

Samples are drawn from the static surveillance region $\mathcal{R} = \mathcal{R}_{\mathbf{z}_x} \times \mathcal{R}_{\mathbf{z}_y}$, where $\mathcal{R}_{\mathbf{z}_x} = [-1,\ 1]$, and $\mathcal{R}_{\mathbf{z}_y} = [-2,\ 2]$. Each signal is measured with a probability of detection $p_{\mathrm{D}} = 0.8$ with an average clutter rate of $\lambda_{\mathrm{c}} = 0.6$ measurements per time step. The measurement noise is $R = {\sigma_R}^2$, where $\sigma_R = 0.01$. The parameters of the R-RANSAC algorithm are $\tau_{\mathrm{R}} = 3\sigma_R$, $\ell = 60$, $N = 200$, $\gamma = \frac{2}{3}p_{\mathrm{D}}$, $\mathcal{M} = 6$, $\tau_{\mathbf{x}} = [0.1,\ 0.1,\ 0.1,\ 0.1]^\top$ and $\tau_\rho = 0.7$. After simulating observations for $K = 5000$ scans, we find that the RMSE for each component of the five

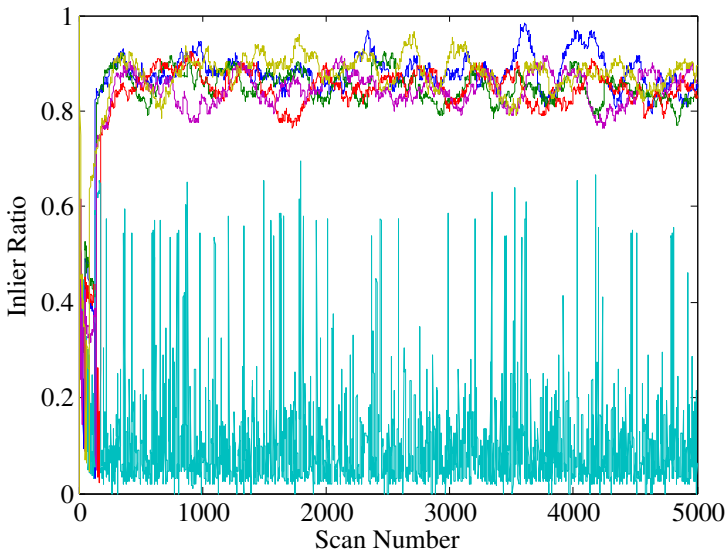

Figure 4.4: The inlier ratios of all R-RANSAC hypotheses estimating the coefficients $M = 5$ Legendre polynomials.

signals is given by

$$\begin{aligned}
\mathsf{RMSE}_{\mathbf{x}^1} &= [\ 0.638,\ \ 2.41,\ \ 2.40,\ \ 2.98\ ]^\top \times 10^{-3},\\
\mathsf{RMSE}_{\mathbf{x}^2} &= [\ 0.296,\ \ 0.299,\ \ 1.05,\ \ 0.453\ ]^\top \times 10^{-3},\\
\mathsf{RMSE}_{\mathbf{x}^3} &= [\ 0.377,\ \ 0.797,\ \ 0.711,\ \ 1.50\ ]^\top \times 10^{-3},\\
\mathsf{RMSE}_{\mathbf{x}^4} &= [\ 0.301,\ \ 0.192,\ \ 0.913,\ \ 0.660\ ]^\top \times 10^{-3},\\
\mathsf{RMSE}_{\mathbf{x}^5} &= [\ 0.143,\ \ 0.428,\ \ 0.529,\ \ 0.605\ ]^\top \times 10^{-3}.
\end{aligned}$$

The average ER of R-RANSAC per scan was 3.599×10^{-3} seconds or 278 Hz. In Fig. 4.4, we see that ρ_k^j for each of the five good parameter estimates is larger than the remaining hypotheses.

Number of Targets is Unknown

Relaxing the assumption that the number of signals is known, we perform the same experiment except we set the number of stored hypotheses to $\mathcal{M} = 20$ to account for our uncertainty in the number of true signals. We find that R-RANSAC is still able to accurately

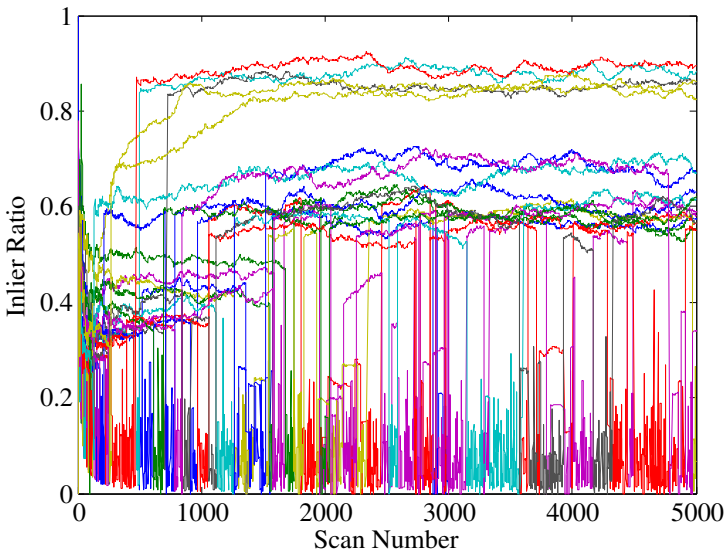

Figure 4.5: The inlier ratios when estimating the coefficients of M Legendre polynomials when M is unknown. The inlier ratios of the five good hypotheses are found at the top of the figure. Inaccurate hypotheses with high inlier ratios make it much more difficult to distinguish good tracks.

track the true signal parameters of each of the M signals. However, due to the overlap of the true signals, it is much more difficult to discern between the true signal estimates and other inaccurate R-RANSAC hypotheses with high inlier ratios. When $\mathcal{M} = M$, inaccurate R-RANSAC hypotheses with high inlier ratios are still formed but are immediately pruned since they are less accurate than estimates of the true signals. When $\mathcal{M} > M$, most of the inaccurate R-RANSAC hypotheses with high inlier ratios are not replaced and can be classified as good hypotheses if τ_ρ is set low enough.

In order to more easily distinguish the good and bad parameter estimates, we increase the window size to $N = 750$ and increase the number of RANSAC iterations to $\ell = 150$. This increases the gap between the good and bad parameter estimates at the expense of reducing the ER to 101 Hz in addition to increasing the time needed to first find the good hypotheses, as shown in Fig. 4.5. The five good parameter estimates have inlier ratios that are nearly equal to the true probability of detection of 0.8. Note that there are also multiple inaccurate parameter estimates with high inlier ratios. Figure 4.6 shows the two next-best hypotheses, whose largest inlier ratios are just below the good track threshold. Notice how they estimate piecewise segments of several signals.

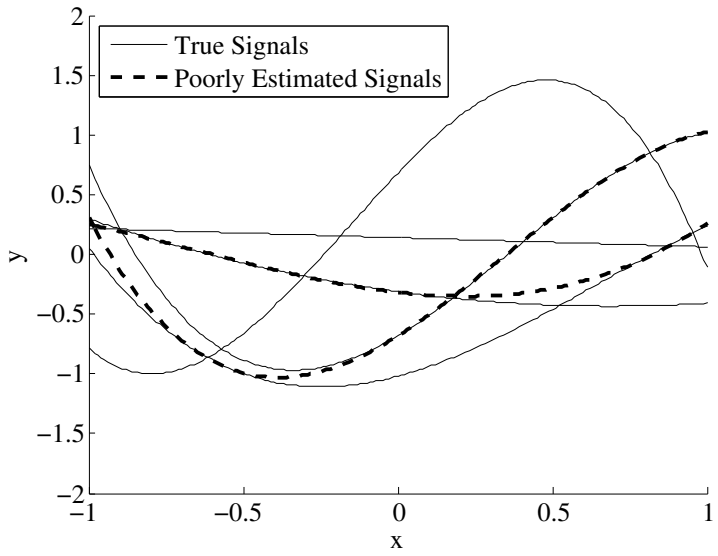

Figure 4.6: Inaccurate, high-probability hypotheses formed by R-RANSAC compared to the true signals. Note that each hypothesis is estimating segments of multiple true signals.

4.3 Geolocation Simulation

We now apply R-RANSAC to a simple, yet practical example to estimate the position of multiple ground locations from an aerial vehicle. We compare the R-RANSAC results to the probabilistic data association (PDA) filter [13]. The PDA filter is a standard multiple target tracking algorithm that is often used to track a target with dynamic states. While typically used to track a single target in clutter, PDA works equally well as JPDA when the stationary targets are well-separated. We modified an existing implementation by Dubin to track stationary locations [48]. We demonstrate the importance of R-RANSAC's ability of simultaneously estimating the number of signals present in the data set without any prior knowledge of the number of signals that exist. Two scenarios are considered: the first scenario tracks stationary targets; the second scenario tracks slowly drifting targets by setting the forgetting factor in (3.9) and (3.10) to be less than one.

Simulation and Algorithm Parameters

Let $\mathbf{z}^{\mathrm{v}}_{x,k} \in \mathbb{R}^3$ be the true position of the aerial vehicle, and let $\mathbf{z}^{i}_{x,k} \in \mathbb{R}^3$ be the true position of the i^{th} target. The position of the aerial vehicle can be measured by the global

positioning system (GPS), for example, and the position measurement is modeled by

$$\mathbf{z}^{\mathrm{v}}_{y,k} = \mathbf{z}^{\mathrm{v}}_{x,k} + \mathbf{v}^{\mathrm{v}}_{k},$$

where $\mathbf{v}^{\mathrm{v}}_{k}$ is a zero-mean Gaussian random variable with covariance matrix $R^{\mathrm{v}} = {\sigma_{R^{\mathrm{v}}}}^2\, I_{3\times 3}$, where $\sigma_{R^{\mathrm{v}}} = 2$. A scaled version of the line of sight vector to each ground target can be measured, for example, by obtaining bearing from a vision sensor. Accordingly, the measurement of the line of sight vector is given by

$$\mathbf{z}^{i}_{y,k} = \frac{1}{c_{\mathrm{s}}}\left(\mathbf{x}^{i} - \mathbf{z}^{\mathrm{v}}_{x,k} + \mathbf{v}^{i}_{k}\right),$$

where c_{s} is the (unknown) scale ambiguity, $\mathbf{x}^{i}$ is the position of the i^{th} ground target, and $\mathbf{v}^{i}_{k}$ is a zero-mean Gaussian random variable with covariance matrix $R^{i} = {\sigma_{R^{i}}}^2\, I_{3\times 3}$, where $\sigma_{R^{i}} = 1, \forall i$. A common approximation is to assume a flat earth, as in [34]. Under this assumption, the scale ambiguity can be approximated by using the estimated vehicle altitude relative to a flat world. We also assume that the sensor measuring the line of sight vector to each target is omni-directional. The aerial vehicle flight path is an orbit with radius of 100 meters around the center of the surveillance region, at an altitude of 100 meters above flat terrain flying at an airspeed of 15 m/s.

The probability of detecting each target is equal to $p_{\mathrm{D}} = 0.8$, where the average number of clutter returns is $\lambda_{\mathrm{c}} = 0.5$ per time step. We demonstrate the ability of R-RANSAC to identify a target and track all five targets, while the PDA filter can only track the initial four targets. The parameters of the R-RANSAC algorithm are $\tau_{\mathrm{R}} = 10$, $\ell = 30$, $\gamma = 0.2$, $N = 50$, $\mathcal{M} = 10$ and $\tau_{\mathbf{x}} = 3$ meters. The good track threshold is $\tau_{\rho} = 0.5$.

In order for the PDA filter to work, it must be supplied with initial estimates and covariances of each target; we assume that the first two observations are valid. This requirement is not necessary with R-RANSAC since no prior information on the states is required. The total number of scans is $K = 1000$. The static surveillance region is $\mathcal{R} = \mathcal{R}_{\mathbf{z}_x} \times \mathcal{R}_{\mathbf{z}_y}$, where $\mathcal{R}_{\mathbf{z}_x} = [0,\ 600]$, and $\mathcal{R}_{\mathbf{z}_y} = [0,\ 600]$. For time $k = 1:499$, $M_k = 4$ targets are randomly positioned within $\mathcal{R}$, but at $K = 500$ a fifth target appears so that $M_k = 5$ for $k = 500:K$.

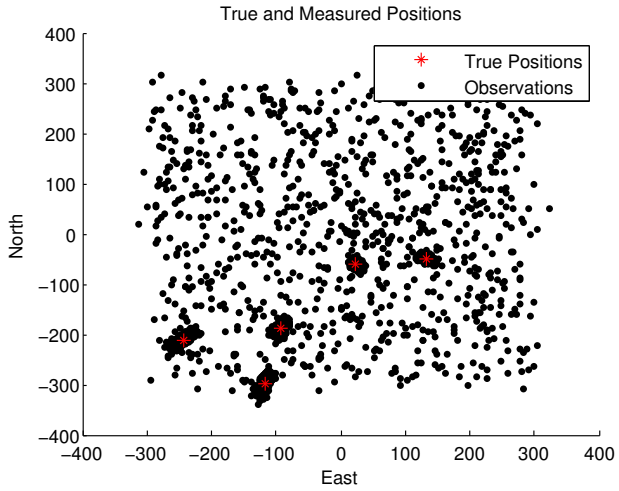

Figure 4.7: Measurements of five stationary ground targets projected onto the flat earth.

Table 4.1: Position RMSE of static ground targets tracked by an aerial vehicle.

RMSE in meters					
Algorithm	Target 1	Target 2	Target 3	Target 4	Target 5
PDA	2.769	2.477	1.378	0.822	NA
R-RANSAC	1.821	1.746	0.988	0.331	0.426

Stationary Targets

In this simulation, since the targets are known to be stationary, the RLS forgetting factor in (3.9) and (3.10) is set to $\lambda_f = 1$. The received measurements of randomly positioned targets of a typical simulation are projected onto a flat earth and are shown in Fig. 4.7. The error of both the R-RANSAC and the PDA filter of the individual targets for a single run are shown in Fig. 4.11, with the RMSE for each target shown in Table 4.2. Both algorithms are able to accurately track each of the four original targets. The PDA filter is about 15% faster since it does not include a track management algorithm; however, this renders the PDA filter unable to detect the existence of new targets, while R-RANSAC detects and tracks Target 5.

Figure 4.12 shows the inlier ratio of all R-RANSAC hypotheses. Note that the hypothesis tracking the fifth target locks on soon after it appears. There is a delay before identifying tracks as good tracks, due to the need for the inlier ratio to exceed the good

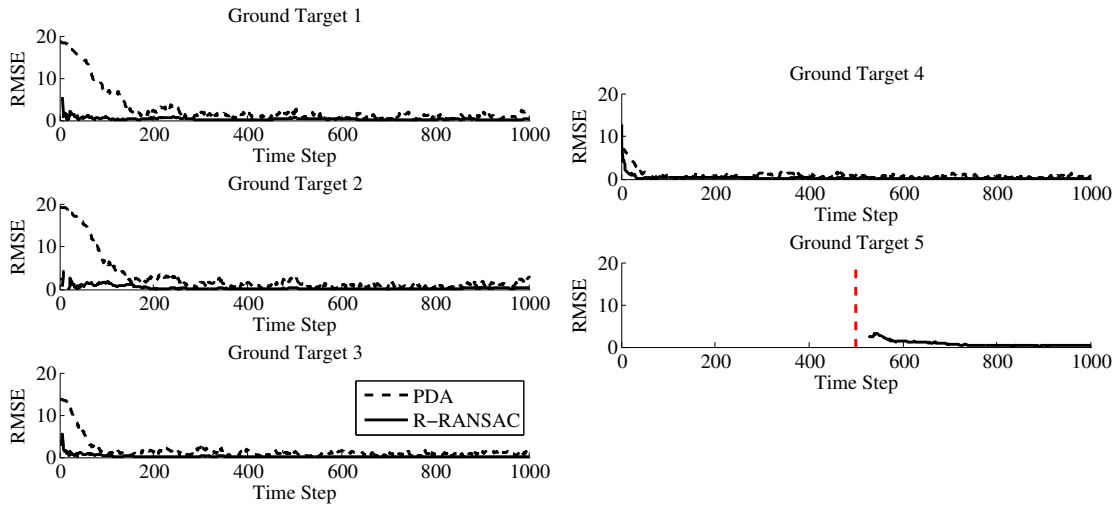

Figure 4.8: Single simulation results showing error of stationary ground targets converging in mean. The vertical dashed red line denotes when target five first appeared.

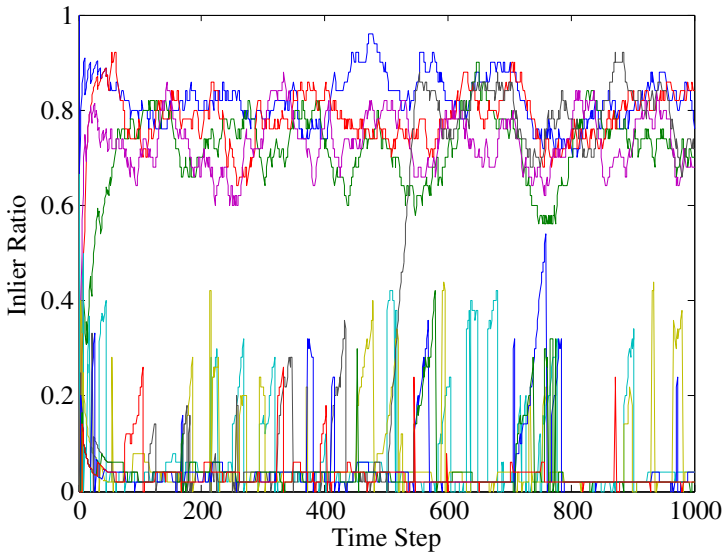

Figure 4.9: Inlier ratio of all R-RANSAC tracks in the static geolocation example. Note that the targets are clearly recognizable, and that the algorithm quickly recognizes the addition of the fifth target around scan 500.

track threshold τ_ρ. Also note that outliers near the true targets generate hypotheses that start to converge to the true target position, but once they are within the similar track threshold $\tau_{\mathbf{x}}$ of an existing hypothesis they are removed.

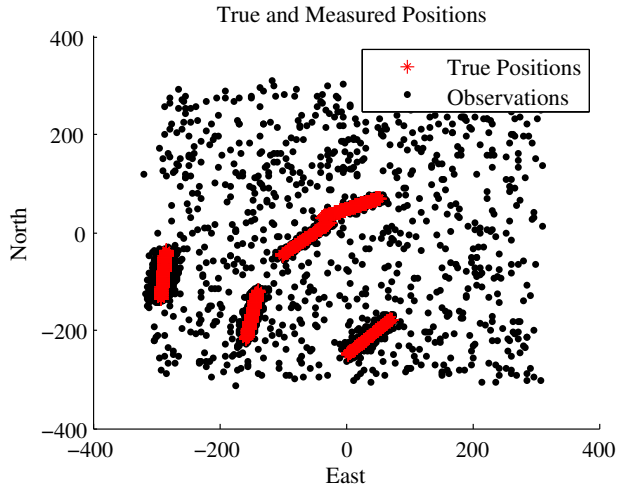

Figure 4.10: Measurements of five slowly drifting ground targets projected onto the flat earth.

Table 4.2: Position RMSE of slowly drifting ground targets tracked by an aerial vehicle.

RMSE in meters					
Algorithm	Target 1	Target 2	Target 3	Target 4	Target 5
PDA	4.218	4.307	2.900	4.099	NA
R-RANSAC	3.627	3.948	3.043	3.914	2.514

Slowly Drifting Targets

In this set of simulations, targets slowly drift at a constant speed of 0.1 m/s with a constant, but random, heading. To account for the drifting targets the RLS forgetting factor in (3.9) and (3.10) is set to $\lambda_{\mathrm{f}} = 0.97$. The received measurements of a typical simulation with randomly positioned targets are projected onto a flat earth and are shown in Fig. 4.10.

Since the error convergence is similar to the static case shown in Fig. 4.8, we show the results of a single target in Fig. 4.11. The average RMSE for all five targets is shown in Table 4.2. Both algorithms are able to accurately track each of the four original targets. As before, without prior information, the PDA filter is unable to detect the existence of a new target, while R-RANSAC detects and tracks new targets. Figure 4.12 shows the inlier ratios of all estimated signals.

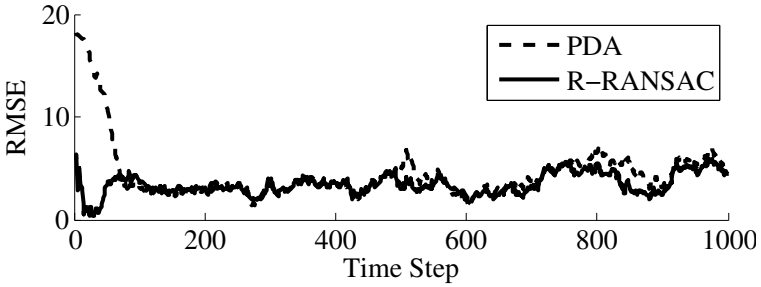

Figure 4.11: Single simulation result showing the position estimate of a stationary ground target converging in mean.

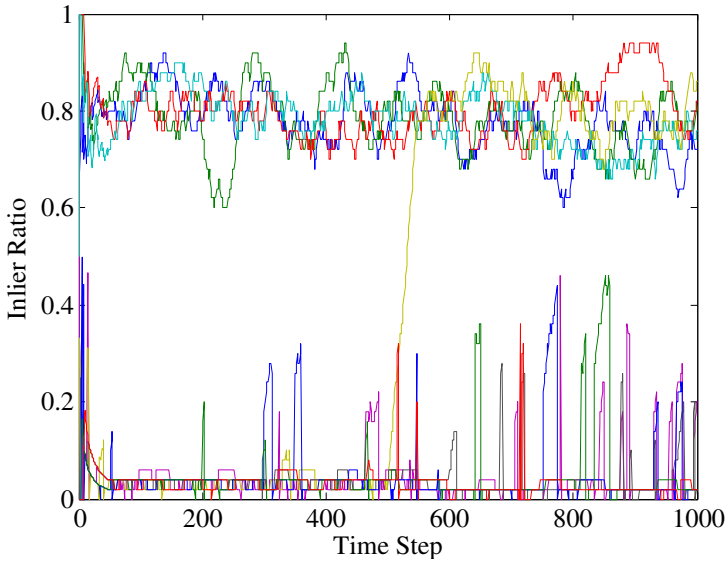

Figure 4.12: Inlier ratio of all R-RANSAC tracks in the static geolocation example. Note that the targets are clearly recognizable, and that the algorithm quickly recognized the addition of the fifth target around scan 500.

We also perform a Monte Carlo analysis of 100 simulations tracking five slowly drifting targets to compare the ER and RMSE of the PDA filter and R-RANSAC algorithms. We found that the PDA filter is faster, but also slightly less accurate than R-RANSAC. The results are shown in Fig. 4.13. It is possible that the accuracy of the PDA filter could be improved slightly by extending it to the joint-PDA algorithm, which is better equipped to track targets that are close together [56]. However, since the targets are nearly stationary, we expect only a small bias in these rare cases. The important result is that R-RANSAC is very comparable in terms of accuracy and speed to this well-established algorithm, with

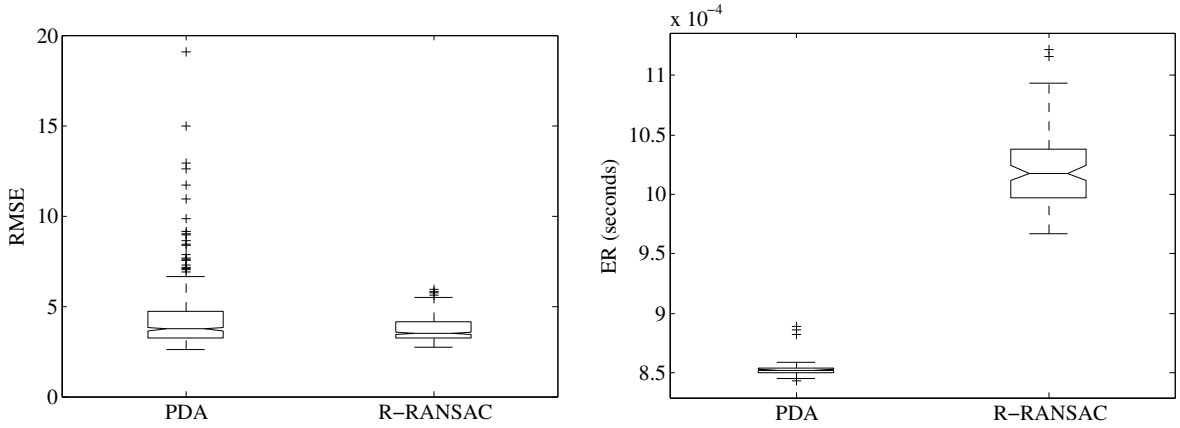

Figure 4.13: Monte Carlo geolocation RMSE and ER of slowly drifting targets. The mean is marked by the notch in the box, while the box defines the 25^{th} and 75^{th} percentiles.

the added benefit of being able to track an unknown and varying number of targets without any prior information.

4.4 Characterizing the Range Progression of SAR Point Scatterers

Current civilian and military navigation technology requires position and velocity measurements from the GPS. In air vehicles, sensor fusion with GPS measurements allows a navigation system to eliminate the drift inherent to inertial navigation systems. Unfortunately, there are scenarios where GPS is unavailable, such as in urban canyons or during GPS jamming or spoofing. In these instances, measurements from other sensors can be incorporated to prevent the air vehicle state estimates from drifting.

Odometry is a technique that uses on-board sensors to measure changes in vehicle location by tracking reference features. Numerous sensors can be used to calculate the odometry measurements, such as infrared and electro-optical cameras, laser range finders, sonar, radar, and even star-tracking sensors. In this work, our objective is to use an aerial SAR to efficiently and robustly track point scatterers. SAR, as opposed to visual sensors, provides high range resolution, while being more robust to environmental conditions such as night, fog, and rain.

The SAR utilized in our experiments is a linear frequency modulated, continuous wave (LFM-CW) SAR with a single transmit antenna and a single receive antenna. As the radar transmits a chirp, objects on the ground, referred to as scatterers, reflect the pulse back at the receiver. The range-dependent delay between transmission and reception is then measured by mixing and filtering the transmit and receive signals. Using the resulting range-compressed chirp, the range to any ground scatterer may be calculated [177].

Using the radar described in [62], Kauffman et al. developed a sparse reflector tracking technique for navigation [84, 86]. Using ground reflectors, Kauffman et al. sought to reduce the inertial navigation system drift over time by including SAR measurements of ground reflectors [61, 84, 85, 88–91]. In [90], Kauffman et al. continued the sparse data assumption in proposing an M/N detector given a signal-to-noise ratio, which uses the global nearest neighbor (GNN) algorithm for data association. Using this method, they were able to track reflectors and update the state estimates of the air vehicle in higher noise levels without degradation of the accuracy of the state estimates.

In this section, we relax three of the assumptions typically employed in the existing literature for tracking using SAR imagery. First, since SAR imagery is inherently dense [57, 133, 160], we do not make assumptions regarding the sparsity of scatterers, noise, or clutter. Secondly, phase history assumptions are not needed [84, 87]. Finally, the size and power of current SAR implementations (specifically involving the selection of the 5 GS/s analog-to-digital converter) are focused on accuracy, and have only been applied in ground tests using strong metallic reflectors [87]. We utilize a small SAR [78] well-suited for light-weight mobile platforms and conduct simulated and actual ground point scatterer tracking. We are grateful for Eric Quist providing the simulated and actual radar return data.

One of the effects of relaxing the constraints on the scatterer sparsity is that there are often multiple scatterers next to each other. Additionally, clutter is typically present in radar returns. In both cases, it has been shown that GNN is not robust and can diverge from the true data association [19]. Similar to Kauffman, recent work by Quist et al. uses the range return from scatterers to estimate motion [131]. Rather than use a GNN data association algorithm, a HT fits hyperbolas to the time-varying range returns for robust estimation of scatterer locations [9, 72]. Unfortunately, the HT is a voting algorithm and is

computationally expensive. Instead of the HT, we apply the R-RANSAC algorithm to track ground scatterers from SAR returns.

4.4.1 Point Scatterer Return Geometry

Our objective is to recursively and robustly estimate ground scatterer locations relative to an air vehicle. One method to detect and track ground scatterer locations is with a side scanning SAR unit. Let the inertial north-east-down velocity vector of the air vehicle be denoted as $V_g = [v_n,\, v_e,\, v_d]^\top$. In this section, we make four assumptions to facilitate tracking of the ground scatterers:

Assumption 4.1: Level flight, i.e., $v_d = 0$.

Assumption 4.2: Velocity is bounded, i.e., $\underline{v} \leq v_n, v_e \leq \overline{v}$.

Assumption 4.3: Straight flight with unknown constant velocity, i.e., $[\dot{v}_n,\, \dot{v}_e,\, \dot{v}_d]^\top = \mathbf{0}$.

Assumption 4.4: Level terrain.

Assumption 4.1 is typical and often applied as a method of conserving fuel [94]. Assumption 4.2 is realistic since the performance capabilities of the air vehicle will be known. Finally, Assumptions 4.3 and 4.4 are required in this iteration of our work and are also used in [89,90].

In a range compressed radar image, the radar returns over time of a single scatterer are given by the equation of a hyperbola [131]. In this section, we break from our original notation and redefine the measurement pair $(\mathbf{z}_x, \mathbf{z}_y)$. In continuous time, we define the input variable $\mathbf{z}_x$ to be the time t of a particular SAR scan, and define the output variable $\mathbf{z}_y$ as the range r_t^i to the i^{th} scatterer at time t. In discrete time, we define the input variable $\mathbf{z}_x$ to be the radar chirp k, and define the output variable $\mathbf{z}_y$ as the discrete range bin b_k^i characterizing the range of the i^{th} scatterer at chirp k. Assuming that the aircraft is flying straight and level, the hyperbola describing the range to the i^{th} scatterer over a time period is given by

$$\frac{1}{{r^i_{\min}}^2} {r_t^i}^2 - \frac{{V_{\text{g}}}^2}{{r^i_{\min}}^2}(t - t^i_{\min})^2 = 1, \tag{4.1}$$

where V_{g} is the relative ground speed, and $t^i_{\min}$ and $r^i_{\min}$ refer to the time and range when the range to the i^{th} scatterer is minimized, respectively [131]. In hardware, both the time

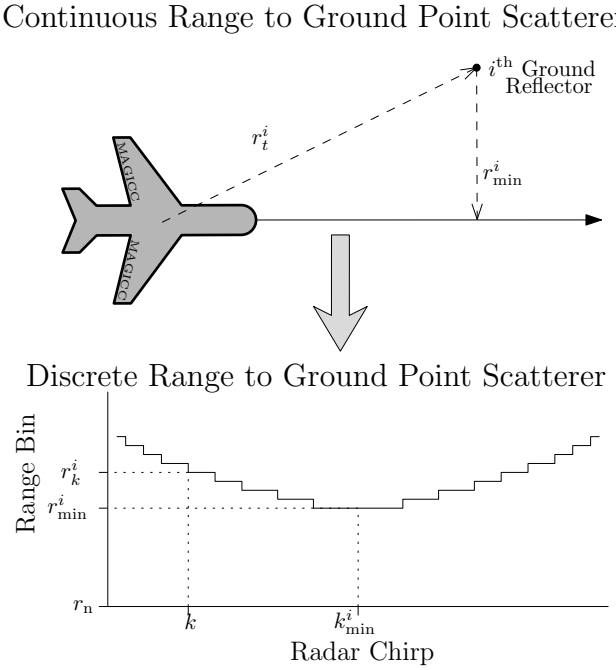

Figure 4.14: Geometry describing a SAR return of a ground point scatterer from an air vehicle with constant velocity.

and the range are discretized into chirps k and range bins b, such that

$$r_t = r_{\text{n}} + b_k r_{\text{r}}, \tag{4.2}$$

$$t = \frac{k}{\tau}, \tag{4.3}$$

where r_{n} is the radar near range, r_{r} is the radar bin resolution, and τ is the chirp rate. Substituting (4.2) and (4.3) into (4.1), we have

$$\frac{(r_{\text{n}} + b_k^i r_{\text{r}})^2}{(r_{\text{n}} + b_{\text{min}}^i r_{\text{r}})^2} - \frac{{V_{\text{g}}}^2 \left(\frac{k - k_{\text{min}}^i}{\tau}\right)^2}{(r_{\text{n}} + b_{\text{min}}^i r_{\text{r}})^2} = 1, \tag{4.4}$$

where k_{min}^i and b_{min}^i are the chirp and range bin where the minimum range is achieved, respectively [131]. Figure 4.14 displays an example of a noiseless return of a single scatterer.

False Alarms using SAR

We modeled the clutter return based on empirical results collected from the airborne radar. It involved modeling the noise floor and level of two distinct environments: the range return of air (i.e. at ranges above ground), and the range return of the ground. For each environment, the noise floor was simulated by adding random noise to match the noise-floor and magnitude found in flight tests. Accordingly, the noise-floor of the ground decreased over range as the radar power return decreases with range.

4.4.2 Hyperbola Estimation using RANSAC

The general conic equation is given by

$$x_1 \mathbf{z}_x{}^2 + x_2 \mathbf{z}_x \mathbf{z}_y + x_3 \mathbf{z}_y{}^2 + x_4 \mathbf{z}_x + x_5 \mathbf{z}_y + x_6 = 0, \tag{4.5}$$

where $\mathbf{x} = \begin{bmatrix} x_1 & x_2 & x_3 & x_4 & x_5 & x_6 \end{bmatrix}^\top$ are the unknown parameters. By definition, a hyperbola must satisfy the following condition,

$$x_2^2 - 4x_1x_3 > 0. \tag{4.6}$$

Using the notation from Section 3.1, we rewrite (4.5) as

$$\begin{bmatrix} \mathbf{z}_x{}^2 & \mathbf{z}_x\mathbf{z}_y & \mathbf{z}_y{}^2 & \mathbf{z}_x & \mathbf{z}_y & 1 \end{bmatrix} \begin{bmatrix} x_1 & x_2 & x_3 & x_4 & x_5 & x_6 \end{bmatrix}^\top = 0$$
$$\phi(\mathbf{z}_x, \mathbf{z}_y)\mathbf{x} = 0. \tag{4.7}$$

Due to the discretization of the range space, the measurements $\mathbf{z}_k = (k, b_k^i)$ do not form a perfect hyperbola, and some sort of smoothing algorithm is needed. Generally, there are two approaches to find the parameters that best fit a data set to a conic equation: minimize the geometric distance, or minimize the algebraic distance. It is known that while the approach of minimizing the geometric distance often yields more accurate results, the process is prohibitively slow to be run in real-time applications [67, 123]. For this reason, we

use algebraic distances, specifically extending the approach developed by Fitzgibbon, et al. to track hyperbolas [55].

In the following, we review Fitzgibbon's method to estimate a hyperbola from noisy measurements. Note that the hyperbola given by (4.4) has no cross term parameter, therefore $x_2 \equiv 0$ in (4.7) and only five measurements are needed in a minimum subset to estimate the five remaining parameters. In order to further reduce the dimensionality of the minimum subsets, we introduce two additional constraints to Fitzgibbon's method using Assumptions 4.1–4.4. We finish by describing how to implement the general RANSAC framework in Algorithm 1 and how to estimate the critical hyperbola parameters V_{g}, $k^i_{\min}$, and $b^i_{\min}$ of the hyperbola in (4.4) characterizing the range progression to a point scatterer from a batch of windowed measurements in clutter.

Fitzgibbon's Method

The general approach of Fitzgibbon et al. is to minimize the sum of squared algebraic distances to the points on the conic by optimizing $\hat{\mathbf{x}}$, while simultaneously ensuring that the hyperbola condition is satisfied [55]. Note that for any scalar $\alpha \neq 0$, the hyperbola $\alpha\mathbf{x}$ is equal to the original hyperbola. This implies that the coefficients can be freely scaled. Fitzgibbon et al. suggest scaling the constraint such that

$$x_2^2 - 4x_1x_3 = 1. \tag{4.8}$$

For now, suppose that the set of measurement pairs $\mathcal{Z}_{k_N:k}$ contains only returns to the i^{th} scatter and clutter. Define the measurement vector

$$Y_{y,k} = \Big[\phi\big(k_N, b_{k_N}\big), \quad \phi\big(k_N+1, b_{k_N+1}\big), \quad \ldots, \phi\big(k, b_k\big)\Big]^\top,$$

which contains the noisy measurements of a single hyperbola over the time window $k_N : k$.

Gander develops techniques to solve an LS problem with a quadratic constraint [60]. Fitzgibbon et al. formulate the hyperbola estimation problem as

$$\begin{aligned} \hat{\mathbf{x}} = \arg\min_{\mathbf{x}} \quad & \|Y_{y,k}\mathbf{x}\|^2 \\ \text{subject to} \quad & \mathbf{x}^\top \mathcal{C}\mathbf{x} = 1, \end{aligned} \tag{4.9}$$

and where the constraint matrix $\mathcal{C}$ is given by

$$\mathcal{C} = \begin{bmatrix} 0 & 0 & -2 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ -2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}.$$

Using Lagrange multipliers, the problem must satisfy the following conditions

$$\begin{aligned} \mathcal{A}\mathbf{x} &= \lambda \mathcal{C}\mathbf{x}, \\ \mathbf{x}^\top \mathcal{C}\mathbf{x} &= 1, \end{aligned} \tag{4.10}$$

where $\mathcal{A} = Y_{y,k}^\top Y_{y,k}$. Using generalized eigenvectors, the problem admits up to six real solutions, but since

$$\|Y_{y,k}\mathbf{x}\|^2 = \mathbf{x}^\top Y_{y,k}^\top Y_{y,k}\mathbf{x} = \mathbf{x}^\top \mathcal{A}\mathbf{x} = \lambda \mathbf{x}^\top \mathcal{C}\mathbf{x} = 1,$$

Fitzgibbon et al. show that the correct solution is the eigenvector corresponding to the minimum eigenvalue. After scaling the parameters to ensure that $\mathbf{x}^\top \mathcal{C}\mathbf{x} = 1$, the result minimizes the algebraic error in (4.9). Halíř uses a very similar technique, but uses the block structure of the matrices to produce a more stable algorithm [67]. For a general hyperbola, six measurements are needed to estimate the six unknown parameters. However, observe that due to Assumptions 4.1–4.4, the cross term parameter $k\, b_k^i$ must always equal zero.

Unfortunately, reformulating both algorithms using $\mathbf{x}' = \begin{bmatrix} x_1 & x_3 & x_4 & x_5 & x_6 \end{bmatrix}^\top$ still resulted in inaccurate estimates due to the discretized range data.

In the following section, we apply Assumptions 4.1–4.4, which constrain the range return hyperbola to a ground scatterer to be symmetric about the axis perpendicular to the flight path. By taking into account the structure of the hyperbola, we introduce two additional constraints that allow efficient and accurate estimates of the hyperbola parameters using only three measurements.

Fitzgibbon's Method with Two Additional Constraints

In this section, we make use of the special hyperbola structure under Assumptions 4.1–4.4. Expanding (4.4) and multiplying by the term $\left(r_{\mathrm{n}} + b^i_{\min}\, r_{\mathrm{r}}\right)^2$, we have

$$\begin{bmatrix} k^2 \\ k\, b^i_k \\ {b^i_k}^2 \\ k \\ b^i_k \\ 1 \end{bmatrix}^\top \begin{bmatrix} -{V_{\mathrm{g}}}^2 \left(\frac{1}{\tau^2}\right) \\ 0 \\ {r_{\mathrm{r}}}^2 \\ 2{V_{\mathrm{g}}}^2 k^i_{\min} \left(\frac{1}{\tau^2}\right) \\ 2r_{\mathrm{n}} r_{\mathrm{r}} \\ {r_{\mathrm{n}}}^2 - \frac{{V_{\mathrm{g}}}^2 {k^i_{\min}}^2}{\tau^2} - \left(r_{\mathrm{n}} + b^i_{\min} r_{\mathrm{r}}\right)^2 \end{bmatrix} = 0$$

$$\phi(k, b^i_k)\mathbf{x} = 0. \tag{4.11}$$

We note that parameters x_3 and x_5 of (4.4) are functions of known parameters, implying

$$\frac{x_3}{{r_{\mathrm{r}}}^2} = \frac{x_5}{2r_{\mathrm{n}} r_{\mathrm{r}}} = 1. \tag{4.12}$$

This leads to the following condition

$$\begin{aligned} 2r_{\mathrm{n}} r_{\mathrm{r}} x_3 - {r_{\mathrm{r}}}^2 x_5 &= 0 \\ 2r_{\mathrm{n}} x_3 - r_{\mathrm{r}} x_5 &= 0 \\ (2r_{\mathrm{n}} x_3 - r_{\mathrm{r}} x_5)^2 &= 0 \\ 4{r_{\mathrm{n}}}^2 x_3^2 + {r_{\mathrm{r}}}^2 x_5^2 - 4r_{\mathrm{n}} r_{\mathrm{r}} x_3 x_5 &= 0. \end{aligned} \tag{4.13}$$

Combining (4.8) and (4.13), we have the constraint

$$x_2^2 - 4x_1x_3 + 4{r_\mathrm{n}}^2x_3^2 + {r_\mathrm{r}}^2x_5^2 - 4r_\mathrm{n}r_\mathrm{r}x_3x_5 = 1. \tag{4.14}$$

Since $x_2 = 0$, we remove it from the parameter vector and form the quadratic constraint $\mathbf{x}'^\top \mathcal{C}'\mathbf{x}' = 1$, where

$$\mathcal{C}' = \begin{bmatrix} 0 & -2 & 0 & 0 & 0 \\ -2 & 4{r_\mathrm{n}}^2 & 0 & -2r_\mathrm{n}r_\mathrm{r} & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & -2r_\mathrm{n}r_\mathrm{r} & 0 & {r_\mathrm{r}}^2 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}. \tag{4.15}$$

With these additional constraints, accurate estimates $\hat{\mathbf{x}}$ of the hyperbola parameters can be achieved with only three measurements.

RANSAC Hyperbola Estimation with SAR Returns

One consequence of the constraint matrix given in (4.15) is that the cardinality of the minimum subset of measurement pairs required to form a model is reduced from five to three. Recall from (2.8) that the convergence of the standard RANSAC algorithm is governed not only by probability p of selecting a measurement of the i^{th} scatterer, but also by the cardinality s of the minimum subset. Note that by decreasing s, the probability of success ζ in (2.8) increases. Let

$$Z_{y,k} = \left[\phi\big(k_N, b^1_{k_N}\big), \quad \ldots, \quad \phi\big(k_N, b^{\psi_{k_N}}_{k_N}\big), \quad \phi\big(k_N+1, b^1_{k_N+1}\big), \quad \ldots, \quad \phi\big(k, b^{\psi_k}_{k}\big)\right]^\top,$$

be the vector of windowed measurements of multiple point scatterers in clutter. A RANSAC hypothesis characterizing the range to a point scatter is found by solving

$$\begin{aligned} \hat{\mathbf{x}}_k \stackrel{\triangle}{=} g(\mathcal{S}_\mathbf{q}) = \arg\min_{\mathbf{x}} \quad & \left\| B(\mathcal{S}_\mathbf{q})\, Z_{y,k}\, \mathbf{x} \right\|^2 \\ \text{subject to} \quad & \mathbf{x}^\top \mathcal{C}' \mathbf{x} = 1, \end{aligned} \tag{4.16}$$

where the binary indicator matrix $B(\mathcal{S}_{\mathbf{q}})$ is defined in (3.4) and the constraint matrix $\mathcal{C}'$ is given in (4.15).

After generating a RANSAC hypothesis, we compute the support of each hypothesis as the number of measurements whose residual is less than a threshold to the hyperbola estimate. Two possible metrics to compute the residual between measurements and expected measurements of a hyperbola are the algebraic distance and the error in expected range. The algebraic distance of the i^{th} measurement to the j^{th} hyperbola estimate is given by $\left|\phi(k, b_k^i)\,\hat{\mathbf{x}}^j\right|$. However, the range tolerance is not uniform as distance to the hyperbola vertex varies. In order to maintain a constant range tolerance for all values along the hyperbola, we compute the predicted range from the hyperbola estimates as

$$\hat{r}(k, \hat{\mathbf{x}}) = \frac{-(x_2k+x_5) + \sqrt{(x_2k+x_5)^2 - 4x_3(x_1k^2+x_4k+x_6)}}{2x_3}. \tag{4.17}$$

The consensus set of the hyperbola described by $\hat{\mathbf{x}}^j$ can then be computed by

$$\chi^j = \mathsf{Inlier}\big(\mathcal{Z}_{k_N:k}, \hat{\mathbf{x}}^j, \tau_{\mathrm{R}},\big) \triangleq \Big\{\mathbf{z}_k^i \in \mathcal{Z}_{k_N:k} : \big\|b_k^i - \hat{r}(k, \hat{\mathbf{x}}^j)\big\|_\infty < \tau_{\mathrm{R}}\Big\}. \tag{4.18}$$

Finally, the purpose of estimating the ellipse given $\mathcal{Z}_{k_N:k}$ is to determine the three unknown parameters, V_{g}, $r_{\min}^i$, and $k_{\min}^i$, characterizing the hyperbola describing a ground scatterer. The estimated values of $\hat{\mathbf{x}}$ can be used to compute the aircraft ground speed and the feature position relative to the aircraft, as

$$V_{\mathrm{g}} = \sqrt{-x_1\tau^2}, \tag{4.19}$$

$$t_{\min}^i = x_4\frac{\tau}{2{V_{\mathrm{g}}}^2}, \tag{4.20}$$

$$r_{\min}^i = \sqrt{{r_{\mathrm{n}}}^2 - {V_{\mathrm{g}}}^2{t_{\min}^i}^2 - (x_6)}. \tag{4.21}$$

4.4.3 Hyperbola Estimation with R-RANSAC

The RANSAC algorithm developed in the previous section can be used within the general R-RANSAC framework described in Algorithm 2. New hypotheses are generated

using (4.16), with consensus set given by (4.18). New measurements are determined to be inliers to existing hyperbola hypotheses according to $\mathsf{Inlier}\big(\mathcal{Z}_k, \hat{\mathbf{x}}^j, \tau_{\mathrm{R}},\big)$, as defined in (4.18). Inliers to existing hyperbola hypotheses use RLS to update the $\mathcal{A}$ matrix in (4.10), which is then used to update the hyperbola parameters after solving the constrained LS problem in (4.16) using Fitzgibbon's method [55].

4.4.4 Results

In this section, we test the R-RANSAC SAR ground scatterer tracking algorithm using both synthetic and real SAR returns. Both simulations are performed with 64-bit 3.0 GHz Intel Core 2 Duo processor running non-optimized MATLAB code.

Synthetic Data

Using synthetic data generated from randomly located ground point scatterers, we compare the hyperbola detection and tracking abilities of both R-RANSAC and the HT. The HT is a voting algorithm and the range, time and velocity parameter space must be divided into discrete bins. The measurement window is 8 seconds long over the 80 second simulation. The range resolution is 0.25 meters, the time resolution is 20 ms, and the velocity resolution is 2 seconds. There is a trade-off between the HT accuracy and the complexity based on the resolution of these parameters.

The R-RANSAC algorithm parameters were defined as follows. The measurement window is defined as $N = 6$ seconds, $\mathcal{M} = 40$, good models are determined when $\rho_k^j > \tau_\rho = 0.25$, and the similarity threshold $\tau_{\mathbf{x}}$ combines models when their ranges are within 2 meters and their minimum times are within 0.5 seconds of each other. The RANSAC parameters are defined as the inlier error threshold $\tau_{\mathrm{R}} = 0.5$ and the maximum number of iterations $\ell = 15$. To increase computational efficiency, we allow early termination from RANSAC if a RANSAC hypothesis has more than γN inliers, where $\gamma = \tau_\rho$.

Figure 4.15 shows the radar returns and the estimated location of the hyperbola vertices for both algorithms over the entire simulation. We find that the HT finds more of the possible scatterers (99%) than R-RANSAC (96%) due to the random sampling of the

R-RANSAC algorithm compared to the brute force voting scheme of the HT. However, a sufficient number of scatterers are found for our future radar odometry application. Most importantly, the R-RANSAC algorithm required an average 0.046 seconds per iteration, while the HT took on average 1.910 seconds per iteration to run—a 40× performance improvement. The accuracy in both range and along-track range is similar, with the Euclidean error being 4.51 meters and 3.80 meters for HT and R-RANSAC, respectively. The HT false alarm rate is 9.2×10^{-3} false alarms per time step compared to 7.9×10^{-2} false alarms per time step for R-RANSAC. The slightly higher false alarm rate of R-RANSAC is mostly due to hyperbola estimates formed by temporarily combining two hyperbolas into one, as seen at coordinates $k_{\min} = 63$ and $r_{\min} = 500$. We expect that these occurrences can be easily filtered by considering the persistence of the estimates.

Real Data

We use an LFM-CW NanoSARC® [78] to collect real SAR returns. The NanoSARC® is well-suited for light unmanned air vehicles, weighing 2.6 lb and requiring less than 25 W total power. The radar resolution is 0.3 meters, with a 1 kHz pulse repetition frequency. The radar was mounted in a Cessna flying at around 40 m/s at around 160 meters above ground level, with nadir visible at all times.

Real SAR returns are much more difficult to process due to the complexity in the environment and radar path. As environmental point scatterers are used, it is difficult to determine truth data for comparison. Additionally, radar returns from ground scatterers are corrupted due to signal multi-path, and reflections from non-spherical and closely-spaced scatterers. This causes blurring in the return and can make scatterer differentiation difficult. To assist the tracking algorithms, the raw SAR returns are filtered to remove some of the clutter and dim scatterers. The filtering removes the average from a 9×13 pixel window. Additionally a weighted above-ground-level range average and chirp averages are removed.

The time and velocity resolutions of the HT discretization remain the same, but the range resolution is increased to 0.2998 meters. The measurement window is 8 seconds long. The R-RANSAC algorithm parameters were defined as follows. The measurement window is also defined as $N = 8$ seconds, $\mathcal{M} = 60$, good models are determined when $\rho_k^j > \tau_\rho = 0.3$,

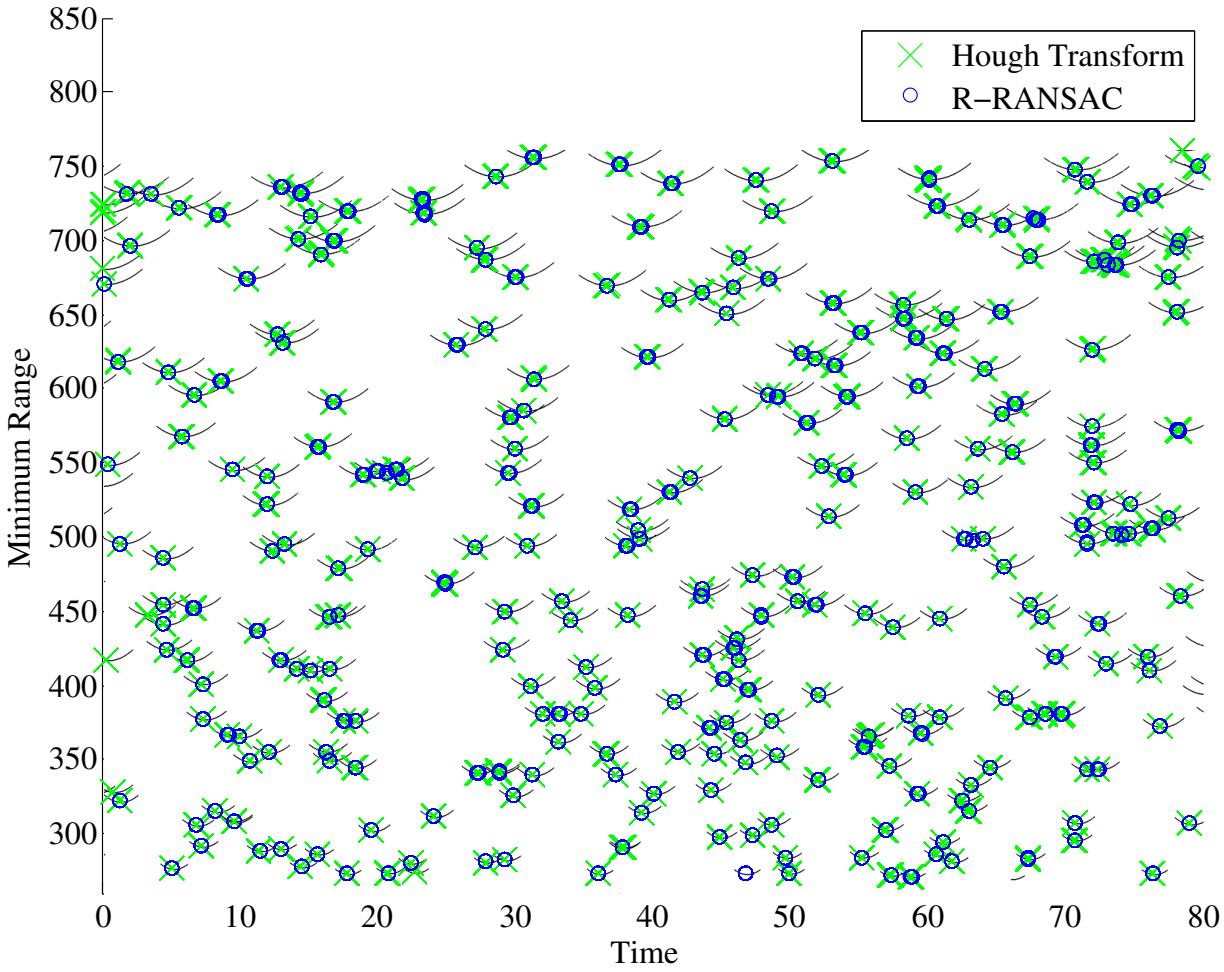

Figure 4.15: Simulated radar returns of ground point scatterers and estimated scatterer location using the HT and R-RANSAC algorithms.

and the similarity threshold $\tau_{\mathbf{x}}$ combines models when their ranges are within 5 meters and their minimum times are within 1.5 seconds of each other. The RANSAC parameters are defined as $\tau_{\mathrm{R}} = 1.5$, $\ell = 50$, $\gamma = \tau_{\rho}$. Due to the large number of measurements per time step, we processed every third measurement to improve the performance. This is reasonable simplification since multiple measurements of the same scan often measure the same scatterer due to multi-path blurring. The RANSAC framework internal to R-RANSAC is modified to use the consensus set of the best hypothesis and iterate three times to refine the hyperbola parameters and consensus set; this greatly improved the performance when multiple radar returns measure a single signal.

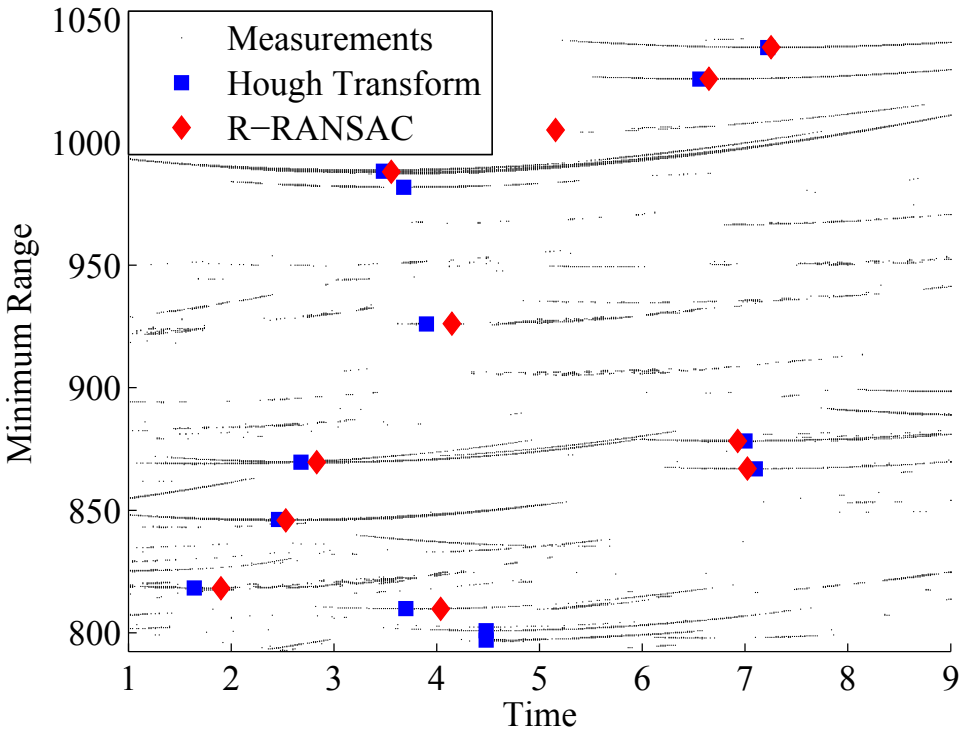

Figure 4.16: Snapshot of measurement window of previous N SAR scans. The hyperbola center estimates of the HT and R-RANSAC are shown.

Since the HT is a brute force voting algorithm, we expect it to outperform R-RANSAC by finding more scatterers to track. In general, this is true. However, as shown in Figure 4.16, we see that while R-RANSAC did not detect all possible scatterers, neither did the HT. Since there is no ground truth, it is difficult to compare the accuracy of the HT compared to R-RANSAC, but we see that in most cases the estimates are similar.

The improvement of R-RANSAC over the HT implementation of the recursive hyperbola detector is in computational complexity. The HT required an average of 15.3 s per iteration, or 0.066 Hz. R-RANSAC required an average of 0.799 s per iteration, or 1.25 Hz, which is almost 20 times faster.

Chapter 5. Tracking Multiple Dynamic Targets

Multiple target tracking (MTT) scenarios typically require estimating the number of targets present in a measurement sequence and estimating the trajectory of those targets over time. A classical example of MTT is where an unknown number of vehicles enter and leave a camera field of view, and we desire to continuously track the position of each vehicle while distinguishing it from other vehicles within the camera field of view. As discussed in Chapter 2, the difficulty of MTT is to identify the correct data association and differentiate between true measurements of multiple targets and spurious measurements, or clutter. There is no labeling on the measurements, so the data association problem is NP-hard [68].

Since at least the 1960s [146], various solutions to the MTT problem have been proposed. A high-level discussion and references to several existing multiple target algorithms are presented in Chapter 1. The primary focus of this chapter is to develop a robust MTT algorithm that is simple, computationally efficient, and fully autonomous while maintaining accuracy. The recursive-RANSAC (R-RANSAC) framework is well-suited for this task.

While not a cure-all, R-RANSAC does offer several advantages over the existing algorithms within the MTT literature. Where the GNN filter diverges in clutter, R-RANSAC relies on the robust qualities of RANSAC [53] to mitigate the effects of spurious measurements. Where the JPDA filter requires a separate track management system, R-RANSAC automatically generates and removes hypothesis tracks while simultaneously identifying good tracks describing probable targets. Where the MHT filter exhaustively enumerates possible measurement or track hypotheses, R-RANSAC stores the most probable set of measurement-to-track hypotheses. Where the PHD filter does not perform well in low probability of detection scenarios, R-RANSAC uses a sliding measurement window that facilitates tracking low detection rate targets.

In this chapter, we describe the R-RANSAC algorithm for tracking multiple dynamic targets. After formulating the problem for linear time-invariant systems, this chapter is divided into four main sections. First, R-RANSAC is developed to estimate the states of multiple dynamic targets, which was first reported in [116]. Second, we account for possibly known inputs when R-RANSAC is utilized as a robust observer. Third, we discuss techniques for tracking maneuvering targets. Specifically, we first develop a maximum likelihood technique to estimate the inputs of a maneuvering target using minimum subsets; we also introduce the interacting multiple model (IMM) variant of the R-RANSAC algorithm. Finally, a theoretical analysis of R-RANSAC is explored and we show that R-RANSAC hypothesis tracks converges in mean to true state estimates.

5.1 Problem Description

Let $\mathbf{x}_k \in \mathbb{R}^n$ represent the states of a dynamic system at time k, and let $\mathbf{u}_k \in \mathbb{R}^r$ be the control vector at time k. In this chapter, the general state dynamics in (2.1) are assumed to be linear, and are modeled as

$$\mathbf{x}_k = A\mathbf{x}_{k-1} + B\mathbf{u}_k + \mathbf{w}_k, \tag{5.1}$$

where $A \in \mathbb{R}^{n\times n}$, $B \in \mathbb{R}^{n\times r}$, and where $\mathbf{w}$ is the process noise and is a wide-sense stationary (WSS), zero-mean Gaussian random process with covariance Q. In non-cooperative tracking scenarios the control vector is unknown; in such situations, a kinematic state model is used, where the derivative of the lowest-order state is approximated as zero-mean white noise. For example, ground targets tracked by an unmanned air system are well-modeled using a constant jerk model [107]. The resulting state dynamics are given by

$$\mathbf{x}_k = A\mathbf{x}_{k-1} + \mathbf{w}_k, \tag{5.2}$$

where the unknown maneuvers are captured by inflating the process noise [14, 19].

The output $\mathbf{y}_k \in \mathbb{R}^m$ of the signal at time k is measured according to a random process $\mathcal{P}_k(\omega)$, where $\omega \in \Omega = \{0, 1\}$ and $\omega = 1$ represents a valid measurement of the true

signal. Mathematically, the linear variant of the general output in (2.2) is given by

$$\mathbf{y}_k = \begin{cases} C\mathbf{x}_k + \mathbf{v}_k & \text{when } \omega = 1 \\ \emptyset & \text{when } \omega = 0, \end{cases} \tag{5.3}$$

where $C \in \mathbb{R}^{m \times n}$, and $\mathbf{v}$ is the measurement noise that follows a WSS, zero-mean Gaussian random process with covariance R. The probability distribution on the events in $\mathcal{P}_k(\omega)$ and the faulty measurement distribution is problem specific. We assume that the system (A, C) is observable and, when using (5.1), that the system (A, B) is controllable.

5.2 Multiple Target Tracking using R-RANSAC

When a tracking system does not have knowledge of the target control inputs, as modeled in (5.2), we design the R-RANSAC algorithm in the context of MTT in clutter. This section is divided into two sections: first, we derive a RANSAC-based maximum likelihood estimate (MLE) of a trajectory hypothesis from a window of measurements, and second, we formulate the full R-RANSAC algorithm.

5.2.1 Trajectory Estimation using RANSAC

In this section, we describe RANSAC-based trajectory estimation in two steps. First, we develop a MLE of the initial conditions of a target trajectory when there is no clutter. Second, we incorporate clutter and use the concept of RANSAC minimum subsets to estimate a hypothesis trajectory of an underlying dynamic target.

Single Target without Clutter

Consider the scenario when there is a single target that is measured with probability $p_{\mathrm{D}} = 1$ and where there is no clutter, i.e. $\lambda_{\mathrm{c}} = 0$. Under these assumptions, there is exactly one (noisy) measurement per time step that corresponds to the target of interest. Our first objective is to estimate the initial conditions of the states $\hat{\mathbf{x}}_{k_N}$ and covariance matrix $P_{k_N} \in \mathbb{R}^{n \times n}$. Second, by propagating the initial conditions with a Kalman filter, we can

estimate the trajectory $\hat{\mathbf{x}}_{k_N:k}$. Consider a sequence of N measurements characterized by

$$\begin{bmatrix} \mathbf{y}_{k_N} \\ \mathbf{y}_{k_N+1} \\ \vdots \\ \mathbf{y}_k \end{bmatrix} = \begin{bmatrix} C\mathbf{x}_{k_N} \\ C\mathbf{x}_{k_N+1} \\ \vdots \\ C\mathbf{x}_k \end{bmatrix} + \begin{bmatrix} \mathbf{v}_{k_N} \\ \mathbf{v}_{k_N+1} \\ \vdots \\ \mathbf{v}_k \end{bmatrix}. \tag{5.4}$$

Using (5.2), we write (5.4) in terms of $\mathbf{x}_{k_N}$, as

$$\begin{bmatrix} \mathbf{y}_{k_N} \\ \mathbf{y}_{k_N+1} \\ \vdots \\ \mathbf{y}_k \end{bmatrix} = \begin{bmatrix} C \\ CA \\ \vdots \\ CA^{N-1} \end{bmatrix} \mathbf{x}_{k_N} + \begin{bmatrix} 0 & 0 & 0 & \dots & 0 \\ 0 & C & 0 & \dots & 0 \\ 0 & CA & C & & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & CA^{N-2} & CA^{N-3} & \dots & C \end{bmatrix} \begin{bmatrix} \mathbf{w}_{k_N} \\ \mathbf{w}_{k_N+1} \\ \mathbf{w}_{k_N+2} \\ \vdots \\ \mathbf{w}_k \end{bmatrix} + \begin{bmatrix} \mathbf{v}_{k_N} \\ \mathbf{v}_{k_N+1} \\ \vdots \\ \mathbf{v}_k \end{bmatrix}. \tag{5.5}$$

Define

$$Y_k = \left[\mathbf{y}_{k_N}, \mathbf{y}_{k_N+1}, \dots, \mathbf{y}_k\right]^\top, \tag{5.6}$$

$$W_k = \left[\mathbf{w}_{k_N}, \mathbf{w}_{k_N+1}, \dots, \mathbf{w}_k\right]^\top, \tag{5.7}$$

$$V_k = \left[\mathbf{v}_{k_N}, \mathbf{v}_{k_N+1}, \dots, \mathbf{v}_k\right]^\top, \tag{5.8}$$

$$\mathcal{O} = \left[C, \ \ CA, \ \ \dots, \ \ CA^{N-1}\right]^\top, \tag{5.9}$$

and

$$G = \begin{bmatrix} 0 & 0 & 0 & \dots & 0 \\ 0 & C & 0 & \dots & 0 \\ 0 & CA & C & & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & CA^{N-2} & CA^{N-3} & \dots & C \end{bmatrix}, \tag{5.10}$$

where $G \in \mathbb{R}^{mN \times nN}$ describes the propagation of the process noise, and $\mathcal{O} \in \mathbb{R}^{mN \times n}$ characterizes an expected trajectory given the initial states $\mathbf{x}_{k_N}$. Note that $\mathcal{O}$ is the observability matrix if $N = n$.

Defining

$$\xi_k = GW_k + V_k \tag{5.11}$$

as the combined propagated process noise and the measurement noise, we can write (5.5) as

$$Y_k = \mathcal{O}\mathbf{x}_{k_N} + \xi_k. \tag{5.12}$$

Since $\mathbf{w}$ and $\mathbf{v}$ are i.i.d. and zero-mean Gaussian, then ξ_k is also zero-mean Gaussian with a time-invariant covariance Ξ given by

$$\begin{aligned} \Xi &= E\big[\xi_k \xi_k^\top\big] \\ &= GE\big[W_k W_k^\top\big] G^\top + E\big[V_k V_k^\top\big]. \end{aligned} \tag{5.13}$$

Define $\mathbf{Q} = E\big[W_k W_k^\top\big]$ and $\mathbf{R} = E\big[V_k V_k^\top\big]$ where, due to the assumed independence of the process and measurement noise, $\mathbf{Q}$ and $\mathbf{R}$ are block diagonal matrices. Using this notation, (5.13) can be written as

$$\Xi = G\mathbf{Q}G^\top + \mathbf{R}. \tag{5.14}$$

Briefly consider the scenario where there is no process noise, i.e., $\mathbf{w}_k = 0,\ \forall k$. In this case, (5.12) reduces to $Y_k = \mathcal{O}\mathbf{x}_{k_N} + \mathbf{R}$, and assuming $\mathcal{O}$ is full rank, the initial states are found using least-squares (LS), as

$$\hat{\mathbf{x}}_{k_N} = (\mathcal{O}^\top \mathcal{O})^{-1} \mathcal{O}^\top Y_k. \tag{5.15}$$

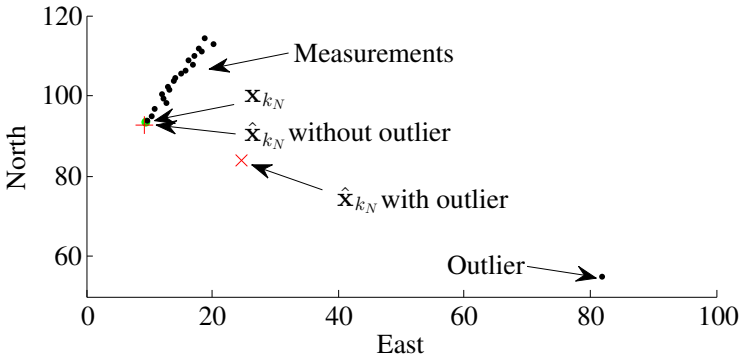

Figure 5.1: Maximum likelihood estimate of trajectory initial conditions diverges with a single spurious measurement.

By observability, the matrix $(\mathcal{O}^\top \mathcal{O})$ is full rank if $N \geq n$. When including the process noise, the MLE of $\mathbf{x}_{k_N}$ and the covariance P_{k_N} are derived in Appendix B and are given by

$$\hat{\mathbf{x}}_{k_N} = \left(\mathcal{O}^\top \Xi^{-1} \mathcal{O}\right)^{-1} \mathcal{O}^\top \Xi^{-1} Y_k, \tag{5.16}$$

$$P_{k_N} = \left(\mathcal{O}^\top \Xi^{-1} \mathcal{O}\right)^{-1}. \tag{5.17}$$

Note that (5.16) is similar to the psuedoinverse in (5.15), but weighted by the process noise covariance.

After computing the initial conditions in (5.16) and (5.17), the Kalman filter is used to estimate the current states. The initial states and covariance are propagated forward and updated using the measurements $\{\mathbf{y}_{k_N}, \ldots, \mathbf{y}_k\}$ for each subsequent time step. The state estimates over the measurement window comprise the estimated trajectory $\hat{\mathbf{x}}_{k_N:k}$.

Unfortunately, even a single false measurement may cause the MLE of the initial states to diverge. A simple example is shown in Fig. 5.1, and is analogous to Fischler and Bolles' demonstration that one false measurement can corrupt the LS solution [53]. In the following, we use a minimum subset of data to find the initial states of a dynamic target, and subsequently use RANSAC to robustly estimate the target track of a particular measurement.

Multiple Targets in Clutter

Let the number of true targets be $M_k \geq 1$ and the clutter rate $\lambda_c \geq 0$. Under these conditions, multiple measurements are received at each time step. The first objective is to estimate the initial states $\mathbf{x}_{k_N}^i$, for all targets $i = 1, \ldots, M_k$. We do so by applying the RANSAC paradigm and using only a minimal number of randomly selected measurements s to estimate the initial states, where s is defined as the observability index of the system. The observability index is determined by finding the smallest natural number s such that $\text{rank}\,(\mathcal{O}_s) = \text{rank}\,(\mathcal{O}_{s+1})$, where

$$\mathcal{O}_s = \begin{bmatrix} C, & CA, & \ldots, & CA^{s-1} \end{bmatrix}^\top.$$

Second, we propagate the initial conditions forward in time using a Kalman filter to obtain an estimate of the trajectory $\hat{\mathbf{x}}_{k_N:k}^i$.

Define the matrix $\Phi_k \in \mathbb{R}^{m\Psi_{k_N:k} \times mN}$ to associate the correct time indices to all the measurements received during that time step. Let $\mathbf{1}_{\psi_k}$ be defined as a column vector of ψ_k ones, and let I_m be defined as an $m \times m$ identity matrix. Also, let the $\otimes$ operator be the standard Kronecker product. Define

$$\Phi_k = \begin{bmatrix} \mathbf{1}_{\psi_{k_N}} \otimes I_m & \mathbf{0} & \ldots & \mathbf{0} \\ \mathbf{0} & \mathbf{1}_{\psi_{k_N+1}} \otimes I_m & \ddots & \vdots \\ \vdots & \ddots & \ddots & \mathbf{0} \\ \mathbf{0} & \ldots & \mathbf{0} & \mathbf{1}_{\psi_k} \otimes I_m \end{bmatrix}. \tag{5.18}$$

With Φ_k, the vector of windowed measurements,

$$Z_k = \begin{bmatrix} \mathbf{z}_{k_N}^1, & \ldots, & \mathbf{z}_{k_N}^{\psi_{k_N}}, & \mathbf{z}_{k_N+1}^1, & \ldots, & \mathbf{z}_k^{\psi_k} \end{bmatrix}^\top, \tag{5.19}$$

can be expressed as

$$\begin{aligned} Z_k &= \Phi_k \mathcal{O} \mathbf{x}_{k_N}^i + \Phi_k G_k W_k + \Phi_k V_k \\ &= \Phi_k \mathcal{O} \mathbf{x}_{k_N}^i + \Phi_k \xi_k. \end{aligned} \tag{5.20}$$

An estimate of the initial conditions can be found utilizing the concept of minimum subsets to modify both (5.16) and (5.17). Define $\mathcal{S}_{\mathbf{q}} \in \mathbb{S}^{s}_{k_N:k}$ as a minimum subset of s measurements randomly selected from the set of measurement combinations, where $\mathbf{q}$ defines the indices of the selected measurements. To improve performance, we constrain $\mathcal{S}_{\mathbf{q}}$ to contain no more than one measurement per scan. We define a binary indicator matrix that allows us to select specific measurements of interest from the measurement vector Z_k. In general, define $\mathbf{d} = [\mathbf{d}_1, \ldots, \mathbf{d}_d]^\top$ to be a vector where each element is an index to a specific measurement in Z_k. Let $\mathcal{B}(\mathbf{d}) \in \mathbb{R}^{md \times m\Psi_{k_N:k}}$ be defined as a binary indicator matrix, given by

$$\mathcal{B}(\mathbf{d}) = \begin{bmatrix} \mathbf{0}_{m\times m(\mathbf{d}_1-1)} & I_m & \mathbf{0}_{m\times m(\Psi_{k_N:k}-\mathbf{d}_1)} \\ \vdots & \vdots & \vdots \\ \mathbf{0}_{m\times m(\mathbf{d}_d-1)} & I_m & \mathbf{0}_{m\times m(\Psi_{k_N:k}-\mathbf{d}_d)} \end{bmatrix}. \tag{5.21}$$

Starting with (5.20), the binary indicator matrix $\mathcal{B}(\mathbf{q})$ is pre-multiplied on both sides. Letting $\bar{Z}_k = \mathcal{B}(\mathbf{q})\, Z_k$, $\bar{\mathcal{O}} = \mathcal{B}(\mathbf{q})\, \Phi_k\, \mathcal{O}$, and $\bar{\xi}_k = \mathcal{B}(\mathbf{q})\, \Phi_k\, \xi_k$, we have

$$\bar{Z}_k = \bar{\mathcal{O}}\mathbf{x}_{k_N} + \bar{\xi}_k, \tag{5.22}$$

where the noise term $\bar{\xi}_k$ has covariance $\bar{\Xi}$ given by

$$\begin{aligned} \bar{\Xi} &= E\Big[\bar{\xi}_k\, \bar{\xi}_k^\top\Big] \\ &= \mathcal{B}(\mathbf{q})\, \Phi_k\, E\big[\xi_k\, \xi_k^\top\big]\, \Phi_k^\top\, \mathcal{B}(\mathbf{q})^\top \\ &= \mathcal{B}(\mathbf{q})\, \Phi_k\, \Xi\, \Phi_k^\top\, \mathcal{B}(\mathbf{q})^\top. \end{aligned} \tag{5.23}$$

The MLE of the initial states and covariance is given by

$$\hat{\mathbf{x}}_{k_N}(\mathbf{q}) = \left(\bar{\mathcal{O}}^\top \bar{\Xi}^{-1} \bar{\mathcal{O}}\right)^{-1} \bar{\mathcal{O}}^\top \bar{\Xi}^{-1} \bar{Z}_k, \tag{5.24}$$

$$P_{k_N}(\mathbf{q}) = \left(\bar{\mathcal{O}}^\top \bar{\Xi}^{-1} \bar{\mathcal{O}}\right)^{-1}. \tag{5.25}$$

The hypothesis track during the time window $k_N : k$ is found by propagating (5.24) forward in time, according to $\hat{\mathbf{x}}_k = A\, \hat{\mathbf{x}}_{k-1}$. Let $g_k(\mathcal{S}_{\mathbf{q}})$ represent the function mapping a minimal

subset of measurements with indices $\mathbf{q}$ to an estimate of the state at time k. Therefore, $g_k(\mathcal{S}_\mathbf{q})$ is given by propagating the estimated initial states of a trajectory to time k, as

$$\hat{\mathbf{x}}_k(\mathbf{q}) = g_k(\mathcal{S}_\mathbf{q}) \triangleq A^{N-1}\,\hat{\mathbf{x}}_{k_N}(\mathbf{q}), \tag{5.26}$$

where the initial states are given by (5.24) The consensus set to the hypothesis track with initial states given by (5.24) can be found by finding the measurements whose residual is less than a threshold τ_R,

$$\begin{aligned}\chi_k(\mathbf{q}) &= \mathsf{Inlier}\big(\mathcal{Z}_{k_N:k}, \hat{\mathbf{x}}_{k_N}(\mathbf{q}), \tau_\mathrm{R},\big) \\ &= \Big\{\mathbf{z}_\kappa \in \mathcal{Z}_\kappa \;:\; \big\|\mathbf{z}_\kappa - CA^{\kappa-1}\hat{\mathbf{x}}_{k_N}(\mathbf{q})\big\|_R < \tau_\mathrm{R},\ \kappa = k_N \colon k\Big\}.\end{aligned} \tag{5.27}$$

We summarize by combining the preceding equations with the RANSAC algorithm described in Algorithm 1 in Chapter 2. Hypotheses are generated in Line 3 by finding the initial conditions given by (5.24) and (5.25). Hypothesis validation in Line 4 occurs by finding the hypothesis with the largest consensus set $|\chi_k(\mathbf{q})|$, which are given by (5.27). The final step of RANSAC in Line 12 is to smooth over the consensus set, where we propagate the initial conditions of the best hypothesis forward in time and update with a standard Kalman filter using the measurements in the consensus set.

5.2.2 Trajectory Estimation using R-RANSAC

This section describes how to wrap the RANSAC-based trajectory estimation discussed in Section 5.2.1 into the R-RANSAC framework described in Algorithm 2 in Chapter 2 to recursively track multiple targets in clutter. The first step in Line 2 of Algorithm 2 is to propagate the state estimates of all active tracks, which in this case is done using the prediction step of the Kalman filter. Line 3 determines which current measurements are inliers to the existing tracks, and Line 4 determines the measurement-to-track weighting w_{ij} based on the chosen association scheme. Recall from Section 2.3 that there are several available techniques for data association: hard association techniques such as nearest neighbor or all-neighbor, or soft association techniques such as PDA and PMHT. If a

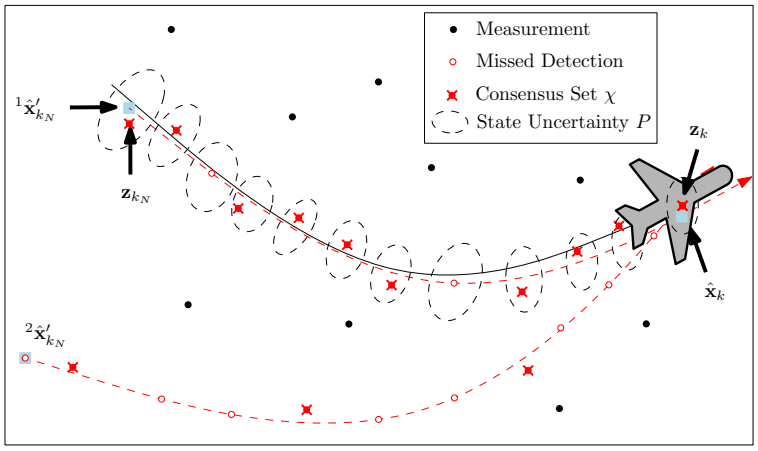

Figure 5.2: R-RANSAC hypothesis trajectories ${}^1\hat{\mathbf{x}}'$ and ${}^2\hat{\mathbf{x}}'$ are generated to fit the current measurement $\mathbf{z}_k$. The hypothesis described by ${}^1\hat{\mathbf{x}}'$ is selected as the best track since it has ten inliers, as opposed to four for the other hypothesis. The final estimate is smoothed by propagating the initial states ${}^1\hat{\mathbf{x}}'_{k_N}$ using a Kalman filter and updating as necessary using the associated inliers.

measurement is an outlier to all existing tracks, then a RANSAC hypothesis is generated using (5.24), (5.25), and (5.27). The only difference is that minimum subsets are selected from the set $\mathcal{S}_{\bar{\mathbf{q}}} \in \left\{\mathbb{S}^{s-1}_{k_N:k-1} \times \{\mathbf{z}^i_k\}\right\}$, where $\bar{\mathbf{q}}$ is a uniformly distributed random index over the set of $\binom{\Psi_{k_N:k-1}}{s-1}$ minimum subsets. In essence, we are fitting a trajectory to the windowed data that fits the outlier. Finally, existing tracks are updated by the weighted inlier measurements as described in Appendix A.

Figure 5.2 gives a simple example. Suppose we want to fit the current measurement $\mathbf{z}_k$ to previous data where we assume that the target dynamics are described by a nearly-constant acceleration model. In this case, three measurements are needed to estimate the initial states at time k_N. For each hypothesis, two additional measurements are randomly selected from previous measurements to form a minimum subset, which is used to estimate the initial states $\hat{\mathbf{x}}'_{k_N}$. Two hypotheses are considered, where the hypothesis with the most inliers is selected to be the most accurate. The final estimate $\hat{\mathbf{x}}_k$ is found by propagating the initial states, updating as necessary using the associated inlier measurements with a Kalman filter.

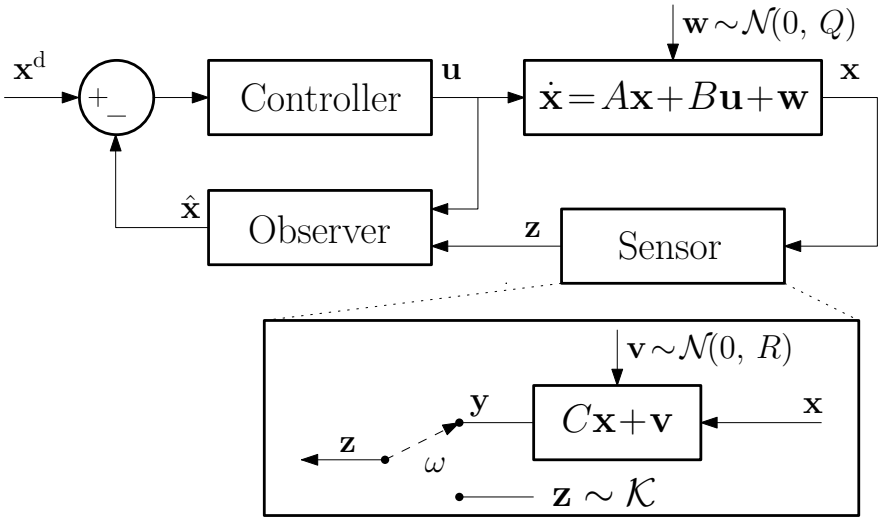

Figure 5.3: Architecture of a feedback system given a faulty sensor.

5.3 R-RANSAC Tracking with Known Inputs

In the beginning of the chapter, we originally formulated the tracking problem in (5.1) as a function of the inputs, or commands, to the system. However, in the previous section we assume that these inputs are unknown and instead use (5.2) to model the target dynamics. In this section, we examine the case where the inputs in (5.1) are known and the objective is to use R-RANSAC as a robust observer within a feedback loop. In these scenarios, there is typically only a single signal to track, but there may be faulty measurements due to sensor failures or multi-path. Figure 5.3 shows an overview of the general system architecture. The controller's objective is to determine the proper input vector $\mathbf{u}$ to achieve some desired state $\mathbf{x}^{\mathrm{d}}$.

As in the previous section, we initially assume a clutter-free sensor $p_{\mathrm{D}} = 1$ and $\lambda_{\mathrm{c}} = 0$ such that N measurements have been received. Our objective is to use maximum likelihood to estimate the initial states while accounting for the input vector,

$$U_k = [\mathbf{u}_{k_N+1},\, \ldots,\, \mathbf{u}_k]^\top. \tag{5.28}$$

Define the input propagation matrix as

$$H = \begin{bmatrix} 0 & 0 & \dots & 0 \\ CB & 0 & \dots & 0 \\ CAB & CB & \ddots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ CA^{N-2}B & CA^{N-3}B & \dots & CB \end{bmatrix}, \tag{5.29}$$

where $H \in \mathbb{R}^{mN \times r(N-1)}$. Using (5.6)–(5.10), the measurement vector can be expressed in terms of the initial states as

$$\begin{aligned} Y_k &= \mathcal{O}\mathbf{x}_{k_N} + HU_k + GW_k + V_k \\ Y_k - HU_k &= \mathcal{O}\mathbf{x}_{k_N} + \xi_k, \end{aligned} \tag{5.30}$$

where the combined propagated process noise and measurement noise ξ_k is given by (5.11) with covariance given in (5.14).

Since the terms on the left hand side of (5.30) are known, we can estimate the initial states $\mathbf{x}_{k_N}$ using MLE as described in Appendix B. The MLE of the initial states are given by

$$\hat{\mathbf{x}}_{k_N} = \left(\mathcal{O}^\top \Xi^{-1} \mathcal{O}\right)^{-1} \mathcal{O}^\top \Xi^{-1} (Y_k - H\, U_k). \tag{5.31}$$

Using the matrices Φ_k and $\mathcal{B}(\mathcal{S}_\mathbf{q})$ as defined in (5.18) and (5.21), respectively, minimum subsets of the windowed measurements in (5.19) can be used to estimate the initial states, such that

$$\hat{\mathbf{x}}_{k_N}(\mathbf{q}) = \left(\bar{\mathcal{O}}^\top \bar{\Xi}^{-1} \bar{\mathcal{O}}\right)^{-1} \bar{\mathcal{O}}^\top \bar{\Xi}^{-1} \left(\bar{Z}_k - \bar{H} U_k\right), \tag{5.32}$$

where $\bar{H} = \mathcal{B}(\mathbf{q})\, \Phi_k\, H$. The initial covariance $P_{k_N}(\mathbf{q})$ is equal to the initial covariance computed previously in (5.25), and the consensus set of each hypothesis is given by modifying (5.27) to account for the known inputs, as

$$\chi_k(\mathbf{q}) = \left\{ \mathbf{z}_\kappa \in \mathcal{Z}_\kappa \ : \ \left\| \mathbf{z}_\kappa - (CA^{\kappa-1}\hat{\mathbf{x}}_{k_N}(\mathbf{q}) + B\mathbf{u}_\kappa) \right\|_R < \tau_\mathrm{R},\ \kappa = k_N \colon k \right\}. \tag{5.33}$$

The only change to the R-RANSAC algorithm is in Line 2 of Algorithm 2, where the existing states are propagated forward in time. In this case, the known inputs are included in the prediction step of the Kalman filter. The remainder of the R-RANSAC algorithm is the same as in Section 5.2.2.

5.4 R-RANSAC Tracking of Multiple Maneuvering Targets

In this section, consider the problem of tracking multiple *maneuvering* targets using the R-RANSAC algorithm. As in the previous sections, we divide the problem into two components. First, assuming the target dynamics are given by (5.1), we estimate both the initial conditions of a hypothesis trajectory and a piecewise-constant input vector to estimate target maneuvers during the measurement window. Second, after reviewing the IMM filter [21], we develop the IMM R-RANSAC algorithm to recursively track multiple maneuvering targets in clutter.

5.4.1 RANSAC with Input Estimation

In this section we outline another variation of the RANSAC-based trajectory estimation technique. In Sections 5.2 and 5.3, we assumed that the target inputs were either zero-mean or known to estimate a trajectory using minimum subsets over a time window. In this section, we account for possible target maneuvers by estimating a piecewise-constant input vector describing the controls during the measurement window. First we describe estimating the initial conditions and full input vector using LS when there is no clutter. We then use the concept of minimum subsets to estimate the initial conditions and a piecewise-constant input vector.

Single Target without Clutter

We initially assume a clutter-free sensor, where $p_{\mathrm{D}} = 1$ and $\lambda_{\mathrm{c}} = 0$, such that N measurements have been received. Our objective is to estimate both the initial conditions of a trajectory as well as the vector of unknown inputs using MLE. We again write a set of measurements using (5.1) in matrix notation, only this time we augment the initial states of

the trajectory with the input vector to the system during the time window, given by

$$Y_k = \begin{bmatrix} C & 0 & 0 & \dots & 0 \\ CA & CB & 0 & \dots & 0 \\ CA^2 & CAB & CB & \ddots & \vdots \\ \vdots & \vdots & \vdots & \ddots & 0 \\ CA^{N-1} & CA^{N-2}B & CA^{N-3}B & \dots & CB \end{bmatrix} \begin{bmatrix} \mathbf{x}_0 \\ \mathbf{u}_{k_N+1} \\ \mathbf{u}_{k_N+2} \\ \vdots \\ \mathbf{u}_k \end{bmatrix} + GW_k + V_k, \tag{5.34}$$

where Y_k, G, W_k, and V_k are defined in (5.6)–(5.10). Unfortunately, (5.34) now requires a measurement for every time step in order to estimate the initial states and input vector, i.e. the observability index is $s = N$. Not only is the maximum likelihood solution to (5.34) computationally intensive, there is no mechanism to reduce that complexity since the cardinality of the minimum subsets is $s = N$. In fact, (5.34) will fail whenever $p_{\mathrm{D}} < 1$ since there will be missing measurements.

In practical tracking scenarios, targets often apply a particular input for several time steps. We therefore approximate the input vector as a piecewise-constant function of $n_{\mathbf{u}}$ inputs $U_k = [\mathbf{u}^0, \dots, \mathbf{u}^{n_{\mathbf{u}}-1}]$, where the input at time κ is

$$\mathbf{u}_\kappa = \begin{cases} \mathbf{u}^0 & \text{if } \kappa < k_{\mathbf{u}^1} \\ \mathbf{u}^1 & \text{if } k_{\mathbf{u}^1} \le \kappa < k_{\mathbf{u}^2} \\ \vdots & \\ \mathbf{u}^{n_{\mathbf{u}}-1} & \text{if } k_{\mathbf{u}^{n_{\mathbf{u}}-1}} \le \kappa < k. \end{cases} \tag{5.35}$$

Re-deriving (5.34) under this simplification gives

$$Y_k = \underbrace{\begin{bmatrix} C & 0 & 0 & \dots & 0 \\ CA & CB & 0 & \dots & 0 \\ CA^2 & CAB+CB & 0 & \dots & 0 \\ \vdots & \vdots & \vdots & & \vdots \\ CA^{k_{\mathbf{u}_1}-k_N-1} & \sum_{\kappa=0}^{k_{\mathbf{u}_1}-k_N-2} CA^\kappa B & 0 & \dots & 0 \\ CA^{k_{\mathbf{u}_1}-k_N} & \sum_{\kappa=1}^{k_{\mathbf{u}_1}-k_N-1} CA^\kappa B & CB & \dots & 0 \\ CA^{k_{\mathbf{u}_1}-k_N+1} & \sum_{\kappa=2}^{k_{\mathbf{u}_1}-k_N} CA^\kappa B & CAB+CB & \dots & 0 \\ \vdots & \vdots & \vdots & & \vdots \\ CA^{k_{\mathbf{u}_2}-k_N-1} & \sum_{\kappa=k_{\mathbf{u}_2}-k_{\mathbf{u}_1}}^{k_{\mathbf{u}_2}-k_N-2} CA^\kappa B & \sum_{\kappa=0}^{k_{\mathbf{u}_2}-k_{\mathbf{u}_1}-1} CA^\kappa B & \dots & 0 \\ CA^{k_{\mathbf{u}_2}-k_N} & \sum_{\kappa=k_{\mathbf{u}_2}-k_{\mathbf{u}_1}+1}^{k_{\mathbf{u}_2}-k_N-1} CA^\kappa B & \sum_{\kappa=1}^{k_{\mathbf{u}_2}-k_{\mathbf{u}_1}} CA^\kappa B & \dots & CB \\ CA^{k_{\mathbf{u}_2}-k_N+1} & \sum_{\kappa=k_{\mathbf{u}_2}-k_{\mathbf{u}_1}+2}^{k_{\mathbf{u}_2}-k_N} CA^\kappa B & \sum_{\kappa=2}^{k_{\mathbf{u}_2}-k_{\mathbf{u}_1}+1} CA^\kappa B & \dots & CAB+CB \\ \vdots & \vdots & \vdots & & \vdots \\ CA^{k-k_N} & \sum_{\kappa=k-k_{\mathbf{u}_1}+1}^{k-k_N-1} CA^\kappa B & \sum_{\kappa=k-k_{\mathbf{u}_2}+1}^{k-k_{\mathbf{u}_1}} CA^\kappa B & \dots & \sum_{\kappa=0}^{k-k_{\mathbf{u}_2}} CA^\kappa B \end{bmatrix}}_{\mathcal{O}^\dagger} \underbrace{\begin{bmatrix} \mathbf{x}_{k_N} \\ \mathbf{u}^0 \\ \mathbf{u}^1 \\ \vdots \\ \mathbf{u}^{n_{\mathbf{u}}-1} \end{bmatrix}}_{\mathbf{x}^\dagger} + \xi_k. \tag{5.36}$$

Admittedly, (5.36) is a daunting equation, which is best understood by looking at the columns of $\mathcal{O}^\dagger$. The first column is related to the standard observability matrix $\mathcal{O}$. The remaining columns of $\mathcal{O}^\dagger$ are obtained by summing columns of the dynamics matrix in (5.34) if the previous control is still active. This results in each remaining column corresponding to each input, where the summations are a result of the same input being active over a particular

time window. Note that when a new control is activated, the previous control must still be propagated forward in time.

Multiple Targets in Clutter

We use the matrices Φ_k and $\mathcal{B}(\mathcal{S}_{\mathbf{q}})$ as defined in (5.18) and (5.21), respectively, to estimate the initial states and piecewise-constant input vector using minimum subsets. Define $\bar{\mathcal{O}}^{\dagger} = \mathcal{B}(\mathcal{S}_{\mathbf{q}})\Phi_k\mathcal{O}^{\dagger}$. As before, the initial conditions of each RANSAC hypothesis can be found solving for the MLE, resulting in

$$\hat{\mathbf{x}}^{\dagger}{}_{k_N}(\mathbf{q}) = \left(\bar{\mathcal{O}}^{\dagger\,\top}\bar{\Xi}^{-1}\bar{\mathcal{O}}^{\dagger}\right)^{-1}\bar{\mathcal{O}}^{\dagger\,\top}\bar{\Xi}^{-1}\bar{Z}_k, \tag{5.37}$$

$$P^{\dagger}_{k_N}(\mathbf{q}) = \left(\bar{\mathcal{O}}^{\dagger\,\top}\bar{\Xi}^{-1}\bar{\mathcal{O}}^{\dagger}\right)^{-1}. \tag{5.38}$$

Note that the observability index of $\bar{\mathcal{O}}^{\dagger}$ is equal to $s^{\dagger} = n + n_{\mathbf{u}}$ when the time indices $\mathbf{q}$ of the minimum subset are chosen appropriately. In order for $\bar{\mathcal{O}}^{\dagger}$ to be observable, minimum subsets must have at least one measurement selected from each of the time regions when the $n_{\mathbf{u}}$ inputs are active.

Three changes to the standard RANSAC algorithm must be accounted for when estimating the input vector. First, the $n_{\mathbf{u}} - 1$ switching times in (5.35) must be randomly selected. Second, the minimum subsets must be selected to insure observability of $\bar{\mathcal{O}}^{\dagger}$ based on the selected switching times. The initial conditions are found according to (5.37) and (5.38). The consensus set of each hypothesis is found by propagating the initial conditions forward while adding in the appropriate input control based on the time index, as

$$\chi_k(\mathbf{q}) = \left\{\mathbf{z}_{\kappa} \in \mathcal{Z}_{\kappa} \;:\; \left\|\mathbf{z}_{\kappa} - (CA^{\kappa-1}\hat{\mathbf{x}}_{k_N}(\mathbf{q}) + B\hat{\mathbf{u}}_{\kappa})\right\|_R < \tau_{\mathrm{R}},\ \kappa = k_N \colon k\right\}, \tag{5.39}$$

where $\hat{\mathbf{u}}_{\kappa}$ is given by the estimated piecewise-constant control vector. The final modification to RANSAC is that the input vector must be accounted for when smoothing the trajectory over the consensus set. For simplicity, we refer to the input estimation technique developed in this section as RANSAC with IE.

While (5.36) greatly reduces the dimensionality of the problem, we still have the curse of dimensionality affecting the minimum subset size. The expected number of iterations needed to find a good minimum subset is discussed in the following section. In summary, the main problem is not only the increased cardinality of the minimum subset after augmenting the initial state vector with the control vector, but also the need to randomly select the switching times $k_{\mathbf{u}^1}, \ldots, k_{\mathbf{u}^{n_\mathbf{u}-1}}$. For this reason, the input vector is usually partitioned as one or two control modes.

Convergence Rate of RANSAC with IE

The cardinality of a RANSAC with IE minimum subset is given by $s^\dagger = s + n_\mathbf{u}$, where we again assume that the RANSAC minimum subsets are seeded with one good measurement value. The increased cardinality of the minimum subsets and the need to randomly choose the switching times dramatically increases the expected number of hypotheses needed to identify a good hypothesis. In this section, we derive the modified expected value of $\ell^\dagger$ by including additional terms in the original expected value given in (2.6).

As before, we must randomly select s measurements to estimate the initial conditions of the trajectory, where the probability of picking s good measurements is given by p^s. In addition, we randomly select another $n_\mathbf{u}$ measurements to estimate the input vector, where the probability of picking all good measurements for the inputs is $p^{n_\mathbf{u}}$. Finally, we must compute the probability of selecting the switching times, where we constrain $k_N \leq k_{\mathbf{u}^1} < \ldots < k_{\mathbf{u}^{n_\mathbf{u}-1}} \leq k$. The probability of correctly selecting the switching time $k_{\mathbf{u}^1}$ from the set of time steps within the measurement window is $\frac{1}{N}$. Thereafter, subsequent measurements are selected from the measurement window that are greater than $k_{\mathbf{u}^1}$; on average, when selecting the switching times uniformly over the measurement window, the sample set is reduced by half for each subsequent input. Taken together, the expected number of measurements needed to find a good minimum subset is

$$\mathrm{E}\big[\ell^\dagger\big] = \frac{1}{p^s\, p^{n_\mathbf{u}} \prod_{\varrho=1}^{n_\mathbf{u}-1} \left(\frac{\alpha_k}{\left(\frac{N}{2^{\varrho-1}}\right)} \right)}, \tag{5.40}$$

where $\alpha_k = 1, 2, \ldots$ is a tolerance on the required accuracy of the switching times, with smaller values denoting higher required accuracy.

We find that the expected number of iterations given by (5.40) to successfully identify the initial conditions, piece-wise control vector, and the switching times is quite large. Even if the tolerance α_k on the switching times is fairly large, it is expected that $\prod_{\varrho=1}^{n_{\mathbf{u}}-1} \left(\frac{\alpha_k}{\left(\frac{N}{2^{\varrho-1}}\right)} \right) \ll 1$. This is especially true when N is large, which is when the input estimation becomes more critical to achieve high percentages of correct inlier identification. Future work should explore how to more effectively select the switching times. One possible solution would be to develop a set of pre-defined switching times and to randomly select from among that set. In some scenarios, another possible simplification would be to assume that the control is zero at first, and then subsequent maneuvers persist for a specified time duration; in this way a strait-turn-straight maneuver could be modeled by identifying only the start of the turn, reducing the number unknown switching times by half.

5.4.2 Review of the Interacting Multiple Model Filter

In this section, we review the well-known IMM filter [21] that can be used to track a single maneuvering target in clutter. Thus far, we have only described recursive filtering by assuming that the process noise in (5.1) is zero-mean about the highest derivative; e.g. nearly-constant velocity and nearly-constant acceleration models [14]. However, there is a trade-off when designing a tracking filter for maneuvering targets. If the process noise is set too low, then track loss may occur while the target maneuvers. If the process noise is set too high, the gate size of the filter increases and is more sensitive to gross errors and it is more difficult to track through occlusions due to the increased uncertainty.

One class of algorithms that account for maneuvering targets are called multiple model filters, which model the maneuvers as a Markov jump linear system [127]. The motivation of these algorithms is to recognize that the dynamics of a system can be grouped into one of a finite number of dynamic modes. For example, a car can either drive straight or turn, accelerate or maintain a constant velocity. By characterizing the possible dynamic modes as a set of Markov jump linear systems, we can more accurately track the target as it switches

between the different dynamic modes. The premise of multiple model filters is to run multiple filters in parallel that are based on different model assumptions. The measurements are used to compute the mixing probabilities that estimate which modality best describes the current target maneuver.

We emphasize that multiple model filters are a framework that wraps around a Kalman filter or extended Kalman filter. Therefore, as with other tracking algorithms, a multiple model filter can be easily incorporated into the R-RANSAC framework to model the possible dynamic modes of targets currently being tracked, as is demonstrated in the following section. The trade-off for the increased accuracy is computational complexity, which scales quadratically with the number of models.

Following the notation in [127], the state propagation equation of (5.2) takes the form

$$\mathbf{x}_k = A_k^{(\iota)} \mathbf{x}_{k-1} + \mathbf{w}_k^{(\iota)}, \tag{5.41}$$

and the measurement matrix becomes

$$\mathbf{y}_k = \begin{cases} C_k^{(\iota)} \mathbf{x}_k + \mathbf{v}_k^{(\iota)} & \text{when } \omega = 1 \\ \emptyset & \text{when } \omega = 0, \end{cases} \tag{5.42}$$

where ι denotes the index to one of a finite set of $\bar{m}$ dynamical models $\{\mathbb{M}_1, \ldots, \mathbb{M}_{\bar{m}}\}$. The model $\mathbb{M}_\iota$ is characterized by the propagation and measurement matrices $A_k^{(\iota)}$ and $C_k^{(\iota)}$, and the process and measurement noise $\mathbf{w}^{(\iota)}$ and $\mathbf{v}^{(\iota)}$, with covariances $Q^{(\iota)}$ and $R^{(\iota)}$, respectively. The probability of switching between models is characterized by the transition probabilities

$$\mathcal{P}(\mathbb{M}_k^{(\iota)} \mid \mathbb{M}_k^{(\iota')}) = \pi_{\iota\iota'}, \forall k, \iota, \iota'. \tag{5.43}$$

A comparative study of several of the well-known algorithms that estimate the transitions between the various dynamic modes is found in [127]. In this section, we review only the IMM [21] as it has the fastest execution rate among the algorithms with acceptable error levels [127].

The IMM filter is basically composed of ι Bayesian filters running in parallel. Kalman filters are used when the systems are linear time-invariant. Prior to the propagation and update steps of the Kalman filter, there is a pre-mixing step that computes the initial conditions of the state estimates to each filter based on the mixed previous states and mixing probabilities of all the Kalman filters. After the Kalman propagation and update, the model probabilities are updated and the output for the time step is a weighted fusion of the updated states. Besides an estimate of the initial states $\mathbf{x}_0^{(\iota)}$ and covariance $P_0^{(\iota)}$, an initial distribution on the prior active model $\mu_0^{(\iota)}$ is required. The four steps of the IMM recursion are as follows:

1. Compute the mixed state estimate and state covariance matrices based on the previous mixing probabilities $\mu_{k-1}^{(\iota)}$, where

$$\begin{aligned}
\mu_{k-1}^{\iota'|\iota} &= \frac{\pi_{\iota'\iota}\,\mu_{k-1}^{(\iota')}}{\sum_{\varsigma=1}^{\bar{m}} \pi_{\varsigma\iota}\,\mu_{k-1}^{(\varsigma)}}, \\
\bar{\mathbf{x}}_{k-1}^{(\iota)} &= \sum_{\iota'=1}^{\bar{m}} \hat{\mathbf{x}}_{k-1}^{(\iota')}\,\mu_{k-1}^{\iota'|\iota}, \\
\bar{P}_{k-1}^{(\iota)} &= \sum_{\iota'=1}^{\bar{m}} \left[P_{k-1}^{(\iota')} + \left(\bar{\mathbf{x}}_{k-1}^{(\iota)} - \hat{\mathbf{x}}_{k-1}^{(\iota')}\right)\left(\bar{\mathbf{x}}_{k-1}^{(\iota)} - \hat{\mathbf{x}}_{k-1}^{(\iota')}\right)^{\top} \right] \mu_{k-1}^{\iota'|\iota}.
\end{aligned}$$

2. Propagate and update steps of the Kalman filter estimate for each mixed model.

3. Compute the likelihood that a measurement fits the model $\mathbb{M}^{(\iota)}$ as

$$L_k^{(\iota)} = \frac{1}{\sqrt{(2\pi)^m \left|S_k^{(\iota)}\right|}} \exp\left[-\frac{1}{2}\tilde{\mathbf{z}}_k^{(\iota)\top} S_k^{(\iota)}\,\tilde{\mathbf{z}}_k^{(\iota)}\right],$$

where $S_k^{(\iota)} = C_k^{(\iota)}\left(A_k^{(\iota)} P_{k-1}^{(\iota)} A_k^{(\iota)\top} + Q_k^{(\iota)}\right) C_k^{(\iota)\top} + R_k^{(\iota)}$ is the residual covariance, and $\tilde{\mathbf{z}}_k^{(\iota)} = \mathbf{z}_k - C_k^{(\iota)}\,A_k^{(\iota)}\,\mathbf{x}_{k-1}^{(\iota)}$ is the measurement residual. The updated model probabilities

are given by

$$\mu_k^{(\iota)} = \frac{L_k^{(\iota)} \sum_{\iota'=1}^{\bar{m}} \pi_{\iota'\iota} \mu_{k-1}^{(\iota')}}{\sum_{\iota'=1}^{\bar{m}} \sum_{\varsigma=1}^{\bar{m}} \pi_{\varsigma\iota'} \mu_{k-1}^{(\varsigma)}}.$$

4. Estimate fusion is the weighted sum of all filter states and covariances, given by

$$\hat{\mathbf{x}}_k = \sum_{\iota=1}^{\bar{m}} \hat{\mathbf{x}}_k^{(\iota)} \mu_k^{(\iota)},$$
$$P_k = \sum_{\iota=1}^{\bar{m}} \left[P_k^{(\iota)} + \left(\hat{\mathbf{x}}_k - \hat{\mathbf{x}}_k^{(\iota)} \right) \left(\hat{\mathbf{x}}_k - \hat{\mathbf{x}}_k^{(\iota)} \right)^\top \right] \mu_k^{(\iota)}.$$

5.4.3 IMM R-RANSAC

This section discusses the modular components that can be incorporated into the general R-RANSAC framework introduced in Algorithm 2 to track multiple maneuvering targets in clutter. There are only two modifications needed in Algorithm 2 in Chapter 2 in order to track maneuvering targets: first, the manner of RANSAC hypothesis generation in Line 7 and second, the utilization of the IMM filter that replaces Lines 2 and 9. Including the IMM filter is straightforward: ι Kalman filters are run in parallel with the transition probability matrix in (5.43) governing the weighting of the different IMM modes. There are two options for hypothesis generation. In both cases, as always, minimum subsets are selected from the set $\mathcal{S}_{\bar{\mathbf{q}}} \in \left\{ \mathbb{S}_{k_N:k-1}^{s-1} \times \{\mathbf{z}_k^i\} \right\}$, where $\bar{\mathbf{q}}$ is a uniformly distributed random index over the set of $\binom{\Psi_{k_N:k-1}}{s-1}$ minimum subsets.

The simplest and most practical hypothesis generation technique is to use the standard trajectory hypothesis estimation technique discussed in Section 5.2.1, where trajectories are estimated assuming that there is no maneuver within the measurement window. Obviously, this technique will be less effective in the presence of maneuvers and as the window size increases; however, since the minimum subsets are seeded with the current measurements, we will see in Section 6.3.1 that trajectories tend to end accurately—even if they do not track the true trajectory during the full measurement window. The second technique is

to utilize the RANSAC with IE technique discussed in Section 5.4.1. Using RANSAC with IE, accurate target trajectories can be obtained for maneuvering targets at the expense of computational complexity. Wrapping the RANSAC with IE technique into the R-RANSAC framework allows us to use the outliers to seed the RANSAC hypotheses. Therefore, the expected number of measurements needed to achieve a good minimum subset given in 5.40 becomes

$$\mathrm{E}\big[\ell^{\dagger}\big] = \frac{1}{p^{s-1}\, p^{n_{\mathbf{u}}} \prod_{\varrho=1}^{n_{\mathbf{u}}-1} \left(\frac{\alpha_k}{\left(\frac{N}{2^{\varrho-1}}\right)} \right)}. \tag{5.44}$$

While the expected number of iterations is reduced, the probability of randomly selecting the switching times still dominates the growth of (5.44).

5.5 Analysis

In Section 5.5.1, we show that under moderate assumptions R-RANSAC converges in mean to the true target states. Section 5.5.2 utilizes the convergence of R-RANSAC in mean to show that the inlier ratio converges as the measurement window N increases.

Throughout this section, we refer to the probabilistic origins of the inliers within the consensus set χ^j. At any time κ, the *inlier region* associated with the states $\hat{\mathbf{x}}_{\kappa}^{j}$ is given by a ball of radius τ_{R} as

$$\mathcal{B}_{\tau_{\mathrm{R}}}(C\hat{\mathbf{x}}_{\kappa}^{j}) = \Big\{ \xi \in \mathcal{R}_k : \big\| \xi - C\hat{\mathbf{x}}_{\kappa}^{j} \big\|_R < \tau_{\mathrm{R}} \Big\}, \tag{5.45}$$

where $\|\cdot\|_R$ is the Mahalanobis distance given by (2.11). Note that while the center of the ball $\mathcal{B}_{\tau_{\mathrm{R}}}(C\hat{\mathbf{x}}_{\kappa}^{j})$ is time-varying, the volume of the inlier region is time-invariant and equal for all state estimates, which we denote as $V_I = \mu\big(\mathcal{B}_{\tau_{\mathrm{R}}}(C\hat{\mathbf{x}}_{\kappa}^{j})\big),\ \forall j, k$.

The expected number of measurement inliers to the j^{th} estimated track of the i^{th} target are summed from the three measurement sources: the expected number of measurements $N_{I_{ij}}$ of the i^{th} target, the expected number of measurements $\sum_{i' \neq i} N_{I_{i'j}}$ of other (crossing) targets, and the expected number of clutter measurements $N_{\mathcal{K}}$ within the j^{th} inlier region. Under Assumption 2.3, clutter is uniformly distributed over $\mathcal{R}_k$, so the expected number of

clutter measurements is given by the volume of the inlier region divided by the volume of the entire surveillance region multiplied by the total number of expected clutter measurements in the measurement window,

$$N_{\mathcal{K}} = \mathrm{E}[|\mathcal{K}_{k_N:k}|]\frac{V_I}{V_k} = N\,\lambda_{\mathrm{c}}\,\frac{V_I}{V_k}, \tag{5.46}$$

where the volume $V_k = \mu(\mathcal{R}_k)$ of the surveillance region is known.

To compute the expected number of inliers to the j^{th} track, we first consider the probability that a measurement of the i^{th} target will be an inlier to the j^{th} track. Given the inlier threshold τ_{R}, and assuming a Gaussian measurement distribution as in (5.3), the probability of a measurement of the i^{th} signal being an inlier to the j^{th} track at time step k is given by

$$p_{\text{inlier}}(k,i,j) = \frac{1}{\sqrt{(2\pi)^m\,|R|}}\int\limits_{\xi\,\in\,\mathcal{B}_{\tau_{\mathrm{R}}}(C\hat{\mathbf{x}}_k^j)} \mathrm{e}^{-\frac{1}{2}(\xi - C\mathbf{x}_k^i)^\top R^{-1}(\xi - C\mathbf{x}_k^i)}\mathrm{d}\xi. \tag{5.47}$$

Essentially, (5.47) is equal to the gating probability p_{G} of a measurement being inside a ellipsoidal gate [19], the difference being that we use a fixed covariance R akin to the fixed gate of the traditional RANSAC algorithm. We also use $p_{\text{inlier}}(k,i,j)$ to account for the possible contributions of crossing signals. The number of expected inliers from all true targets is found by summing (5.47) over the measurement window

$$N_{I_{ij}} = \sum_{\kappa=k_N}^{k} \mathrm{E}\big[\big|\mathcal{Y}_\kappa^i \cap \mathcal{B}_{\tau_{\mathrm{R}}}(C\hat{\mathbf{x}}_\kappa^j)\big|\big] = p_{\mathrm{D}}\sum_{\kappa=k_N}^{k} p_{\text{inlier}}(\kappa,i,j). \tag{5.48}$$

The total number of expected inliers to the j^{th} model is the sum of (5.46) and (5.48),

$$\mathrm{E}\big[\big|\chi_k^j\big|\big] = N_{\mathcal{K}} + \sum_{i=1}^{M} N_{I_{ij}}. \tag{5.49}$$

Remark 1: While the integral in (5.47) does not have a closed-form solution in the general case, note that since $V_I > 0$, and $p_{\mathrm{D}} > 0$, we have that $p_{\text{inlier}}(k,i,j) > 0$ since multi-dimensional Gaussians have infinite support.

Remark 2: For the special case when $\mathbf{x}_k^i = \hat{\mathbf{x}}_k^j$, the resulting probability that a measurement of the true target is an inlier is

$$p^*_{\text{inlier}}\big(\tau_{\text{R}}\big) = p_{\text{inlier}}\big(k,i,j\big) = \frac{\gamma\big(\frac{m}{2}, \frac{{\tau_{\text{R}}}^2}{2}\big)}{\Gamma\left(\frac{m}{2}\right)}, \tag{5.50}$$

where τ_{R} is the standard deviation of the gate size in (5.45), $\Gamma(\cdot)$ is the Gamma function, $\gamma(\cdot,\cdot)$ is the lower incomplete Gamma function [59, Result 2]. In this case, there is no dependence on k, i, or j since the true and estimated states are equal; i.e., there is full overlap between the true and estimated inlier regions.

5.5.1 R-RANSAC Converges in Mean

To show convergence of the R-RANSAC algorithm, we first show that with probability one a subset of measurements will be selected using RANSAC that will result in an accurate estimate of the target states. We next show that the R-RANSAC track with the most inliers is the most accurate track. These results are then combined to show that R-RANSAC converges in mean to the true target states.

Recall that for every outlier measurement $\mathbf{z}_k^i$ at each time step k, the R-RANSAC algorithm uses RANSAC to randomly select ℓ minimum subsets from the set of windowed minimum subsets $\big\{\mathbb{S}^{s-1}_{k_N:k-1} \times \{\mathbf{z}_k^i\}\big\}$ to generate a hypothesis that best characterizes the outlier. If $p_{\text{D}} \ll 1$, then R-RANSAC may not correctly identify a true target the first time step it appears; however, the best hypothesis, or the one with the largest consensus set, is stored between time steps. Lemma 2 shows that over time, a good estimate of the target states will be found almost surely. To reflect this 'memory' aspect of R-RANSAC to store the best hypothesis between time steps, define the set ${}^{\ell}\bar{\mathbb{S}}^s_{1:k-1} \subset \mathbb{S}^s_{1:k-1}$ as the set of all minimum subsets randomly selected through the previous time step $k-1$. Note that these subsets are not actually stored as part of the R-RANSAC algorithm, but is just notation to reflect the set of sampled minimum subsets over time. At time k the set of randomly sampled minimum subsets grows recursively as

$${}^{\ell}\bar{\mathbb{S}}^s_{1:k} = {}^{\ell}\bar{\mathbb{S}}^s_{1:k-1} \bigcup \big\{\mathcal{S}^1_{\mathbf{q}}, \ldots, \mathcal{S}^{\ell}_{\mathbf{q}}\big\},\ \mathcal{S}^{\ell'}_{\mathbf{q}} \in \big\{\mathbb{S}^{s-1}_{k_N:k-1} \times \{\mathbf{z}_k^i\}\big\}, \forall i.$$

Lemma 2 *Let*

$$\mathcal{S}_k^* = \arg \min_{\mathcal{S}_{\mathbf{q}} \in {}^{\ell}\bar{\mathbb{S}}_{1:k}^{s}} \left\| g_k(\mathcal{S}_{\mathbf{q}}) - \mathbf{x}_k^i \right\|_R,$$

be a random variable describing the minimum subset resulting in the most accurate estimate of $\mathbf{x}_k^i$ *selected through time step* k*, where* $g_k()$ *is given by* (5.26)*. If the size of the measurement window satisfies*

$$N > \frac{s}{p_{\mathrm{D}}\, p^*_{\mathrm{inlier}}\big(\alpha(\epsilon)\big)}, \tag{5.51}$$

where s *is the cardinality of the minimum subset* $\mathcal{S}_{\mathbf{q}}$*,* p_{D} *is the probability of detection,* p^*_{inlier} *is given by* (5.50)*, and* $\alpha(\epsilon)$ *is the maximum gate size that ensures the minimum subset points are well-conditioned, then for every* $0 < \zeta < 1$ *and for any* $\epsilon > 0$*, there exists a time step* $K(\zeta)$*, such that for all* $k \geq K(\zeta)$

$$\Pr\big\{ \left\| g(\mathcal{S}_k^*) - \mathbf{x}_k^i \right\|_R < \epsilon \big\} \geq \zeta,\ i = 1, \ldots, M. \tag{5.52}$$

Since ζ *can be selected arbitrarily close to one, this implies that*

$$\lim_{k' \to \infty} \Pr\big\{ \left\| g(\mathcal{S}_k^*) - \mathbf{x}_k^i \right\|_R < \epsilon \big\} = 1,\ i = 1, \ldots, M. \tag{5.53}$$

Proof: Fischler and Bolles use $1-(1-p^s)^\ell$ in [53] to compute the probability ζ of successfully selecting a minimum subset with all good points after ℓ iterations, where p is defined in Lemma 1. In the R-RANSAC framework, where ℓ hypotheses are formed at each time step, we have

$$\bar{\zeta} = 1 - (1 - p^s)^{\ell\, k}. \tag{5.54}$$

Therefore, assuming that there are at least s expected good measurements, or in other words, assuming that the window size satisfies $N > s/p_D$, (5.54) can be solved for the number of time steps required so that the probability of selecting a good minimal subset is $\bar{\zeta}$, resulting in $k \geq K(\bar{\zeta}) = \frac{\log 1-\bar{\zeta}}{\ell \log (1-p^s)}$.

Unfortunately, it is known that (5.54) is optimistic [156] due to the conditioning of the measurements within the minimum subsets; i.e., s good measurements may still result

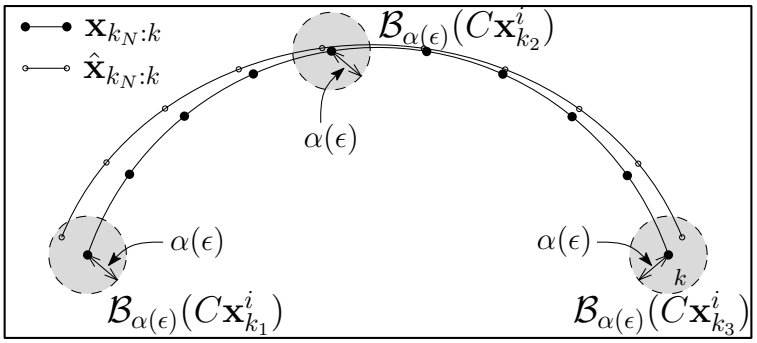

Figure 5.4: Example depicting the well-conditioned regions $\mathcal{B}_{\alpha(\epsilon)}(C\mathbf{x}^i_{k_{s'}})$ for a constant acceleration dynamic model, i.e. $s = 3$. The size $\alpha(\epsilon)$ of the regions is determined such that any minimum subset selected from these regions satisfies $\|g(\mathcal{S}_{\mathbf{q}}) - \mathbf{x}^i_k\|_R < \epsilon$.

in an inaccurate measurement due to measurement noise. Therefore, $\bar{\zeta}$ is near the upper bound of the convergence rate of RANSAC[1].

Therefore, we are interested in finding a lower bound $\underline{\zeta}$ such that we can guarantee that a minimum subset will be found almost surely. Due to the stochastic nature of the measurements, an exact lower bound is difficult to achieve. However, a conservative lower bound can be identified by constructing regions where, if a measurement were to fall within that region, the resulting minimum subset would be well-conditioned and produce an accurate estimate.

We construct s regions around the true signal states at the time steps $k_1, \ldots, k_s$, as

$$\mathcal{B}_{\alpha(\epsilon)}(C\mathbf{x}^i_{k_{s'}}) = \left\{\xi \in \mathcal{R}_k : \left\|\xi - C\mathbf{x}^i_{k_{s'}}\right\|_R < \alpha(\epsilon)\right\},\ s' = 1, \ldots, s, \tag{5.55}$$

where the desired accuracy ϵ determines the size $\alpha(\epsilon)$ of the regions around each each true state $\mathbf{x}^i_{k_{s'}},\ s' = 1, \ldots, s$, as shown schematically in Fig. 5.4.

Without loss of generality, we select s evenly spaced time steps over the measurement window, $k_1, \ldots, k_s$. The probability that a good measurement of the i^{th} target is within the region $\mathcal{B}_{\alpha(\epsilon)}(C\mathbf{x}^i_{k_{s'}})$ is given by (5.50). Therefore, the lower bound of the probability of

[1]The true upper limit would also incorporate the measurements of crossing signals and gross errors that could be used generate a good estimate of the i^{th} signal; however, their contribution is expected to be small.

successfully identifying an accurate minimum subset is

$$\underline{\zeta} = 1 - \Big(1 - \Big(p\, p^*_{\text{inlier}}\big(\alpha(\epsilon)\big)\Big)^s\Big)^{\ell k}. \tag{5.56}$$

The expected minimum number of time steps needed to achieve this minimum success rate is $K(\underline{\zeta}) = \frac{\log \underline{\zeta}}{\ell \log\left(1-(p f_\epsilon^m)^s\right)}$, therefore (5.52) holds as long as $N > \frac{s}{p_{\text{D}}\, p^*_{\text{inlier}}\left(\alpha(\epsilon)\right)}$ to ensure that $\mathrm{E}\Big[\sum_{s'=1}^{s} \Big|\mathcal{Y}^i_{k_{s'}} \bigcap \mathcal{B}_{\alpha(\epsilon)}(C\mathbf{x}^i_{k_{s'}})\Big|\Big] > s,\ \forall i.$

■

Remark 3: Since the process noise of the target dynamics in (5.1) is modeled as zero-mean Gaussian [14], targets are not maneuvering and their trajectories are well-modeled. When the process noise is no longer zero-mean, i.e. the target is performing an unmodeled maneuver, a maximum size of the measurement window must be enforced to maintain the accuracy of the initial state estimate.

In the following lemma, we show that when the j^{th} track has more inliers than all other tracks measuring the i^{th} target, then that implies that its state error is less than all other tracks. In order to prove this, we must consider the case when the targets are well separated, requiring the following final assumption.

Assumption 5.1: We assume that pairs of targets are not too similar to each other, i.e. $\left\|\mathbf{x}^i_k - \mathbf{x}^{i'}_k\right\|_\infty > \tau_{\mathbf{x}}, \forall k,\ \{i \neq i'\}$.

Assumption 5.1 is included to prevent both clustered targets, which would cause the tracks to merge, and crossing targets. This implies that

$$\sum_{i' \neq i} N_{I_{i'j}} = \sum_{i' \neq i} N_{I_{i'j'}}. \tag{5.57}$$

Lemma 3 *Let Assumption 5.1 hold and let* $\hat{\mathbf{x}}^j_k$ *and* $\hat{\mathbf{x}}^{j'}_k$ *be R-RANSAC estimates of the true states* $\mathbf{x}^i_k$ *at time* k*, then*

$$\mathrm{E}\big[\rho^j_k\big] > \mathrm{E}\Big[\rho^{j'}_k\Big] \quad \Leftrightarrow \quad \left\|\hat{\mathbf{x}}^j_k - \mathbf{x}^i_k\right\|_R < \left\|\hat{\mathbf{x}}^{j'}_k - \mathbf{x}^i_k\right\|_R, \tag{5.58}$$

where ρ^j_k *is given by* (2.10)*.*

Proof: It is shown in [59, Result 1] that the minimum region $\mathcal{B}$ needed to achieve a specified probability $P_0 = \int_{\mathcal{B}} f_X(\mathbf{x})\mathrm{d}\mathbf{x}$ is centered about the mean $\mathbf{x}$ of a Gaussian distribution. As a corollary of [59, Result 1], consider the inlier region in (5.45) where the probability $p_{\text{inlier}}(k,i,j)$ is maximized when $\left\|\hat{\mathbf{x}}_k^j - \mathbf{x}_k^i\right\|_R = 0$. Note that the derivative of the Gaussian distribution has a single minimum value, implying that $p_{\text{inlier}}(k,i,j)$ monotonically increases as the distance $\left\|\hat{\mathbf{x}}_k^j - \mathbf{x}_k^i\right\|_R$ decreases. Therefore, when comparing the two R-RANSAC track estimates $\hat{\mathbf{x}}_k^j$ and $\hat{\mathbf{x}}_k^{j'}$ of the true states $\mathbf{x}_k^i$, as $k \to \infty$, then if

$$\begin{aligned} \mathrm{E}\big[\rho_k^j\big] &> \mathrm{E}\Big[\rho_k^{j'}\Big] \\ \Leftrightarrow \frac{\mathrm{E}\big[\big|\chi_k^j\big|\big]}{N} &> \frac{\mathrm{E}\big[\big|\chi_k^j\big|\big]}{N} \\ \Leftrightarrow \sum_{i\in M} N_{I_{ij}} + N_{\mathcal{K}} &> \sum_{i\in M} N_{I_{ij'}} + N_{\mathcal{K}}. \end{aligned}$$

Subtracting the clutter terms and applying (5.57), we have

$$\begin{aligned} \Leftrightarrow N_{I_{i,j}} &> N_{I_{i,j'}} \\ \Leftrightarrow p_{\mathrm{D}} \sum_{\kappa=k_N}^{k} p_{\text{inlier}}(\kappa,i,j) &> p_{\mathrm{D}} \sum_{\kappa=k_N}^{k} p_{\text{inlier}}(\kappa,i,j') \\ \Leftrightarrow \sum_{\kappa=k_N}^{k} \mu\Big(\mathcal{B}_{\tau_{\mathrm{R}}}(C\hat{\mathbf{x}}_\kappa^j) \cap \mathcal{B}_{\tau_{\mathrm{R}}}(C\mathbf{x}_\kappa^i)\Big) &> \sum_{\kappa=k_N}^{k} \mu\Big(\mathcal{B}_{\tau_{\mathrm{R}}}(C\hat{\mathbf{x}}_\kappa^{j'}) \cap \mathcal{B}_{\tau_{\mathrm{R}}}(C\mathbf{x}_\kappa^i)\Big) \\ \Leftrightarrow \left\|\hat{\mathbf{x}}_k^j - \mathbf{x}_k^i\right\|_R &< \left\|\hat{\mathbf{x}}_k^{j'} - \mathbf{x}_k^i\right\|_R. \end{aligned}$$

■

Remark 4: When targets cross tracks, Assumption 5.1 will be violated during crossing. If the crossing time is of short duration, then we expect $N_{I_{i'j}} \approx N_{I_{i'j'}} \ll N_{I_{ij}}$, and the result will still hold. However, if the targets remain similar over a period of time, than the two track estimates may merge or switch labels [116].

The following theorem uses Lemmas 2 and 3 to prove that R-RANSAC converges in mean to the true state estimates. We also utilize the results in [145], where it is shown that the Kalman filter remains consistent in the presence of missed detections if the probability

of detection p_{D} is above a critical value. Unfortunately, their technique to estimate the p_{D} lower bound does not result in a closed-form solution. We therefore consider targets whose state dynamics in (5.2) are given by a nearly-constant velocity model, where

$$A = \begin{bmatrix} 1 & 0 & \delta_k & 0 \\ 0 & 1 & 0 & \delta_k \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \tag{5.59}$$

where the time step $\delta_k = 1$.

Theorem 2 (R-RANSAC Converges in Mean) *Given the system in* (5.59)*, if Assumptions 2.1-2.5, and 5.1 are satisfied, the window size satisfies* $N > \frac{s}{p_{\mathrm{D}}\, p^*_{\mathrm{inlier}}\left(\alpha(\epsilon)\right)}$*, the probability of detection* $p_{\mathrm{D}} > 6.7 \times 10^{-3}$*, and the number of stored tracks satisfies* $\mathcal{M} > M$*, then R-RANSAC will converge in mean to the true estimates, i.e.,* $\exists j$ *such that* $\lim_{k\to\infty} \mathrm{E}\left[\hat{\mathbf{x}}^j_k\right] = \mathbf{x}^i_k$*, for* $i = 1, \ldots, M$*.*

Proof: The proof follows by combining the results of Lemma 2 and Lemma 3. In the R-RANSAC algorithm every new observation is processed as either an inlier to an existing track or an outlier. If a track estimate $\left\|\hat{\mathbf{x}}^j_k - \mathbf{x}^i_k\right\|_\infty < \epsilon$ has not been found, then outliers to current tracks will trigger the creation of new tracks.

According to Lemma 2, by time $K(\zeta)$ a subset $\mathcal{S}^*_k \in {}^{\ell}\bar{\mathbb{S}}^s_{1:k}$ will have been randomly selected with probability ζ such that $\|g_k(\mathcal{S}^*_k) - \mathbf{x}^i_k\|_R < \epsilon$ for all signals $i = 1, \ldots, M$. Thereafter, inliers to the good tracks use a Kalman filter to update and refine the current estimates. With $p_{\mathrm{D}} = 1$ and with perfect data association, the Kalman filter is the minimum mean-squared estimator for LTI systems [23]. Sinopoli et al. show that when $p_{\mathrm{D}} < 1$ the Kalman filter still converges if the probability of detection is above a certain minimum detection rate [145]; for the system given by (5.59), the bounds on the minimum rate for a constant velocity model with $\delta_k = 1$ are $[0,\, 6.7 \times 10^{-3}]$. Therefore, good R-RANSAC tracks will converge in mean to the true estimates when $p_{\mathrm{D}} > 6.7 \times 10^{-3}$ for that scenario. Outliers to the tracks $\hat{\mathbf{x}}^j_k, j = 1, \ldots, \mathcal{M}$ will still form new tracks. Most of these tracks will have fewer inliers and be less accurate than the M previous state estimates, thereby making it impossible to

replace accurate tracks with a less accurate track. In the instances where a track is found with more inliers, then according to Lemma 3, it is more accurate and replaces the existing track. The number of tracks stored by R-RANSAC needs to satisfy $\mathcal{M} > M$ to avoid pruning accurate tracks. ■

Remark 5: We emphasize that R-RANSAC converges in mean for other systems in addition to the system given by (5.59). The only difference is that the exact lower bound on the probability of detection would need to be computed for each particular system according to [145].

5.5.2 R-RANSAC Inlier Ratio Convergence

Using the results of Theorem 2, we show that the inlier ratio converges as the measurement window size increases. This is useful when tuning the good track threshold parameter. If a short measurement window is used, then the following theorem suggests that a lower good track threshold should be used to increase the probability of track detection. Similarly, if a large measurement window is used, then the following theorem suggests that a higher good track threshold can be used to decrease false alarms without increasing the number of missed detections. Also, if the number of stored tracks is $\mathcal{M} \geq M$, then as N increases the likelihood of deleting the R-RANSAC hypothesis corresponding to the true track goes to zero if τ_ρ is set below the expected value of the inlier ratio.

Theorem 3 (Mean and Variance of the Inlier Ratio ρ) *If Theorem 2 holds, then the mean and variance of ρ_k^j are given by*

$$\mathrm{E}\big[\rho_k^j\big] = \lambda_{\mathrm{c}} \frac{V_I}{V_k} + p_{\mathrm{D}}\, p^*_{\mathrm{inlier}}\big(\tau_{\mathrm{R}}\big) + \frac{1}{N} \sum_{i' \neq i} \sum_{\kappa=k_N}^{k} p_{\mathrm{inlier}}(\kappa, i', j), \tag{5.60}$$

$$\begin{aligned} \mathrm{Var}\big[\rho_k^j\big] = {} & \frac{\lambda_{\mathrm{c}} \frac{V_I}{V_k} \Big(1 - \lambda_{\mathrm{c}} \frac{V_I}{V_k}\Big)}{N} + \frac{p_{\mathrm{D}}\, p^*_{\mathrm{inlier}}\big(\tau_{\mathrm{R}}\big)\big(1 - p_{\mathrm{D}}\, p^*_{\mathrm{inlier}}\big(\tau_{\mathrm{R}}\big)\big)}{N} \\ & + \sum_{i' \neq i} \frac{\Big(\frac{1}{N} \sum_{\kappa=k_N}^{k} p_{\mathrm{inlier}}(\kappa, i', j)\Big)\Big(1 - \frac{1}{N} \sum_{\kappa=k_N}^{k} p_{\mathrm{inlier}}(\kappa, i', j)\Big)}{N}. \end{aligned} \tag{5.61}$$

Proof: The inlier ratio of the j^{th} track is defined by (2.10). After taking the expected value and applying (5.49), we have

$$\mathrm{E}\big[\rho_k^j\big] = \frac{\mathrm{E}\big[\big|\chi_k^j\big|\big]}{N} = \frac{1}{N}\left(N_{\mathcal{K}} + \sum_{i=1}^{M} N_{I_{ij}}\right). \tag{5.62}$$

Substituting (5.46) and (5.48) into (5.62) results in

$$\mathrm{E}\big[\rho_k^j\big] = \lambda_{\mathrm{c}}\frac{V_I}{V_k} + \frac{1}{N}\sum_{\kappa=k_N}^{k} p_{\text{inlier}}(\kappa, i, j) + \frac{1}{N}\sum_{i'\neq i}\sum_{\kappa=k_N}^{k} p_{\text{inlier}}(k, i', j).$$

If Theorem 2 holds, then the j^{th} track will converge in mean to the true states of the i^{th} target. This implies that $\mathrm{E}\big[\big\|\hat{\mathbf{x}}_k^j - \mathbf{x}_k^i\big\|_R\big] = 0$ and, by (5.50), implies that the inlier regions are equal, so that

$$\mathrm{E}\big[\rho_k^j\big] = \lambda_{\mathrm{c}}\frac{V_I}{V_k} + p_{\mathrm{D}}\, p^*_{\text{inlier}}\big(\tau_{\mathrm{R}}\big) + \frac{1}{N}\sum_{i'\neq i}\sum_{\kappa=k_N}^{k} p_{\text{inlier}}(\kappa, i', j).$$

By Assumption 2.1, the measurements of all targets and clutter are independent. Therefore, the inlier ratio variance is given by summing the variance of all sources of the inliers affecting the measure ρ_k^j. The variance of the standard error for each term is calculated as $\mathrm{Var}[\varrho] = \frac{\varrho(1-\varrho)}{N}$, and the total inlier ratio variance is given by

$$\begin{aligned}
\mathrm{Var}\big[\rho_k^j\big] &= \mathrm{Var}\left[\lambda_{\mathrm{c}}\frac{V_I}{V_k}\right] + \mathrm{Var}\big[p_{\mathrm{D}}\, p^*_{\text{inlier}}\big(\tau_{\mathrm{R}}\big)\big] + \sum_{i'\neq i}\mathrm{Var}\left[\frac{1}{N}\sum_{\kappa=k_N}^{k} p_{\text{inlier}}(\kappa, i', j)\right] \\
&= \frac{\lambda_{\mathrm{c}}\frac{V_I}{V_k}\Big(1-\lambda_{\mathrm{c}}\frac{V_I}{V_k}\Big)}{N} + \frac{p_{\mathrm{D}}\, p^*_{\text{inlier}}\big(\tau_{\mathrm{R}}\big)\big(1-p_{\mathrm{D}}\, p^*_{\text{inlier}}\big(\tau_{\mathrm{R}}\big)\big)}{N} \\
&\quad + \sum_{i'\neq i}\frac{\Big(\frac{1}{N}\sum_{\kappa=k_N}^{k} p_{\text{inlier}}(\kappa, i', j)\Big)\Big(1-\frac{1}{N}\sum_{\kappa=k_N}^{k} p_{\text{inlier}}(\kappa, i', j)\Big)}{N}.
\end{aligned}$$

■

We emphasize that Theorem 3 applies to both static parameter estimation and dynamic state estimation. Figure 5.5 shows the effects of different window lengths when calculating ρ_k^j when there is a single signal in clutter. In this example, we consider the static

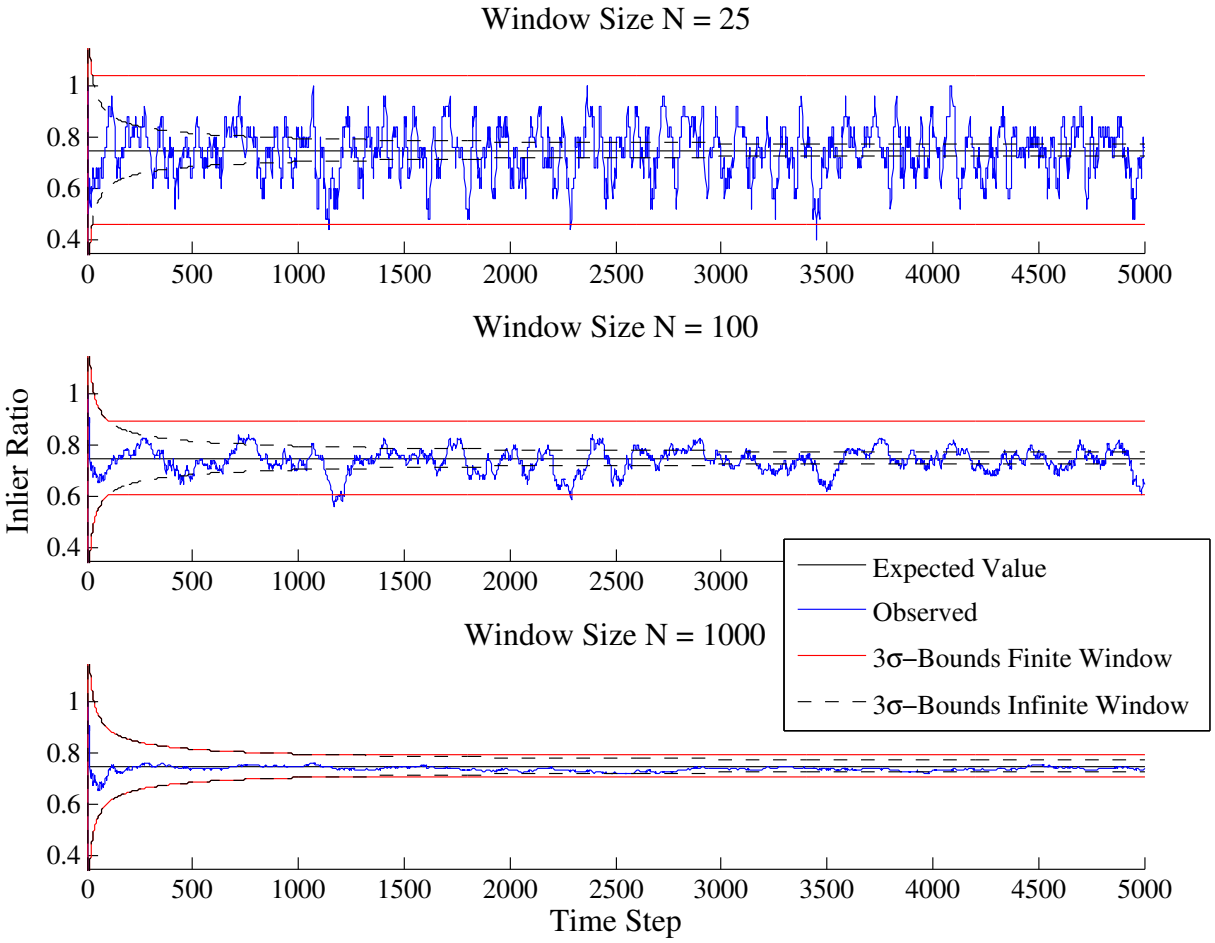

Figure 5.5: Effect of varying the window length N on the expected inlier ratio variance. In each of the subplots, the 3σ-bounds assuming an infinite window are shown for comparison.

parameter estimation discussed in Chapter 3, where the dimensions of the input and output are one, and the parameters $\mathbf{x} \in \mathbb{R}^2$ represent the slope and intercept of a line. The probability of detection is $p_{\mathrm{D}} = 0.75$, the average clutter rate is $\lambda_{\mathrm{c}} = 1$ returns per time step, the standard deviation of the measurement noise is $\sigma_R = 3$. We consider only a single example $M = 1$, implying that the third terms in (5.60) and (5.61) are zero. The R-RANSAC parameters estimating the true signal are $\mathcal{M} = 2$, $\tau_{\mathrm{R}} = 3\sigma_R$, $\ell = 10$, $\gamma = 0.6$, and $\tau_{\mathbf{x}} = [0.05,\ 10]^{\top}$. Note that as the measurement window length N increases, then the inlier ratio ρ_k^j converges to the expected value given by (5.60). The σ-bounds of the inlier ratio are computed as the square root of the variance given by (5.61).

Chapter 6. Applications to Dynamic Signal Tracking

This chapter summarizes our simulations and experiments testing the recursive-RANSAC (R-RANSAC) algorithm when tracking dynamic targets based on the theoretical development in Chapter 5. There are four main sections: first, we compare R-RANSAC with existing algorithms; second, we apply R-RANSAC to video-based tracking; third, we implement an interacting multiple model (IMM) R-RANSAC filter to track maneuvering targets; and fourth, we utilize R-RANSAC as a robust observer to mitigate the effects of occasional faulty measurements. We briefly summarize each section and highlight some of our results.

Section 6.1 compares the R-RANSAC algorithm to a number of existing trackers to demonstrate its capabilities. Specifically, we present our initial comparison of R-RANSAC and the Gaussian-mixture probability hypothesis density (GM-PHD) filter; we find that R-RANSAC has better track continuity while achieving approximately the same computational complexity and accuracy [116]. We also perform an extensive Monte Carlo simulation comparing R-RANSAC to the GNN, JPDA, MHT, GM-PHD, and MCMCDA filters using the multiple target tracking (MTT) metrics described in Section 2.5. In short, we find that R-RANSAC has a better than average execution rate (ER) and has very good accuracy in terms of RMS error. Due to the time required for the inlier ratio to exceed the good track threshold, the track probability of detection (TPD) of R-RANSAC is lower than most algorithms; however, we note that R-RANSAC performed comparatively better as the true probability of detection decreased. The one area where we feel that R-RANSAC excels is its ability to maintain track continuity, which is as good as the MHT filter in all of the scenarios we considered.

Section 6.2 presents some of the experimental results where the R-RANSAC filter is used to track vehicles using a stationary camera. While track continuity was signifi-

cantly improved over the GM-PHD, the nearly-constant velocity (NCV) implementation of R-RANSAC struggled to maintain track continuity of vehicles as they maneuvered around curves and at intersections. This led to the development of a modified RANSAC algorithm to estimate unknown inputs, which we call RANSAC with IE, and the IMM R-RANSAC algorithm.

Section 6.3 first discusses some of the simulation results of using RANSAC with IE to estimate a piecewise-constant input vector approximating the true input vector. We found that estimating multiple inputs using RANSAC comes at a high computational cost since the cardinality of the measurement subset is increased, which in turn lowers the probability of selecting a good measurement and necessitates increasing the number of generated RANSAC hypotheses to find a good trajectory estimate. One advantage of RANSAC with IE is its ability to identify maneuvers over large measurement windows, where a lower-order system may fail to accurately estimate the entire trajectory. We finish the section presenting some preliminary results of our implementation of the IMM R-RANSAC algorithm, which recursively estimates the modality of maneuvering targets.

Finally, in Section 6.4 we utilize the R-RANSAC algorithm as a robust observer to estimate the states of a feedback system, where the sensor may provide occasional faulty measurements. We simulate the altitude controller of an autonomous vehicle that is controlled to fly at a desired altitude above the terrain. Two types of sensor faults are considered: faulty measurements and a sudden (unknown) increase in sensor measurement noise variance. We show that R-RANSAC is robust to both types of sensor faults. A third simulation modifies the assumption on the clutter distribution by switching from a Poisson process to a Markov process to simulate the effects of a unmanned air system (UAS) flying along the edge of a cliff, where observations switch between measuring the top and the bottom of the cliff. In effect, this third scenario tests the ability of R-RANSAC to track through occlusions, which is possible provided the measurement window is large enough.

In each of the following examples and simulations, we utilize a NCV model [14]. The states are described by $\mathbf{x} = [\, x \; y \; \dot{x} \; \dot{y} \,]$, and the dynamics matrix A, process noise covariance

Q, measurement matrix C, and measurement noise covariance R are given by

$$
\begin{aligned}
A &= \begin{bmatrix} 1 & 0 & \delta_k & 0 \\ 0 & 1 & 0 & \delta_k \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \qquad Q = {\sigma_Q}^2 \begin{bmatrix} \frac{{\delta_k}^4}{4} & 0 & \frac{{\delta_k}^3}{2} & 0 \\ 0 & \frac{{\delta_k}^4}{4} & 0 & \frac{{\delta_k}^3}{2} \\ \frac{{\delta_k}^3}{2} & 0 & {\delta_k}^2 & 0 \\ 0 & \frac{{\delta_k}^3}{2} & 0 & {\delta_k}^2 \end{bmatrix}, \\
C &= \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}, \qquad R = {\sigma_R}^2 \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \\
B^\top &= \begin{bmatrix} 0 & 0 & \delta_k & 0 \\ 0 & 0 & 0 & \delta_k \end{bmatrix},
\end{aligned}
\tag{6.1}
$$

where the matrix B is used only when inputs are known or estimated. Current efforts are underway to analyze the trade-offs of utilizing higher-order models. Expected advantages of higher-order models are that the white-noise process noise assumption is more valid, allowing more accurate tracking of maneuvering targets. Expected disadvantages include possible over-modeling of the system and increased number of measurements needed when forming minimum subsets of RANSAC. To solve the latter disadvantage, a higher-order model could be used only within the recursive update of R-RANSAC while a lower-order model could be used to generate new tracks.

6.1 Multiple Target Tracking using R-RANSAC

6.1.1 R-RANSAC vs. PHD Comparison

During the last decade, the PHD filter has shown promise when applied to MTT scenarios. The theoretical and mathematical framework behind the PHD filter was introduced in [101]. Vo and Ma present a closed-form solution to the PHD recursion in [166], where they assume Gaussian dynamics and model the underlying random finite sets as Gaussian mixtures; the closed-form PHD is referred to as the GM-PHD filter. A survey of various PHD filter implementations is found in [103]. It has been shown in some scenarios that the

Table 6.1: R-RANSAC parameter settings for the R-RANSAC and GM-PHD comparison simulation.

Parameter	**Value**
RANSAC iterations ℓ	10
Good track threshold τ_ρ	0.75
Number of stored tracks $\mathcal{M}$	25
Similar track threshold $\tau_{\mathbf{x}}$	4

Parameter	**Value**
Inlier threshold τ_{R}	$3\sigma_R$
Measurement window N	25 time steps
Early termination threshold γ	τ_ρ
Minimum Track Lifetime τ_{T}	10 seconds

PHD filter outperforms classical algorithms such as the multiple hypothesis tracker and the joint probabilistic data association filter [105]. In this section, we compare the GM-PHD filter as derived in [166] to the R-RANSAC algorithm.

Twelve targets are initialized as summarized in Table 6.2. Ten of the targets persist throughout the 200 second simulation, while two targets are born after the simulation starts and die before it ends. The simulation step size is $\delta_k = \frac{1}{3}$ seconds. The state dynamics are defined by the 2D, constant-velocity model defined in (6.1); the true target dynamics follow a true constant velocity model, i.e., $\sigma_Q = 0$, while both the GM-PHD and R-RANSAC algorithms assume that the process noise is $\sigma_Q = 1$. The measurement noise is $\sigma_R = 10$. The targets are measured with probability of detection $p_{\mathrm{D}} = 0.95$ and an average of $\lambda_{\mathrm{c}} = 5$ clutter measurements are received per time step. The GM-PHD filter parameters are also the same as in [166], except that we do not assume a distribution on the birth parameters and instead use the measurements from the prior time step as the birth tracks.

Both algorithms are capable of real-time performance. The algorithms require almost the same amount of computation, with R-RANSAC requiring 0.0739 seconds per iteration and the GM-PHD filter requiring 0.0754 seconds per iteration in MATLAB on a 64-bit 3.0 GHz Core 2 Duo processor. However, the true computational complexity is dependent on the ratio of targets to clutter. As the number of average clutter returns increases, the PHD filter becomes notably faster than R-RANSAC since RANSAC must be used more frequently in the inner loop of the filter. Conversely, R-RANSAC becomes notably faster as the ratio of targets to clutter increases since R-RANSAC only performs a Kalman update. This trade-off is explored in Section 6.1.2.

Table 6.2: Summary of target lifetime and initial conditions.

Target Number	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**
Birth (seconds)	0	0	0	0	0	0	0	0	0	0	25	25
Death (seconds)	200	200	200	200	200	200	200	200	200	200	175	150
North Pos (m)	200	-200	200	600	600	-600	800	800	-900	-900	400	-800
East Pos (m)	0	-200	200	600	-600	-600	-800	800	-900	900	-400	-800
North Vel (m/s)	0	0	0	0	0	3	0	-5	0	0	0	9
East Vel (m/s)	2	0	-3	0	5	0	4	0	9	-9	1	0

In the example in this section, the ability of the GM-PHD filter and R-RANSAC to detect the targets is approximately equal, at 95.7% and 96.0% respectively. The mean root mean-squared error (RMSE) is comparable, with the R-RANSAC algorithm slightly more accurate with an RMSE of 5.6 meters, compared to 7.6 meters for the GM-PHD filter. Most importantly, the false track rate is significantly lower for R-RANSAC at 0.023 false tracks per time step compared to 0.062 for the GM-PHD filter. The number of targets over time estimated by both algorithms and the inlier ratio of the $\mathcal{M}$ stored R-RANSAC hypotheses are shown in Figure 6.2.

We note that track management can be incorporated by assigning a label to each good track which persists between time steps. The bottom plot of Fig. 6.2 illustrates what happens within R-RANSAC when targets 1 and 3 and targets 9 and 10 cross at times 45 and 100, respectively. In the first case, the targets are moving so slow that their tracks were merged due to multiple shared measurements; in the second case, track uniqueness persisted during the crossing targets.

6.1.2 Comparison of R-RANSAC vs. Existing MTT Algorithms

In this section we compare R-RANSAC with several tracking algorithms that are found in the literature. In doing so, we are able to validate many of the contributions we have claimed as to the utility of R-RANSAC under a variety of sensor processing conditions. Specifically, we show that R-RANSAC is computationally efficient, accurate, and produces track estimates that have exceptional track continuity. This comes at the expense of a lower probability of track detection, a higher then average false alarm rate, and longer track

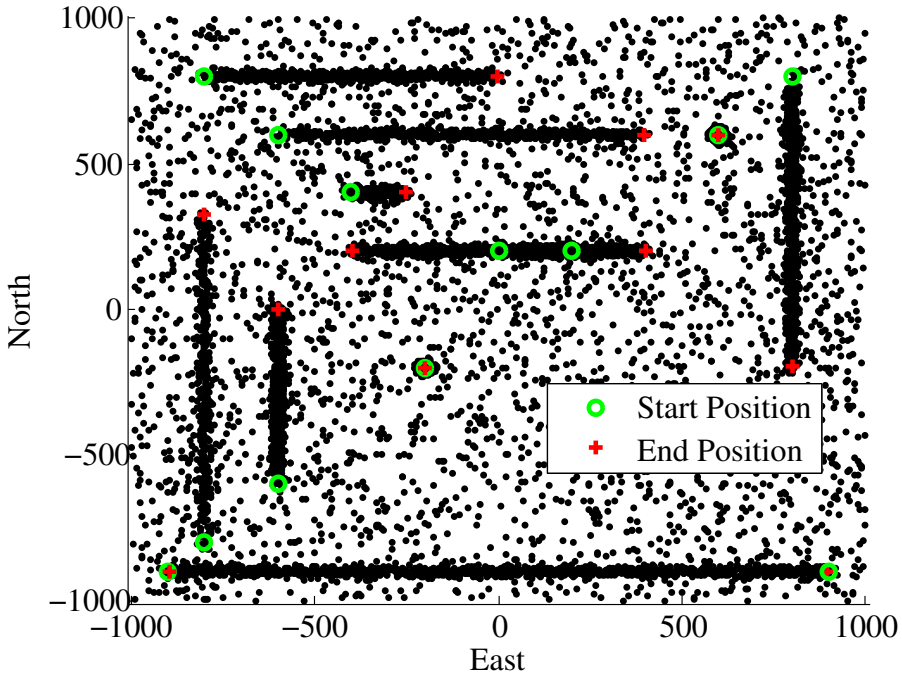

Figure 6.1: Measurement history of R-RANSAC vs. GM-PHD simulation.

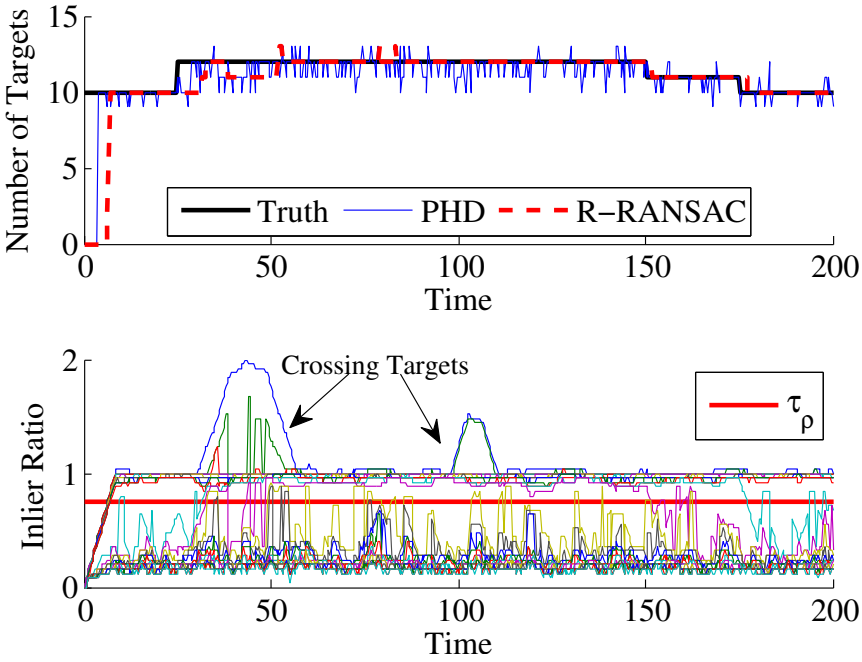

Figure 6.2: Simulation results showing the true and estimated number of targets and the inlier ratio of the R-RANSAC filter during the simulation.

fragment durations. All in all, the R-RANSAC algorithm demonstrates many good tracking qualities that make it well-suited to a number of different tracking problems—especially when ease of implementation is desired.

In the following section, we describe the Monte Carlo simulation designed to test the algorithms in a wide range of circumstances. We subsequently describe each of the algorithms used in the comparison and the specific implementation details required to run each algorithm. Finally, we present the results and our analysis of the tracking comparison.

Simulation Setup

Each simulation is 500 seconds long, during which time targets are randomly initialized in the 500×500 region in the center of a static surveillance region $\mathcal{R} = [0, 1000] \times [0, 1000]$. The volume of the static surveillance region is $V = 10^6$. Each target follows a constant velocity model, such as described in (6.1) where $\sigma_Q = 0$ and $\delta_k = 1$. In this simulation, the targets move up to a maximum constant velocity of $|\dot{\mathbf{x}}| = \frac{2}{3}$ m/s, which is set so that the targets do not usually leave the simulated surveillance region. The targets are born randomly during the course of the simulation, and exist for at least 125 seconds before dying. Targets are measured with measurement noise $\sigma_R = 10$ meters. The track assignment threshold $\tau_{\mathrm{D}} = 3\sigma_R$ meters, introduced in Section 2.5, defines the minimum required accuracy for an estimated track to be possibly assigned to the true target.

We use a Monte Carlo simulation to compare the algorithms while varying several parameters. The parameters that are varied between the various simulations are the probability of detection p_{D}, the clutter rate λ_{c}, and the total number of true targets $M_{\max}$ during the course of the simulation. The specific parameter settings considered in the Monte Carlo simulation are

$$\begin{aligned} p_{\mathrm{D}} &\in \big\{0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 0.99999\big\}, \\ \lambda_{\mathrm{c}} &\in \big\{0.00001, 1, \ldots, 5\big\}, \\ M_{\max} &\in \big\{1, 2, \ldots, 10\big\}, \end{aligned}$$

where the maximum probability of detection is not quite one, and the minimum clutter rate is not quite zero to avoid taking the logarithm of zero in some of the algorithms calculating log-likelihood. We run 25 Monte Carlo iterations at each parameter setting. For a fair comparison, each of the Monte Carlo iterations begins by resetting the random number generator so that each simulation or a particular random seed is the same across all parameter settings. This work focuses on analyzing random target trajectories, where targets may or may not cross. Future work should look in more detail at specific challenging scenarios, such as crossing, clustered, or even evasive targets.

Implementation Details of Tracking Algorithms

We briefly review some of the key aspects and parameter settings of the algorithms used in the MTT comparison. Common to all algorithms is knowledge of the underlying motion model, which is assumed to follow (6.1) with $\delta_k = 1$ and where the process noise is assumed to be $\sigma_Q = 1$. Also, the probability of detection p_{D} is provided to all algorithms as is the minimum track lifetime threshold $\tau_{\mathrm{T}} = 20$ time steps.

We note that we do not derive or specifically describe the algorithmic implementation of each algorithm, but instead will refer the interested reader to the literature from which each algorithm was implemented. We implement the GNN and JPDA filters in conjunction with an M/N track initiation algorithm, which are described in [19]. In our implementation of the MHT algorithm, we utilize the code written by Miller and provided by Cox et al. [37], which is based on their work in [39, 40, 97]. We also implemented the GM-PHD filter as described in [166]. Finally, the Cyber-Physical Systems Laboratory (CPSLAB) provided code for the MCMCDA filter [119] based on the description in [122]. All algorithms are implemented in MATLAB, with the exception of the MHT algorithm that is written in C.

Global Nearest Neighbor (GNN) Filter: The GNN filter is one of the simplest data association algorithms used for MTT. The GNN filter assumes that at most one measurement is generated per target per time step. The Munkres algorithm [112] is used to find the best association of measurement-to-track associations that minimizes the statistical distance between measurements and tracks. The maximum allowed Mahalanobis distance for measurement-to-track association is six. Advantages of the GNN filter are that it is compu-

tationally simple and easy to implement. Disadvantages are that it does not perform well in clutter and that it requires prior knowledge of the number of targets.

We use an M/N track initiator, as described in [19] to initialize tentative tracks from the measurements, identify true tracks, and merge and delete existing tracks. The M/N detector essentially identifies tracks with M detections out of N sequential measurements. First, the M/N detector uses all measurements that are *not* within the gate of existing tracks to form track initiators, which are tentative tracks initialized to the states of the measurement with zero velocity, but high covariance. Subsequent measurements are checked to see if they are inliers to existing tentative tracks. When a track has at least M_1 detections in N sequential measurements, then the tentative track is labeled as a true track. If a tentative track has fewer than M_2 detections in N sequential measurements, then the tentative track is deleted. Tentative tracks are merged if their Mahalanobis distance is less than the the merge threshold $\tau_{\mathbf{x}}$. Confirmed tracks are killed if more than $\overline{\mathsf{MD}}$ consecutive missed detections occur.

In our simulation, we purposefully define as many of the parameters in terms of the R-RANSAC algorithm parameters to develop a comparable experiment between different algorithms The minimum number of detections for track confirmation is $\mathrm{M}_1 = p_{\mathrm{G}}\tau_\rho N$, the minimum number of detections before tentative track deletion is $\mathrm{M}_2 = \frac{1}{2}p_{\mathrm{G}}\tau_\rho N$, the number of windowed measurement scans is $\mathrm{N} = N = 25$, the merge threshold is $\tau_{\mathbf{x}} = 6$, and the maximum allowed number of consecutive missed detections before killing a confirmed track is $\overline{\mathsf{MD}} = 5$.

Joint Probabilistic Data Association (JPDA) Filter: The JPDA filter originally derived in [56] extends the PDA filter [13] to track multiple objects in clutter. The PDA filter is well-suited to track a single target in clutter, but does not account for when closely spaced targets share measurements. An underlying assumption of both the PDA and JPDA algorithms is that a measurement can be assigned to at most one target track. In PDA, this assumption is incorrect when the validation regions of two or more targets overlap; the JPDA explicitly accounts for the possibility of a measurement being associated with more than one track when calculating the weighting probabilities for the measurement update step, thereby improving the tracking performance by reducing track coalescence at the expense of com-

putational complexity. The complexity increases because the set of measurements between overlapping targets must be determined, which is a difficult combinatorial problem as the density of the targets increases. Code for calculating the JPDA association probabilities was used from [124]. The parameters necessary to run the JPDA are the clutter density $\frac{\lambda_c}{V}$ and the probability of a good measurement being inside the measurement gate p_G as computed using the formulas given in [19, Table 6.1].

Similar to the GNN filter, one of the disadvantages of the JPDA algorithm is its requirement of knowing the number of targets present in the surveillance region. We again use the M/N track initiator to begin new tracks, identify good tracks, and merge and delete existing tracks. We use the same parameters that are used for the GNN M/N track initiator.

Multiple Hypothesis Tracking (MHT) Filter: The MHT filter used in this experiment is the one implemented by Cox et al. in [39, 40], and is based on the track-oriented MHT developed by Kurien et al. [96]. Essentially, a set of tracks is maintained between time steps and the K-best measurement assignments are used to update the track hypotheses, where the best assignments are found using Murty's algorithm [113]. One of the advantages of MHT algorithms over JPDA is that they do not need a separate track management algorithm, as it is handled within the MHT framework. This leads to the biggest disadvantage of the MHT algorithm, which is its computational complexity. The track-oriented MHT algorithm propagates a set of tracks from time step to time step. At the reception of new measurements, these tracks are then reformulated into a hypothesis tree of measurement-to-track associations, which are exhaustively enumerated. In each hypothesis group, each measurement is labeled as a true measurement of one of the existing signals, a false alarm, or a measurement of a new target. While theoretically optimal, the complexity of hypothesis-oriented MHT is exponential; the track-oriented MHT algorithm implemented by Cox et al. is polynomial, proportional to the number of measurements to the fourth power. Both algorithms require the suboptimal steps of pruning and merging to maintain a feasible number of propagated tracks.

We modified the C code of the MHT, found in [37], to compare against R-RANSAC. For an effective comparison with the algorithms implemented in MATLAB, we exported the data sets generated in MATLAB for use within MHT code. As before, the parameter

tuning was done to match the R-RANSAC parameters, but could possibly be tuned further for improved accuracy. The likelihood of track termination in the MHT code is set with parameter $\lambda_{\mathrm{X}} = 20$. The clutter density is set to $\frac{\lambda_{\mathrm{c}}}{V}$, and the probability of new targets is set to $\frac{M_{\max}}{V}$. The number of global hypotheses per group is 2000, the tree depth is $N = 25$, the minimum ratio between the likelihoods of worst and best global hypotheses is 0.0001, and the maximum Mahalanobis distance for a measurement to be within a track gate is 6. Also, new tracks are initialized with a state variance of 0.5. We note that the MHT code provides the option of using local features to assist in the data association process; however, as that is not used with any of the other algorithms, the code is modified to ignore that capability.

Gaussian Mixture Probability Hypothesis Density (GM-PHD) Filter: The GM-PHD filter was described in the previous section, but we do make a few additional notes here. Clark et al. showed that track continuity can be easily performed within the GM-PHD filter by passing a label from parent to children Gaussian modes [32]. In the case of merged tracks, two options exist: the track with the largest weight can be kept [32] or the track with the longest lifetime can be kept. We observe improved track continuity using the latter, so we use the weights only to distinguish between two tracks of equal lifetime.

Another implementation note is that the original GM-PHD tracking algorithm assumes a distribution describing the births of new objects within the surveillance region [166]. However, this information is unknown in general. Therefore, we use the measurements from the previous scan to seed the set of birth tracks. This removes the need for a prior distribution of expected birth tracks at the expense of computational complexity. Using notation discussed in [166], let $\mathcal{J}_k$ be the number of stored track modes, let $\mathcal{J}_{\gamma,k}$ be the number of birth tracks, and let $\bar{Z} = \mathrm{E}[|\mathcal{Z}_k|]$ be the expected number of measurements. As shown in [166], the computational complexity of the PHD filter is linear in the number of measurements $|\mathcal{Z}_k|$ and stored tracks $\mathcal{J}_k$. However, when $\mathcal{J}_{\gamma,k} = |\mathcal{Z}_{k-1}|$, the complexity of the PHD filter is proportional to

$$O\Big(\big(\mathcal{J}_k + \mathcal{J}_{\gamma,k}\big)\big(1 + |\mathcal{Z}_k|\big)\Big) \propto O\Big(\mathcal{J}_k\bar{Z} + \bar{Z}^2\Big), \tag{6.2}$$

or linear in the number of stored tracks and squared in the expected number of measurements per time step. Moreover, the complexity in (6.2) does not include the complexity of the

necessary merging step described in [166, Table II]. In the worst case, the complexity of the merging algorithm is proportional to the number of stored modes squared. Therefore, when the number of targets is unknown and the merging step is accounted for, the computational complexity of the GM-PHD is proportional to $\mathcal{O}\Big(\mathcal{J}_k^2 + \bar{Z}^2\Big)$.

The parameters of the GM-PHD filter are as follows. The probability of survival $p_{\mathrm{S}} = 0.999$, the clutter density is set to $\frac{\lambda_{\mathrm{c}}}{V}$, the merging threshold is equal to $\tau_{\mathbf{x}} = 6$, and the pruning threshold is equal to $10\mathrm{e}^{-5}$. The birth target weight is set to 0.1, and the birth target covariance is equal to $10^2 \left[\begin{smallmatrix} I_2 & \mathbf{0} \\ \mathbf{0} & \frac{I_2}{4} \end{smallmatrix}\right]$. The GM-PHD algorithm can also incorporate spawned targets, e.g. to facilitate detecting dismounts leaving a vehicle, but in this scenario this capability only adds computational complexity and is unnecessary and therefore not included in this simulation. Finally, the maximum number of tracks stored between time steps is set to $\mathcal{J}_k = 50$. We found that for 10-15 targets, decreasing the number of stored tracks began decreasing the accuracy of the tracker.

Markov Chain Monte Carlo Data Association (MCMCDA) Filter: We are grateful for the Cyber-Physical Systems Laboratory (CPSLAB) graciously providing code for the MCMCDA filter [119]. Unfortunately, since it is a MATLAB .dll file, it only runs using a 32-bit operating system, which does not run in current versions of MATLAB. We used a separate laptop with a 2008b version of MATLAB to run the MCMCDA algorithm and the R-RANSAC algorithm. This particular computer has a 32-bit 1.6 GHz Core Duo T2050 processor. To compare the ER of MCMCDA with the other algorithms run on different computers, we scale the MCMCDA ER by the ratio of the R-RANSAC ER on the new computer divided by the R-RANSAC ER on the old computer for each simulation. Undoubtedly, there are slight inaccuracies in this approach, but we expect them to be zero-mean about the true MCMCDA ER.

In essence, the MCMCDA algorithm uses Markov chain Monte Carlo (MCMC) sampling techniques to estimate the data association over a measurement window. The authors of [122] describe the convergence of the MCMCDA algorithm to the true solution as a simulated annealing-like algorithm. This is because for each set of measurements, one of several data association 'moves' are assigned, and the likelihood of that assignment is computed to score each of the samples. There are eight types of possible data association 'moves': switch-

ing between target birth and death, splitting and merging tracks, extending and reducing tracks with measurements, updating tracks, and switching tracks to account for whether targets cross or not. The specific parameters required for the MCMCDA algorithm are the clutter density $\frac{\lambda_c}{V}$, and the maximum and minimum target velocities of 2 and -2 m/s, respectively, to account for the uncertainties in measurement noise. The number of MCMC samples is set to 2000 and the window size of the association window is $N = 25$ time steps.

Unfortunately, for some reason our implementation of the MCMCDA algorithm does not work when the standard deviation of the measurement noise is greater than one. We emphasize that this is *not* a problem of the MCMCDA algorithm in general, but we presume it is instead a parameter setting internal to the MCMCDA algorithm that we cannot access since it is internal to the MATLAB .dll file. In order to compare with our other simulations, we scaled the entire simulation such that standard deviation of the measurement noise is equal to 1. Specifically, the target velocities, the standard deviation of the measurement noise, and each side of the surveillance region are all scaled by $\frac{1}{10}$. Plotting the resulting measurements of the normal and scaled simulations appeared exactly the same, just on different scales, but the scaled simulation resulted in accurate estimates of the target trajectories. The only modification to the metrics is that the resulting MCMCDA RMSE is scaled by 10 to compensate for the original scaling.

Recursive-RANSAC (R-RANSAC) Algorithm: Finally, we describe the parameters of the R-RANSAC algorithm. The method of data association used to compute the measurement weights is the same as the weights for the PDA algorithm [13]. We found that if the JPDA weights were used, then long-living, redundant tracks were formed caused by high variance good measurements; these tracks were long-living because the measurement weighting often split the measurements between the original and the redundant tracks, causing slower coalescence, hence a longer time before they were merged. We also examine the track continuity, such that tracks with an inlier ratio $\rho_k^j > \tau_\rho$ are assigned a track label $\mathcal{L}^j$ if no label has been assigned previously. The track merging logic is described as follows. If two tracks are to be merged and the one being retained has not previously been assigned a track label but the one being eliminated has, the retained track adopts the track label of the track being eliminated. If two tracks with labels are to be merged, then the track label of

the track with the longest lifetime is retained. In the rare case when both tracks have the same lifetime, the track with the highest inlier ratio is retained.

The number of stored tracks is $\mathcal{M} = 15$, the measurement window length is $N = 25$ time steps, the number of RANSAC iterations is $\ell = 25$, the inlier threshold of a fixed gate is $\tau_{\mathrm{RR}} = \tau_{\mathrm{R}} = 3\sigma_R$, and the early RANSAC termination threshold $\gamma = \tau_\rho$. Since the PDA weights are computed, then R-RANSAC also requires knowledge of the probability of detection p_{D}, the probability of a measurement being within the track gate p_{G}, and the clutter density $\frac{\lambda_{\mathrm{c}}}{V}$. The probability of survival is $p_{\mathrm{S}} = 0.999$, the merging threshold is $\tau_{\mathbf{x}} = 6$, and the good track threshold is set to the nominal value given in (2.17), where $\alpha_\rho = 2$.

Discussion of Comparison Results

In this section, we analyze the results of the Monte Carlo simulation described above for the listed algorithms. The metrics used to compare the algorithms are those described in Section 2.5. In summary, we find that the R-RANSAC algorithm is as good or better than all of the other algorithms in at least one way. R-RANSAC is faster than the MHT and the MCMCDA filters, has better track continuity than the GM-PHD filter, and is more accurate and has fewer false tracks than the JPDA filter. R-RANSAC does require a ramp-up time to detect new targets, which negatively affects the probability of track detection. However, we find that R-RANSAC is one of the three best algorithms for track continuity, and by far the simplest to implement and fastest ER. The following paragraphs describe the simulation results in regard to each metric.

Figure 6.3 presents the ER of all the algorithms except the MHT algorithm, which is implemented in C and was therefore not included because the comparison is not apples-to-apples. However, we should note that the MHT is considerably slower than the other algorithms. We observe that the efficiency of R-RANSAC improves relative to the other algorithms as the ratio of targets to clutter increases, and degrades as the ratio of targets to clutter decreases. Once a good track is established, R-RANSAC uses the weighted Kalman filter in Appendix A to update tracks and is very fast. Each clutter measurement generates a new hypothesis track, which is the most computationally complex step of R-RANSAC. If the number of clutter measurements is high, one possible method to reduce the ER is

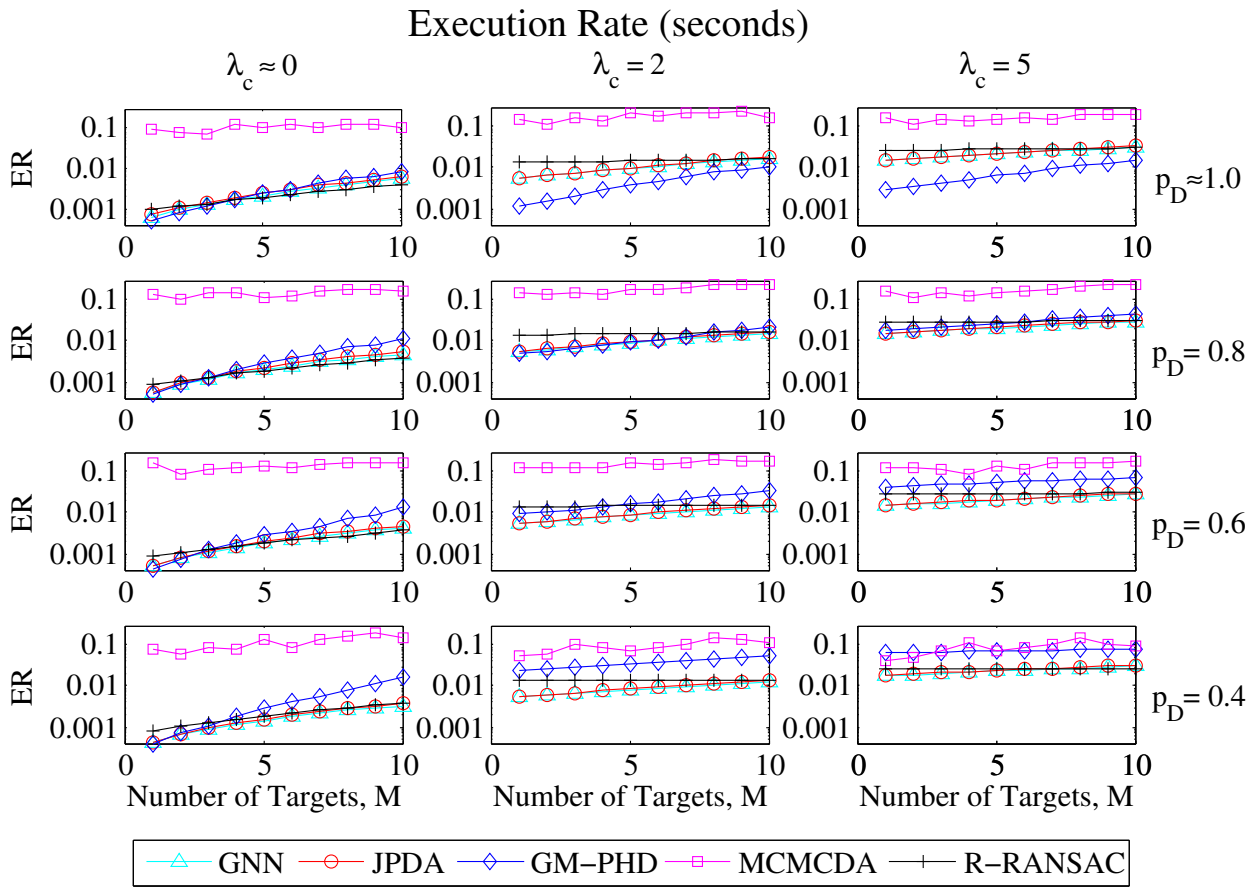

Figure 6.3: Multiple target tracking algorithm comparison: Execution rate. Note that the y-axis is scaled logarithmically to show the various plots.

to generate new tracks for only a subset of the outliers. This will improve the ER at the expense of delaying the detection of new tracks.

Another observation is that the GM-PHD filter suffers as the total number of measurements increase as seen in Fig 6.3 as λ_c increases from 0 to 5 and M increases from 1 to 10. This is due to the large number of stored tracks that must be tested when merging tracks. The slowest algorithm, not considering the MHT filter, is the MCMCDA filter. Decreasing the number of sampled particles or shortening the measurement window will decrease the ER at the expense of accuracy in some of the later metrics.

To quantify the relationship between the R-RANSAC algorithm and GM-PHD filter ER, we increase the number of stored tracks in R-RANSAC to $\mathcal{M} = 60$ and the maximum number of stored tracks in the GM-PHD filter to $\mathcal{J}_k = 200$ and tested when up to 50 targets and 50 clutter returns per time step were present in the data set. Both algorithms, in the

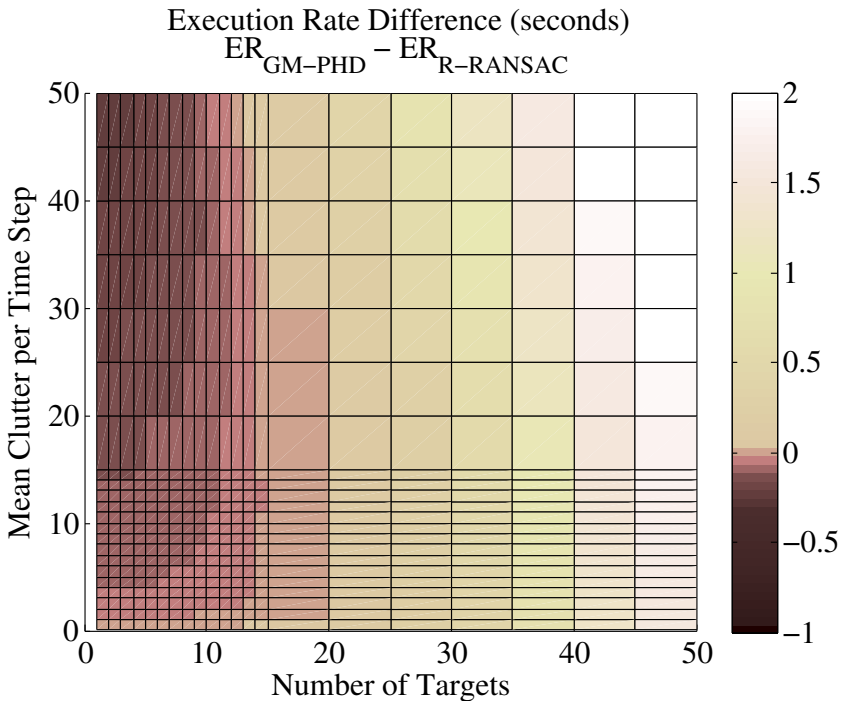

Figure 6.4: Expanded execution rate comparison between R-RANSAC and the GM-PHD filter.

worst case, have computational complexities that are squared in the number of measurements and in the number of stored models. We expect to see the ER depend on the ratio of the number of true target measurements to the number of clutter measurements. Figure 6.4 confirms that when the ratio of targets to clutter is high, R-RANSAC is faster than the GM-PHD filter, whereas when the ratio is low, the GM-PHD filter is faster than R-RANSAC.

In Fig 6.5, we see that the MHT filter has the lowest root mean-squared error (RMSE) in almost all scenarios. We find that the R-RANSAC algorithm and the GM-PHD filter are also both quite accurate. The reason that the JPDA algorithm is less accurate is that it performs soft assignments, which results in degraded accuracy in exchange for the desired robustness characteristics. The GNN algorithm may make incorrect assignments in clutter that degrades its accuracy. The RMSE of MCMCDA is reasonable, but could probably be improved by increasing the number of particles.

When considering the track probability of detection (TPD), as shown in Fig. 6.6, the MHT once again outperforms all other algorithms, with the MCMCDA having the next

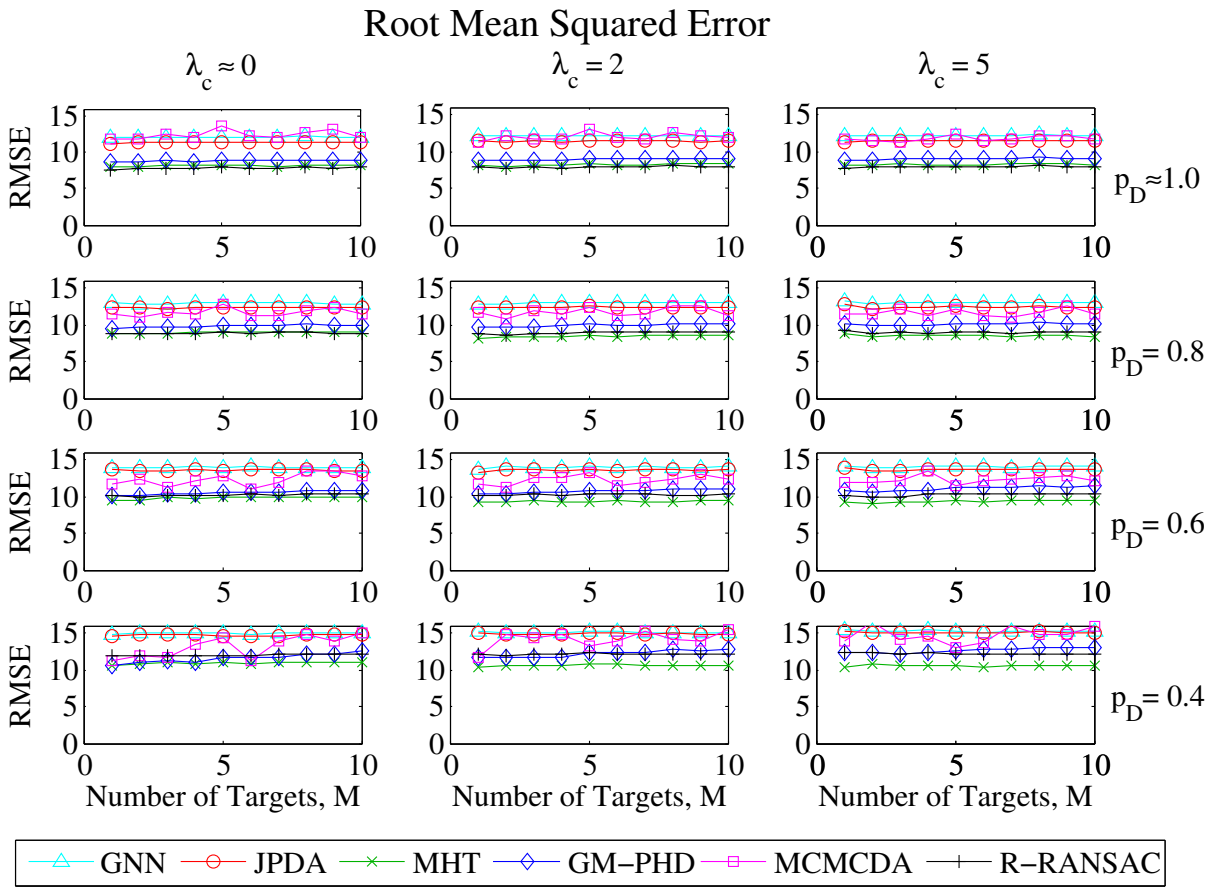

Figure 6.5: Multiple target tracking algorithm comparison: Root mean-squared error.

best detection capability. This is expected, since both algorithms rely on a sliding window of measurements to inform the target detection and tracking logic. We find that the R-RANSAC algorithm has comparatively lower TPD values in high p_{D} and low λ_{c} scenarios. This is due to the time it takes for the inlier ratio ramp-up to achieve the good track threshold. One method that may improve TPD of R-RANSAC is to determine the inlier ratio based on the time since the earliest measurement in the measurement window; however, it is expected that the false alarm rate will simultaneously increase. We recognize that as the probability of detection decreases, the performance of most of the other algorithms degrade, while the performance of R-RANSAC is only marginally reduced. This suggests that the initial ramp-up time to achieve the good track threshold is the main detracting factor in the lower TPD values.

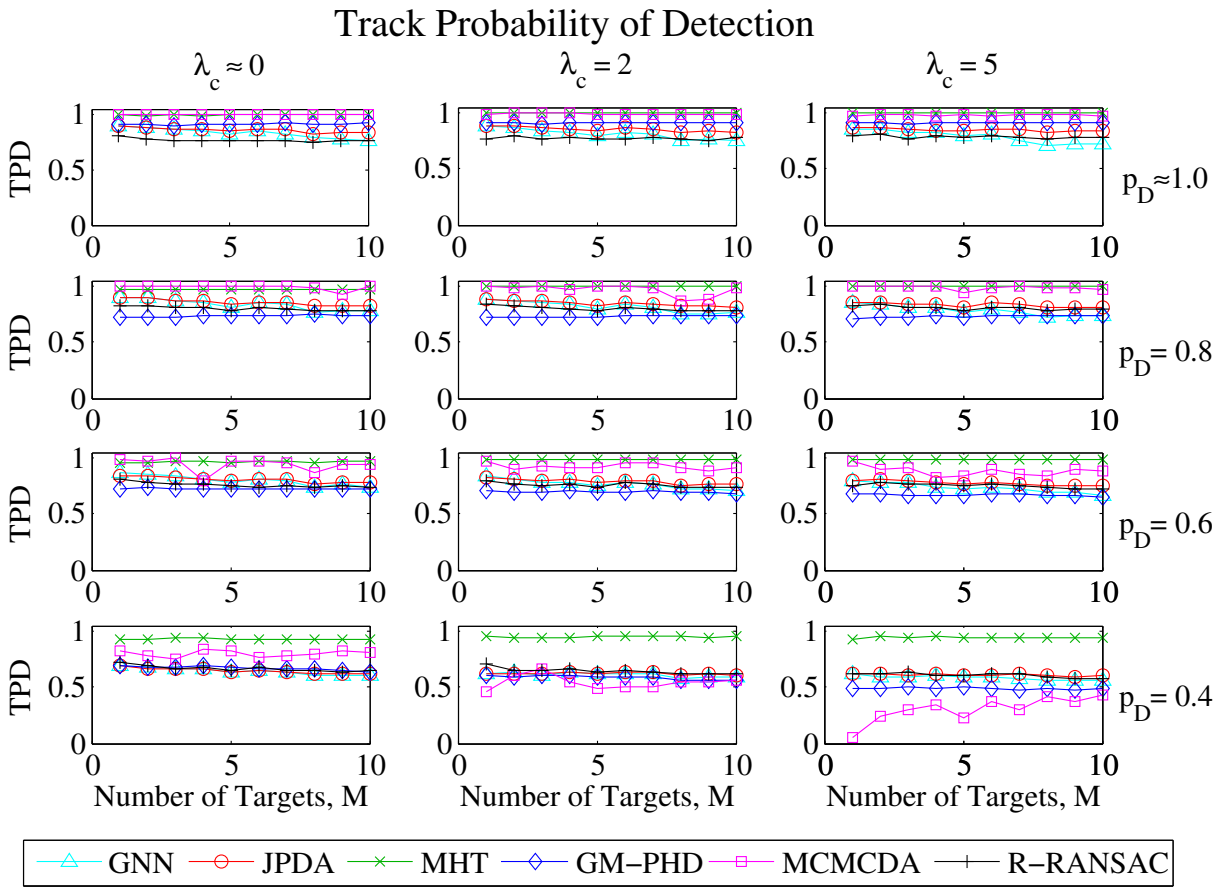

Figure 6.6: Multiple target tracking algorithm comparison: Probability of track detection.

The track false alarm rate (TFAR) for each algorithm is shown in Fig. 6.7. Here, we see that R-RANSAC performs about average compared to the other algorithms. While it seems odd that there can be false tracks with no clutter, this is explained by recalling that false tracks can either be generated by clutter or by high-variance measurements that are outside of the validation gate. We find that both the MHT the MCMCDA filters usually perform the best, that R-RANSAC produces a moderate amount of false tracks, and the GNN, JPDA, and GM-PHD filters produce a relatively large number of redundant tracks. We confirm the well-known phenomenon that the GM-PHD filter has a high variance of the estimated number of targets [105] when the missed detection is less than one and in clutter, and we see that the FAR appears to increase as the number of measurements increases.

One of the strengths of the R-RANSAC algorithm is its ability to robustly maintain track continuity over time, or achieve low path fragmentation rates (PFR), which is shown in

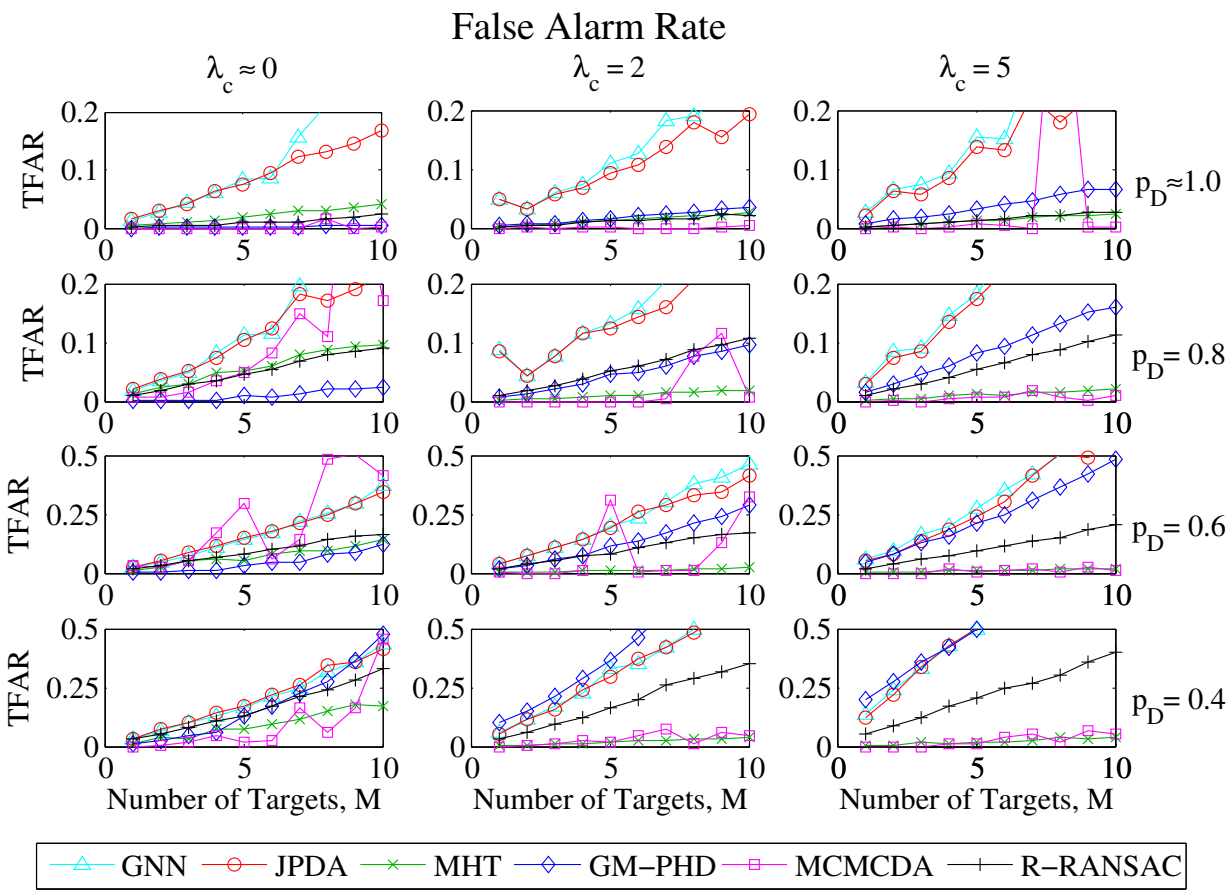

Figure 6.7: Multiple target tracking algorithm comparison: False alarm rate.

Fig. 6.8. We find that it is as accurate as the MHT filter, but at a much lower computational complexity. We also find that the MCMCDA filter performs almost as accurately as the MHT filter. The GM-PHD filter performs especially poorly due to the known fact that tracks are often lost after only a few missed detections [51]. These effects are expected to be somewhat mitigated through the C-PHD filter [104] at the expense of computational complexity. While the R-RANSAC and MHT algorithms seem stable as the probability of detection decreases, we see that the PFR of JPDA increases as the probability of detection decreases.

Figure 6.9 shows the track fragmentation rate (TFR). We find that that the R-RANSAC algorithm typically has a few instances where the inlier ratio drops below the good track threshold, and performs well compared to the other algorithms. Once again, GM-PHD filter typically results in a large number of track fragments due to its inability to operate in low probability of detection environments [51]. The MCMCDA filter also appears

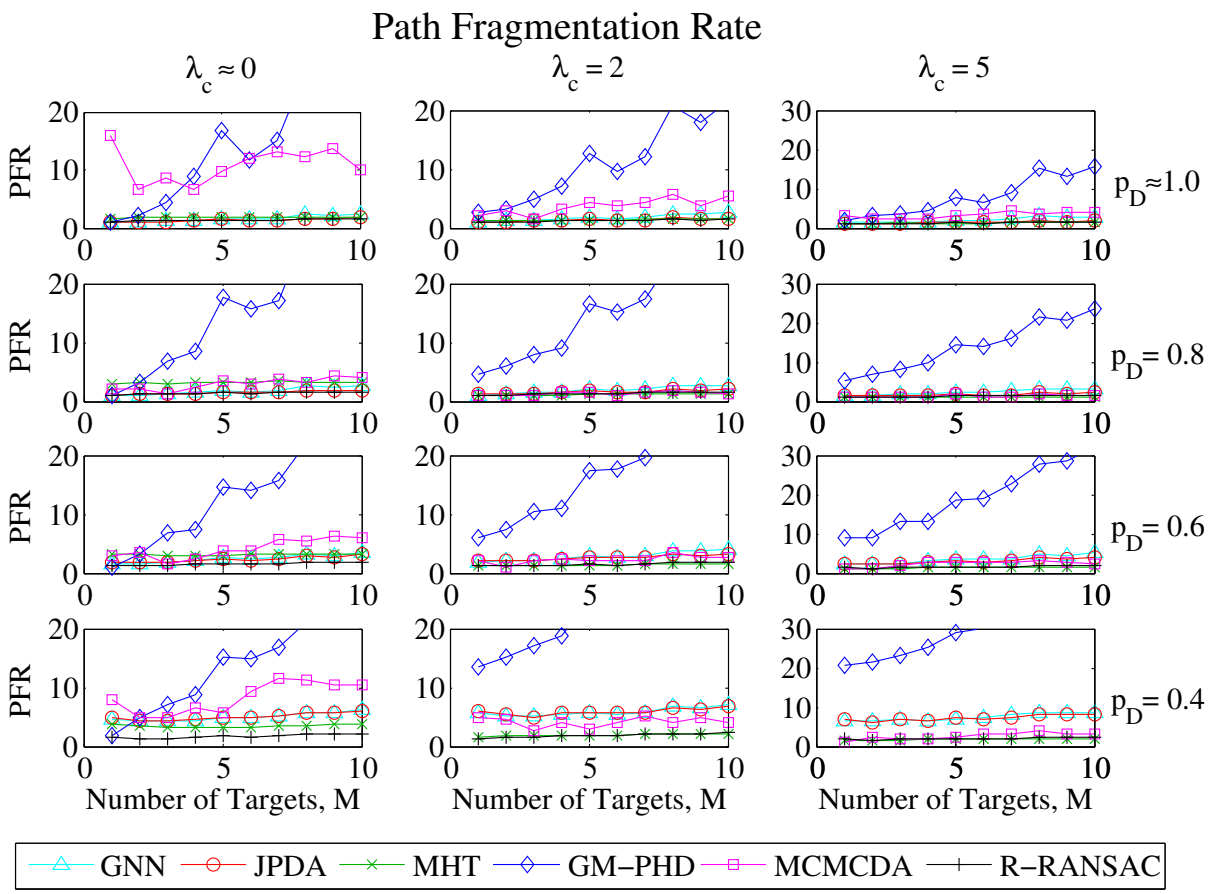

Figure 6.8: Multiple target tracking algorithm comparison: Path fragmentation rate.

to suffer similar problems, although it is not fully understood why; perhaps additional tuning of the algorithm may improve the results. The JPDA and MHT algorithms have TFR values of zero in all cases because the tracks either exist or terminate; i.e., there are never any times where confirmed tracks temporarily revert to a tentative track status.

Figure 6.10 is the final figure of our Monte Carlo analysis and shows the average track fragmentation duration (TFD). In this case, where track fragments occur, the GM-PHD and the MCMCDA filters have very short track fragment durations, since they are able to quickly reacquire the good track after missed detections. We observe that R-RANSAC is outperformed by the other algorithms due to its reliance on the inlier ratio, which may have high variance and cause the track to dip below the track threshold for an extended period of time. On the bright side, we note that R-RANSAC did not delete the track and was able to regain a lock on the target; if not, the TFD would be zero, as it is for the JPDA and

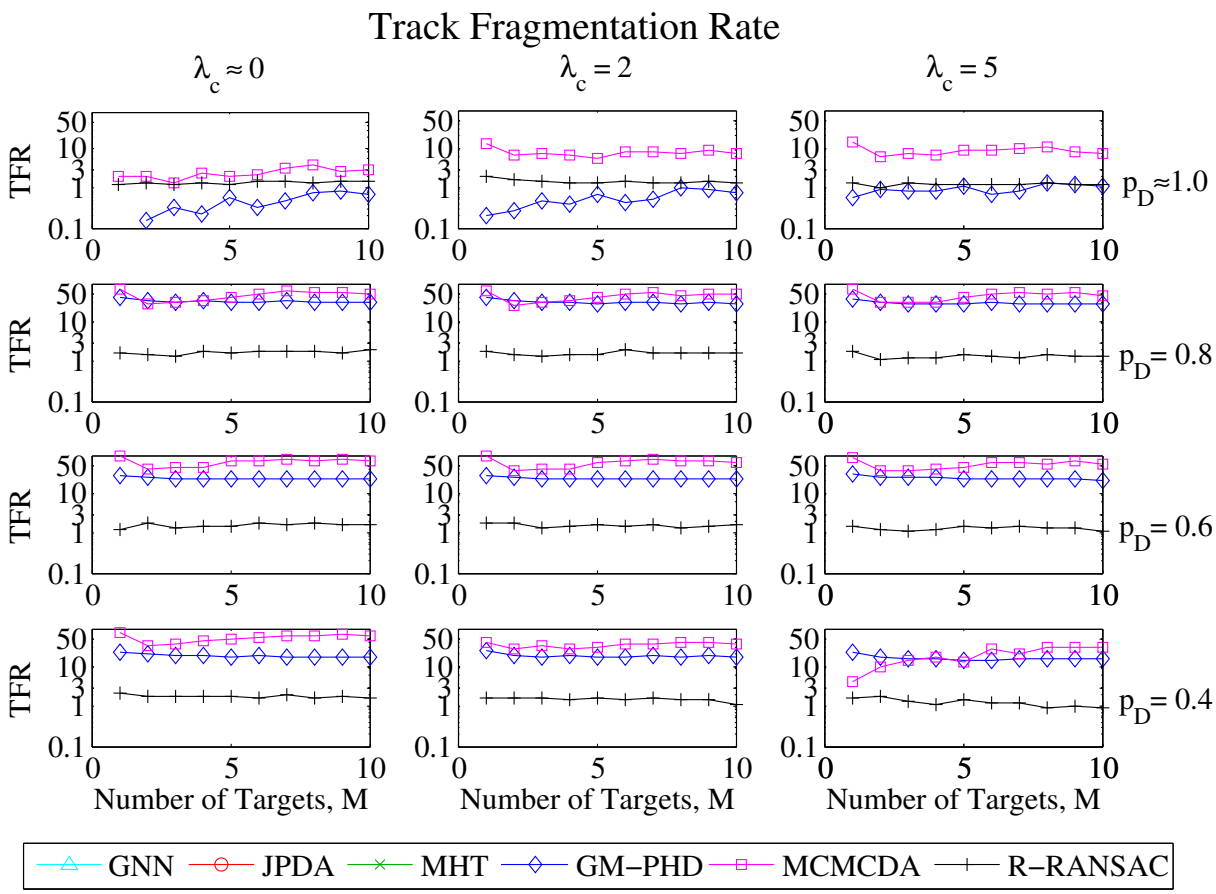

Figure 6.9: Multiple target tracking algorithm comparison: Track fragmentation rate. Note that the y-axis is scaled logarithmically to show the various plots. For this reason, the GNN, JPDA, and MHT filter results are not shown, which do not experience track fragmentation since the stored tracks are either active or deleted.

MHT filters that never have any track fragments. This is one area of possible improvement for R-RANSAC, where an easy fix would be to increase the measurement window N at the expense of computational complexity and the flexibility to track maneuvering targets.

6.2 Video-Based Multiple Target Tracking

In this section, we use R-RANSAC to track both vehicles and pedestrians from camera video data. We acknowledge and are grateful to Kyle Ingersoll, who implemented and wrote substantial portions of this section as we prepare to submit a paper describing this research. Due to the structured motion of the vehicles traveling along the road network, we included several modifications to the standard R-RANSAC to improve performance. The main issue

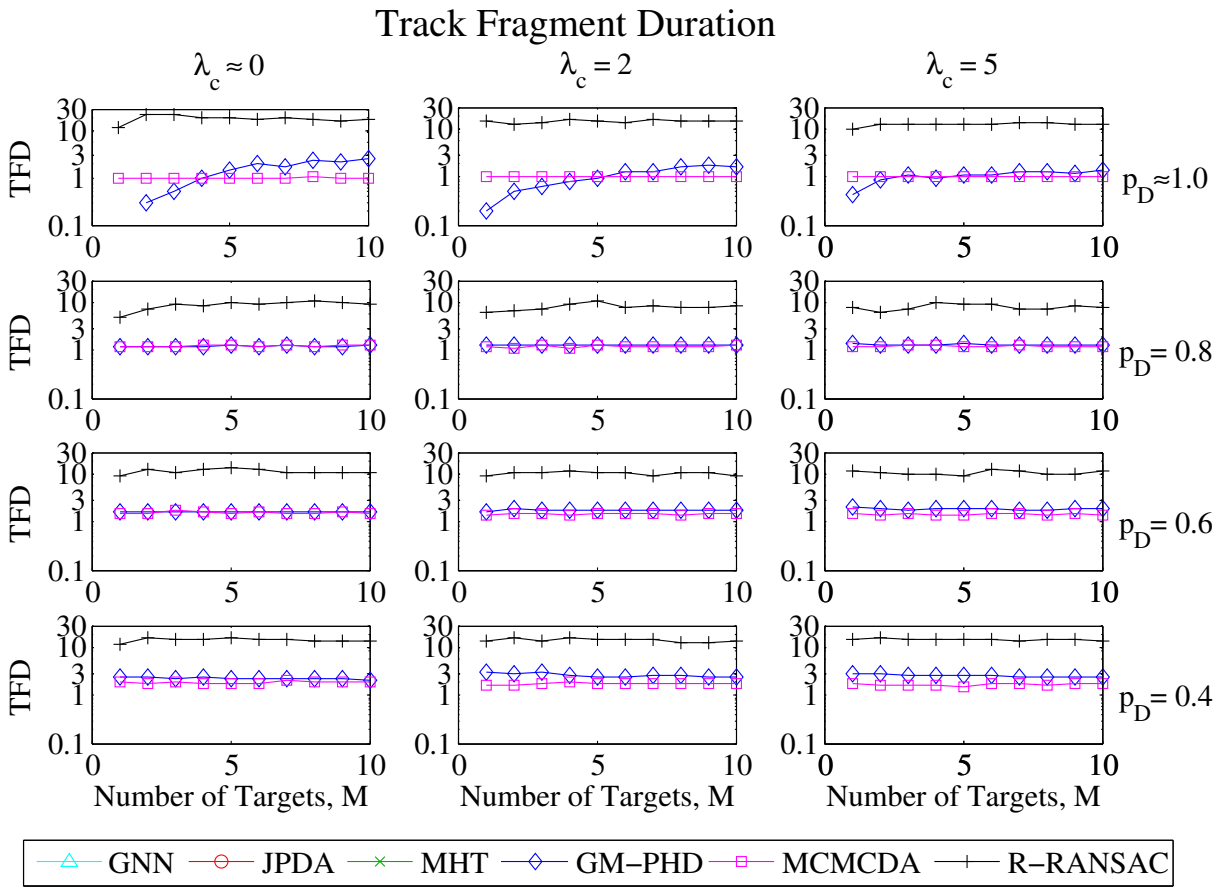

Figure 6.10: Multiple target tracking algorithm comparison: Track fragmentation duration. Note that the y-axis is scaled logarithmically to show the various plots. For this reason, the GNN, JPDA, and MHT filter results are not shown, which do not experience track fragmentation since the stored tracks are either active or deleted.

is the generation of high-velocity spurious tracks with high inlier ratios. When multiple cars are traveling in a row along a street, outliers use RANSAC to estimate a track and can possibly form a trajectory from the combined measurements of multiple vehicles. Due to $p_{\mathrm{D}} < 1$ for the true vehicles, it is possible that these spurious tracks can have an inlier ratio that rivals the inlier ratio of true target tracks. Therefore, if one can assume a range of velocities for the targets to be tracked, then a maximum velocity threshold can be set and all tracks that exceed that threshold can be discarded.

We use MATLAB's Computer Vision System Toolbox [1] to perform blob analysis. This tool uses a Gaussian mixture model to detect the foreground in an image. The procedure outlined in the MATLAB documentation is generally followed, but the code is modified to

output object centroids instead of bounding boxes. The minimum pixel area of detected objects is tuned according to the size of the objects being tracked.

Video Results

Our experiments produced two raw outputs:

1. *Video data.* In our video results, the outputs of the MATLAB blob detection algorithm are superimposed over the video as green plus signs. The position estimates of the good tracks produced by R-RANSAC are superimposed over the video as track label along with a red plus sign. Additionally, the number of active R-RANSAC tracks and the current frame number are displayed in the top left-hand corner. The entire results video of one run of R-RANSAC can be found at `http://www.youtube.com/watch?v=g96enlmsoDI`.

2. *Track information.* Important information associated with each track is stored, namely: the time line of that track, the state estimates at each time step of that track, and the inlier ratio at each time step of that track.

Since ground truth is not available, we manually identify a subset of the vehicles tracked during the video. We consider only a subset of the vehicles in the video, and study all the tracks associated with the first ten randomly selected tracks of vehicles[1]. The metrics were obtained from these ten vehicles. For each vehicle, the tracks associated with that object are determined, from which the PFR metric is calculated. From these tracks, we can compute the number of continuous time segments during which the object was tracked, i.e. the TFR metric.

The probability of track detection is calculated in the following way. The object is considered to be in the camera view when the entire object has entered the field of view; this is determined manually. In a similar manner, the object is considered to have left the camera view when the entire object is out of the field of view. Once the entering and leaving frame numbers are known, the track information produced by R-RANSAC provides the number of frames that the object is actively being tracked.

[1]We resample after discarding a bicycle track and two spurious tracks in order to identify ten vehicles.

Table 6.3: Video tracking metrics per track—10 objects from R-RANSAC.

Probability of track detection, (PTD)	0.927
Path fragmentation rate (PFR)	1.4
Track fragmentation rate (TFR)	3.9

Table 6.4: Video tracking metrics per frame—10 objects from R-RANSAC.

Percentage of objects tracked, p_{TD}^{k}	0.877
False detections, D_{false}	0.125

Again, due to our lack of ground truth, we also compute a few per-frame metrics, such as the false detection rate (D_{false}) and the percentage of vehicles tracked at frame k, (p_{TD}^{k}). To calculate these metrics, 10 random frames were selected. In each of the randomly selected frames, the number of trackable vehicles is determined. The D_{false} metric is computed by counting the number of R-RANSAC tracks unassociated with any vehicle and dividing by the total number of trackable vehicles. Trackable vehicles include both moving and temporarily stationary vehicles; parked cars are not trackable vehicles. Tables 6.3 and 6.4 show the numerical results extracted from the video results. Most importantly, we see that R-RANSAC has a high probability of track detection and low number of tracks per target. Each track has about four fragments, which are caused in part due to occlusions in the data and in part due to the variance of the detection rate. The per frame tracks show that RANSAC detects most targets with false alarms on average every eight frame.

The centroid data was also run through a GM-PHD filter [32, 166] in order to make comparisons to R-RANSAC. The video of the GM-PHD filter can be found at `http://www.youtube.com/watch?v=KPpcd_q_1W4`. The most significant result from the GM-PHD filter was the 7548 unique tracks produced whereas R-RANSAC produced only 852 tracks. This suggests that R-RANSAC has much better track continuity and/or fewer false alarms than the GM-PHD filter. The video itself contained about 12,700 frames and over 150 trackable objects including cars, pedestrians, bicyclists, and birds. In order to illustrate the impact

Table 6.5: Results from the first randomly selected track.

Tracking algorithm	**R-RANSAC**	**GM-PHD filter**
Probability of track detection, (PTD)	1	0.899
Path fragmentation rate (PFR)	3	44

Table 6.6: Results from the second randomly selected track.

Tracking algorithm	**R-RANSAC**	**GM-PHD filter**
Probability of track detection (PTD)	0.845	0.931
Path fragmentation rate (PFR)	4	26

of this finding, the first two randomly selected tracks (from which the metrics in Tables 6.3 and 6.4 are derived) are analyzed in greater detail.

The results of this analysis is shown in Tables 6.5 and 6.6. We find that R-RANSAC and the GM-PHD filter both have high PTD, but that the GM-PHD filter has much poorer track continuity. Figures 6.11, 6.12, 6.13, and 6.14 show the different tracks produced by R-RANSAC and the GM-PHD filter to capture these objects' trajectories. Each color corresponds to a different track. These tables and figures clearly demonstrate that R-RANSAC is superior to the GM-PHD filter in maintaining track continuity from time step to time step due to the lower PFR of R-RANSAC.

Table 6.7 shows the per-frame metrics of the GM-PHD filter; the same 10 frames were used in extracting these results as were used for R-RANSAC. At any instance in time, the GM-PHD filter more reliably tracks each trackable object and has fewer false detections than R-RANSAC. As discussed below, a large part of the reason for the per-frame success of the GM-PHD filter is because it is less reliant on the underlying model since it is a pure scan-to-scan tracker. While the R-RANSAC filter is also a scan-to-scan tracker, the new tracks are generated from the sliding window of measurements, and poor track initialization may result in multiple false tracks and missed detections.

Figure 6.11: The car tracked in this figure enters the field of view from the bottom, left-hand corner and leaves at top, center of the field of view. The R-RANSAC tracks corresponding to this car are displayed as different colors.

Figure 6.12: This figure shows the GM-PHD filter tracks corresponding to the car described in Fig. 6.11.

.

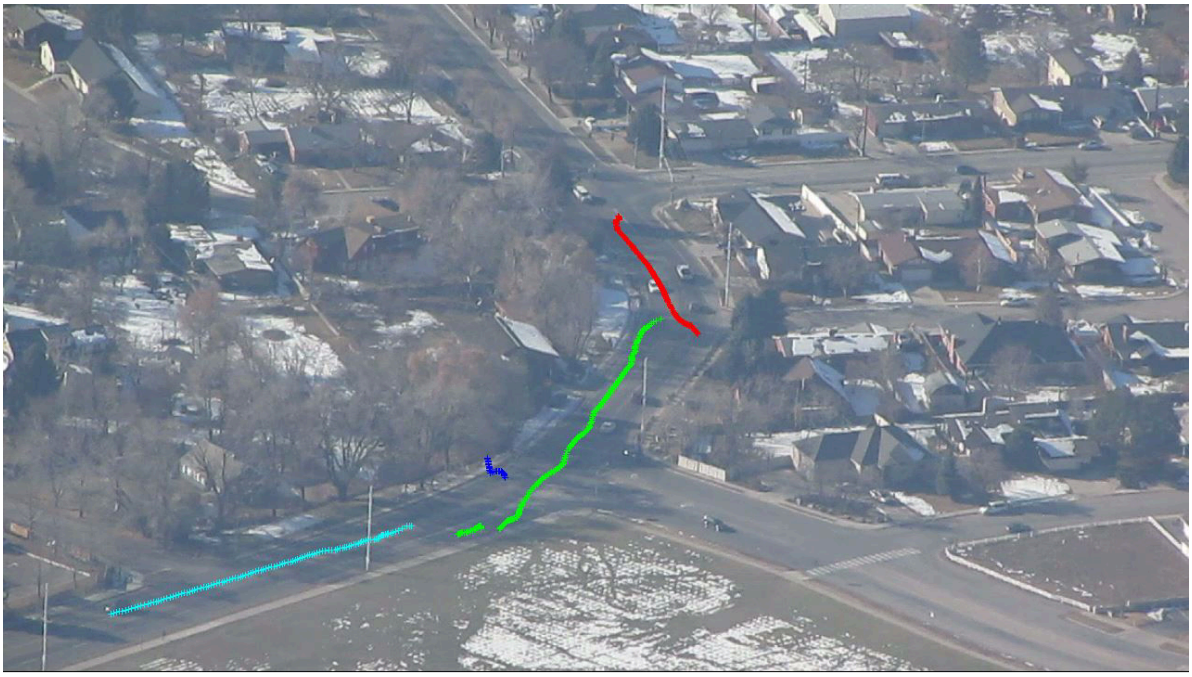

Figure 6.13: The car tracked in this figure was stopped at the stoplight of the upper, middle intersection when filming began. It then made a left-hand turn and exited the field of view at the bottom, left-hand corner. This figure best shows R-RANSAC's start-up period required for the track inlier ratio to achieve the good track threshold.

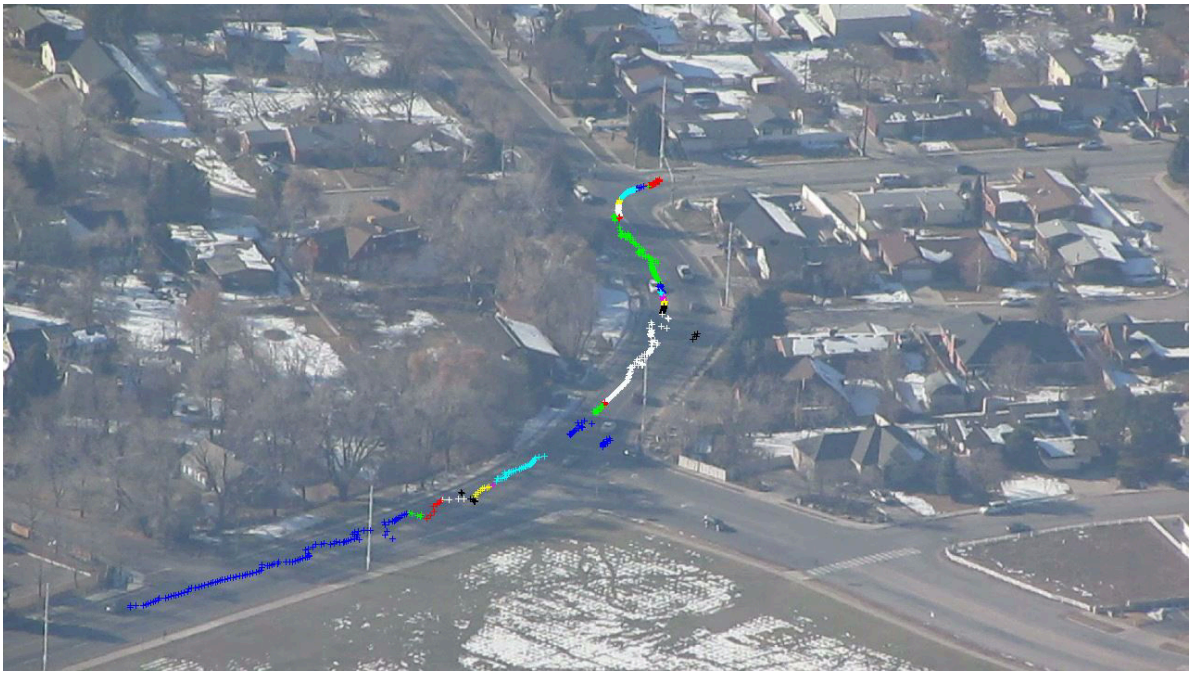

Figure 6.14: This figure shows the GM-PHD filter tracks corresponding to the car described in Fig. 6.13. This figure shows the GM-PHD filter's ability to begin tracking an object more quickly than R-RANSAC.

Table 6.7: Metrics per frame - 10 objects from GM-PHD.

Percentage of objects tracked, p_{TD}^{k}	0.964
False detections, D_{false}	0.036

The current version of R-RANSAC is highly successful at tracking an object that follows the NCV motion model. Figs. 6.11 and 6.13 both show that while the cars are traveling in straight trajectories, they are continuously tracked by the same track. Figs. 6.11 and 6.13 also show that R-RANSAC is even moderately robust when the cars are turning. However, it performs very poorly during the transitions between straight trajectories and turns. Frequently, there are separate R-RANSAC tracks corresponding to the pre-corner straight section, corner, and post-corner straight section. Two methods to possibly reduce false tracks generated by a poor dynamic model are to use a more complex dynamic model, or to incorporate an IMM filter [21] that estimates the different dynamic models of a vehicle.

The other problem evident in Figs. 6.11 and 6.13 is the inability of R-RANSAC or the GM-PHD filter to track objects after a prolonged stop. This is due to primitive object detection used in these simulations that only detects moving objects. Consequently, when objects come to a complete, prolonged stop, they are no longer detected. Thus, tracks of cars stopped by the traffic light in the upper, middle intersection are lost. This class of targets is known as a move-stop-move target [93], and future work should be conducted to account for these situations in video tracking.

6.3 Input Estimation

To better account for maneuvering targets, we implement RANSAC with input estimation (RANSAC with IE) as discussed in Section 5.4.1 and the IMM R-RANSAC algorithm as discussed in Section 5.4.3. The RANSAC with IE algorithm can account for maneuvers in the measurement window when generating a new hypothesis. RANSAC with IE estimates a piecewise-constant input vector while simultaneously estimating when those input changes take place. Sections 6.3.2 and 6.3.3 discuss the implementation of the IMM filter in both

simulation and in video. We would like to explicitly thank and give credit to Kyle Ingersoll who implemented the IMM R-RANSAC filter used in these two sections.

6.3.1 Input Estimation Simulations

In this section, we test the effectiveness of the RANSAC with IE algorithm to simultaneously estimate the initial states of a trajectory augmented by a piecewise-constant control vector using (5.37) to form new hypotheses. We consider an example of estimating the trajectory of a target from a batch of measurements when the target moves straight, then turns, then moves straight again. The specific parameters for the simulation are described as follows. The NCV model is used in (6.1), where the dimension of the measurement vector is $m = 2$, $\delta_k = 1$ and the number of time steps in the measurement window is $N = 100$. The initial states of the target are given by $\mathbf{x}_0 = \begin{bmatrix} 0 & 0 & 1 & 1 \end{bmatrix}^\top$ with $\mathbf{u}^0 = \begin{bmatrix} 0 & 0 \end{bmatrix}^\top$. At time $k_{\mathbf{u}^1} = 40$, the control vector changes to $\mathbf{u}^1 = \begin{bmatrix} -0.2 & -0.05 \end{bmatrix}^\top$, and $k_{\mathbf{u}^2} = 60$, the control vector changes again to $\mathbf{u}^2 = \begin{bmatrix} 0 & 0 \end{bmatrix}^\top$. Although there is no process noise, the RANSAC with IE algorithm assumes $\sigma_Q = 1$. The average number of clutter returns per time step is $\lambda_{\mathrm{c}} = 5$, and the probability of detection is $p_{\mathrm{D}} = 1$.

The true path and a hypothesis path generated by RANSAC with IE is shown in Fig. 6.15, where the number of RANSAC iterations $\ell^\dagger = 64800$ is given by (5.44), where $\alpha_k = 10$. We find that RANSAC with IE is very accurate if the number of RANSAC iterations is large enough. In Fig. 6.16, we examine the same simulation as used to generate Fig. 6.15, but over a wide range of ℓ. Figure 6.16 clearly demonstrates the disadvantage of the RANSAC with IE algorithm, in that a large number of RANSAC iterations are required to find a good fit. As discussed in Section 5.4.1, this is due to the fact that the expected number of iterations is a function of both the probability of selecting a good measurement p and also the length of the measurement window N.

A final thought on Fig 6.16 concerns the non-monotonic error functions of the trajectory, control inputs, and switching times. We see that the underlying cost function of RANSAC, i.e. the size of the consensus set $|\chi|$, is monotonic as expected. However, we show in Section 5.5.1 that a hypothesis with a larger consensus set implies greater state accuracy.

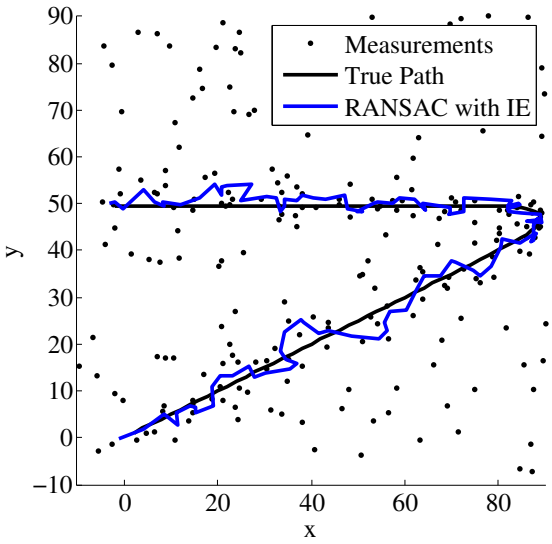

Figure 6.15: The measurements, true path, and estimated path using the RANSAC with IE technique, where the true path starts at the point (0, 0).

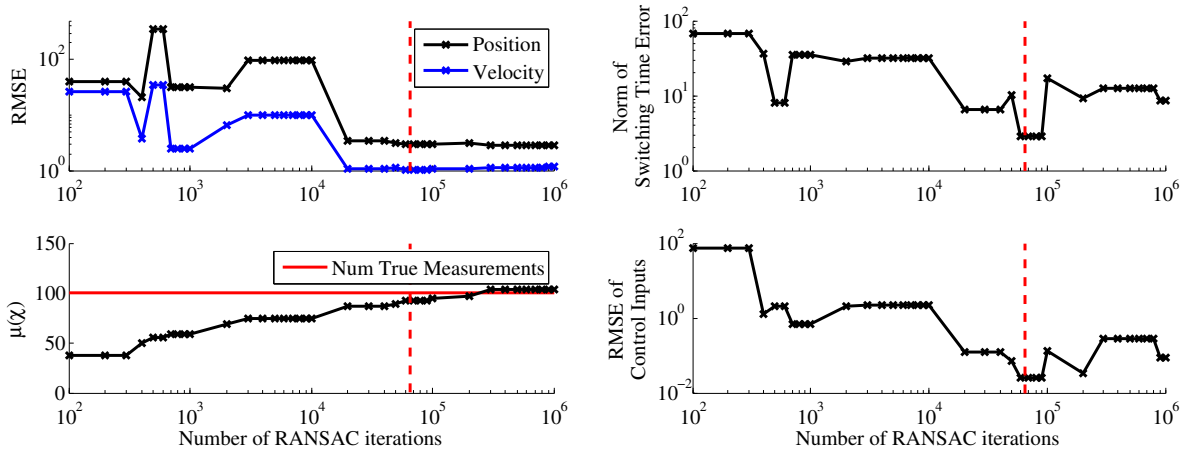

Figure 6.16: As the number of RANSAC iterations is varied, we see the convergence of several metrics quantifying the performance of RANSAC with IE. Namely, the RMSE error of both the position and velocity, the number of inliers in the best consensus set χ, the error of the switching times, and the RMSE error of the control inputs. The dashed vertical line is the nominal value for the number of RANSAC with IE iterations $\ell^\dagger$, as computed in 5.44.

We expect this result to carry over to the input estimation technique as well. The reason why we do not observe the monotonacity in the states, control inputs and switching times is that we are only considering one instance of a random variable, whereas Lemma 3 considers the expected value of these errors over time. We expect that an average of many simulations would result in monotonically decreasing plots for all metrics.

We also conduct another simulation to compare the RANSAC with IE estimate from the previous example to the original RANSAC-based maximum likelihood estimate from Section 5.2 based on a nearly constant velocity, acceleration (NCA), and jerk (NCJ) models, where

$$A_{\text{NCA}} = \begin{bmatrix} 1 & 0 & \delta_k & 0 & \frac{{\delta_k}^2}{2} & 0 \\ 0 & 1 & 0 & \delta_k & 0 & \frac{{\delta_k}^2}{2} \\ 0 & 0 & 1 & 0 & \delta_k & 0 \\ 0 & 0 & 0 & 1 & 0 & \delta_k \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}, \qquad A_{\text{NCJ}} = \begin{bmatrix} 1 & 0 & \delta_k & 0 & \frac{{\delta_k}^2}{2} & 0 & \frac{{\delta_k}^3}{6} & 0 \\ 0 & 1 & 0 & \delta_k & 0 & \frac{{\delta_k}^2}{2} & 0 & \frac{{\delta_k}^3}{6} \\ 0 & 0 & 1 & 0 & \delta_k & 0 & \frac{{\delta_k}^2}{2} & 0 \\ 0 & 0 & 0 & 1 & 0 & \delta_k & 0 & \frac{{\delta_k}^2}{2} \\ 0 & 0 & 0 & 0 & 1 & 0 & \delta_k & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & \delta_k \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix},$$

$$Q_{\text{NCA}} = {\sigma_Q}^2 \begin{bmatrix} \frac{{\delta_k}^5}{20} & 0 & \frac{{\delta_k}^4}{8} & 0 & \frac{{\delta_k}^3}{6} & 0 \\ 0 & \frac{{\delta_k}^5}{20} & 0 & \frac{{\delta_k}^4}{8} & 0 & \frac{{\delta_k}^3}{6} \\ \frac{{\delta_k}^4}{8} & 0 & \frac{{\delta_k}^4}{4} & 0 & \frac{{\delta_k}^3}{2} & 0 \\ 0 & \frac{{\delta_k}^4}{8} & 0 & \frac{{\delta_k}^4}{4} & 0 & \frac{{\delta_k}^3}{2} \\ \frac{{\delta_k}^3}{6} & 0 & \frac{{\delta_k}^3}{2} & 0 & {\delta_k}^2 & 0 \\ 0 & \frac{{\delta_k}^3}{6} & 0 & \frac{{\delta_k}^3}{2} & 0 & {\delta_k}^2 \end{bmatrix}, \qquad C_{\text{NCA}} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \end{bmatrix},$$

$$Q_{\text{NCJ}} = {\sigma_Q}^2 \begin{bmatrix} \frac{{\delta_k}^7}{252} & 0 & \frac{{\delta_k}^6}{72} & 0 & \frac{{\delta_k}^5}{30} & 0 & \frac{{\delta_k}^4}{24} & 0 \\ 0 & \frac{{\delta_k}^7}{252} & 0 & \frac{{\delta_k}^6}{72} & 0 & \frac{{\delta_k}^5}{30} & 0 & \frac{{\delta_k}^4}{24} \\ \frac{{\delta_k}^6}{72} & 0 & \frac{{\delta_k}^5}{20} & 0 & \frac{{\delta_k}^4}{8} & 0 & \frac{{\delta_k}^3}{6} & 0 \\ 0 & \frac{{\delta_k}^6}{72} & 0 & \frac{{\delta_k}^5}{20} & 0 & \frac{{\delta_k}^4}{8} & 0 & \frac{{\delta_k}^3}{6} \\ \frac{{\delta_k}^5}{30} & 0 & \frac{{\delta_k}^4}{8} & 0 & \frac{{\delta_k}^4}{4} & 0 & \frac{{\delta_k}^3}{2} & 0 \\ 0 & \frac{{\delta_k}^5}{30} & 0 & \frac{{\delta_k}^4}{8} & 0 & \frac{{\delta_k}^4}{4} & 0 & \frac{{\delta_k}^3}{2} \\ \frac{{\delta_k}^4}{24} & 0 & \frac{{\delta_k}^3}{6} & 0 & \frac{{\delta_k}^3}{2} & 0 & {\delta_k}^2 & 0 \\ 0 & \frac{{\delta_k}^4}{24} & 0 & \frac{{\delta_k}^3}{6} & 0 & \frac{{\delta_k}^3}{2} & 0 & {\delta_k}^2 \end{bmatrix}, \; C_{\text{NCJ}} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix},$$

where the process noise for the constant velocity, acceleration, and jerk models is $\sigma_Q = 3$, but remains $\sigma_Q = 1$ for the RANSAC with IE. Figures 6.17 and 6.18 compare the resulting estimated trajectories and compare the Euclidean norm of the position errors of each of the techniques. The RMSE values and the final number of inliers are given in Table 6.8. In Fig. 6.17, the number of RANSAC iterations ℓ was specified to be $\mathrm{E}[\ell]$, where $\mathrm{E}[\ell]$ is calculated for the respective models using either (2.6) or (5.44) with $\alpha_k = 10$. Due to the very abrupt changes of the control, only RANSAC with IE was able to accurately estimate the trajectory for the entirety of the measurement window.

Since the RANSAC with IE may have an unfair advantage due to the number of iterations it was able to perform to satisfy the expected value of ℓ, we also perform each algorithm where the number of RANSAC iterations ℓ for each algorithm was specified to be the $\mathrm{E}[\ell]$ calculated using (5.44), where again $\alpha_k = 10$, such that $\ell = 64800$ for all models. The results of this second experiment are shown in Fig. 6.17. While the non-input estimation algorithms improved, they are still less accurate than RANSAC with IE since they are unable to fit the maneuver according to their respective models. However, the increase in the number of iterations needed to achieve a good model for RANSAC with IE is non-trivial; therefore, the computational savings may warrant continued use of RANSAC implementations that do not estimate inputs. In the context of R-RANSAC, this conclusion is further validated considering that all of the lower order models converged to the true solution at the end of the trajectory, since the final measurement is always a good measurement that seeds each of the underlying minimum subsets. However, if accuracy along the entire trajectory is important,

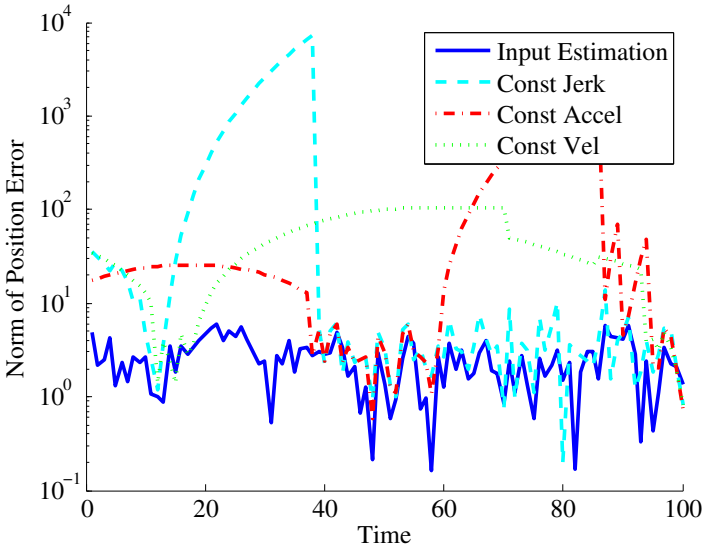

Figure 6.17: Input estimation compared to nearly constant velocity, acceleration, and jerk maximum likelihood RANSAC estimation, where the number of RANSAC iterations ℓ is given by the expected number of iterations needed for each model.

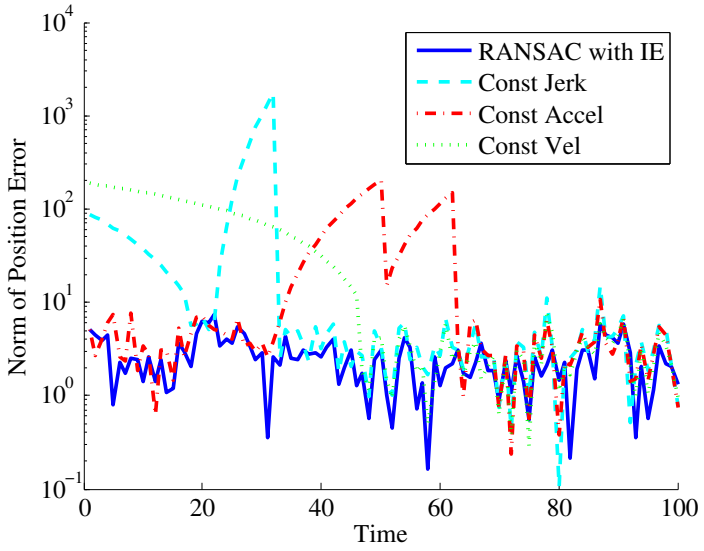

Figure 6.18: Input estimation compared to nearly-constant velocity, acceleration, and jerk maximum likelihood RANSAC estimation, where the number of RANSAC iterations ℓ is given by the expected number of iterations needed for the input estimation technique.

and the algorithm does not need to be run in real-time, then RANSAC with IE provides the most accurate estimate.

Table 6.8: Comparison of the RANSAC with IE with standard RANSAC algorithms with higher-order dynamic models.

	Varying ℓ		Fixed ℓ given by (5.44)	
	RMSE	$\lvert\chi\rvert$	RMSE	$\lvert\chi\rvert$
RANSAC with IE	2.929	92	2.929	92
NCJ	1527.7	62	20.18	82
NCA	425.8	33	47.72	76
NCV	62.09	16	75.31	54

6.3.2 IMM R-RANSAC Simulation

To show the advantage of using an IMM while tracking a maneuvering target, we conduct a simple simulation when there is a single target with no clutter. In both algorithms, we rely on the NCV model described in (6.1). The different models for the IMM filter are characterized solely by a different process noise. Bar-Shalom discusses the design of an IMM filter by using only three underlying models: a second-order 'benign motion model' to track non-maneuvering targets, a second-order 'maneuver model' to track objects during the maneuver, and a third-order 'maneuver detection model' to detect the onset of a maneuver [14]. Following the same logic, we utilize a second-order model for all models with low $\sigma_Q^{(1)} = 0.3$, medium $\sigma_Q^{(2)} = 3$, and high $\sigma_Q^{(3)} = 30$; the NCV model that we compare against uses $\sigma_Q = 3$. The Markov transition matrix is given by

$$\mathcal{P}(\mathbb{M}_k^{(\iota)} \mid \mathbb{M}_k^{(\iota')}) = \begin{bmatrix} 0.7 & 0.2 & 0.1 \\ 0.3 & 0.5 & 0.2 \\ 0.3 & 0.4 & 0.3 \end{bmatrix}.$$

Fig. 6.19 displays the results of the simulation while tracking a target trajectory denoted by the solid black line. Clearly, the IMM filter is better able to track the target during the turns. In scenarios with false alarms and missed detections with multiple crossing targets, the failures of the CV model are exacerbated, as observed in the video-based tracking of Section 6.2.

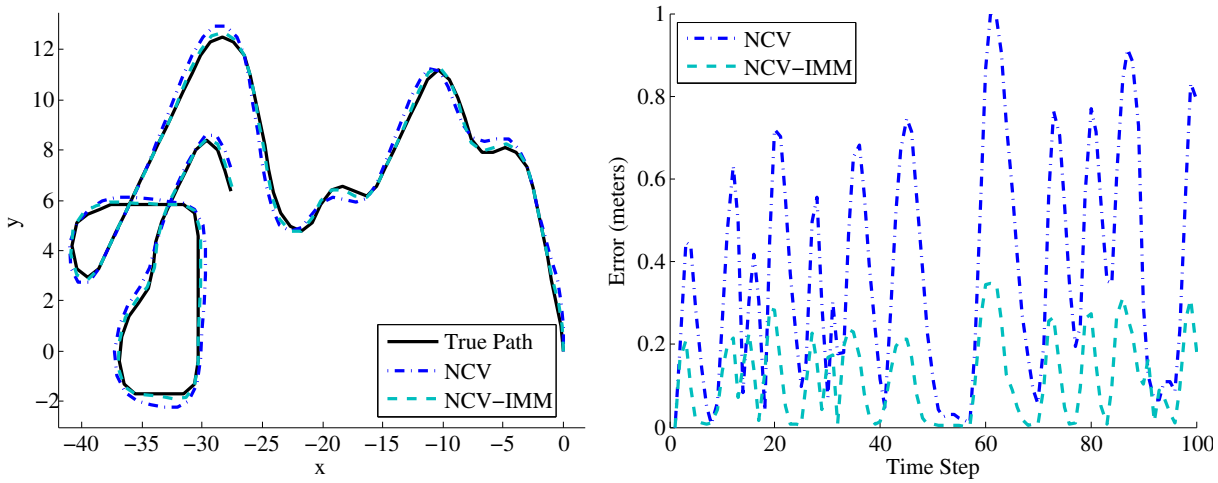

Figure 6.19: Comparison of tracking a maneuvering target using both a Kalman filter assuming a nearly-constant velocity (NCV) model and NCV interacting multiple model (NCV-IMM) filter.

6.3.3 NCV-IMM R-RANSAC in Video-Based Tracking

We now apply the IMM R-RANSAC filter to the same video as used in Section 6.2. In this case, we utilize the second-order equations based on (6.1) within the IMM filter, where the process noise covariance is varied. The three IMM modes used in this filter are $\sigma_Q^{(1)} = 0.3$, $\sigma_Q^{(2)} = 3$, and $\sigma_Q^{(3)} = 30$; the NCV model uses $\sigma_Q = 3$. Although the analysis of the full video is ongoing, we present the results of tracking a particularly difficult target trajectory, where the vehicle performs an unexpected right turn. Truth data is unknown, but the resulting R-RANSAC track using a NCV model is presented in Fig 6.20, and the resulting IMM R-RANSAC track is shown in Fig. 6.21. We emphasize that the IMM R-RANSAC was able to track the vehicle during the maneuver, whereas the a fixed NCV dynamics model lost and then reacquired the track. Additionally, the mixing probabilities during the lifetime of the R-RANSAC track are shown in Fig. 6.22, demonstrating the capability of the IMM to detect the onset of a maneuver.

Figure 6.20: R-RANSAC track assuming NCV dynamics of a vehicle performing an unexpected right turn. Notice the track fragmentation as the vehicle turns due to the underlying NCV dynamics assumption.

Figure 6.21: NCV-IMM R-RANSAC track of a vehicle performing an unexpected right turn.

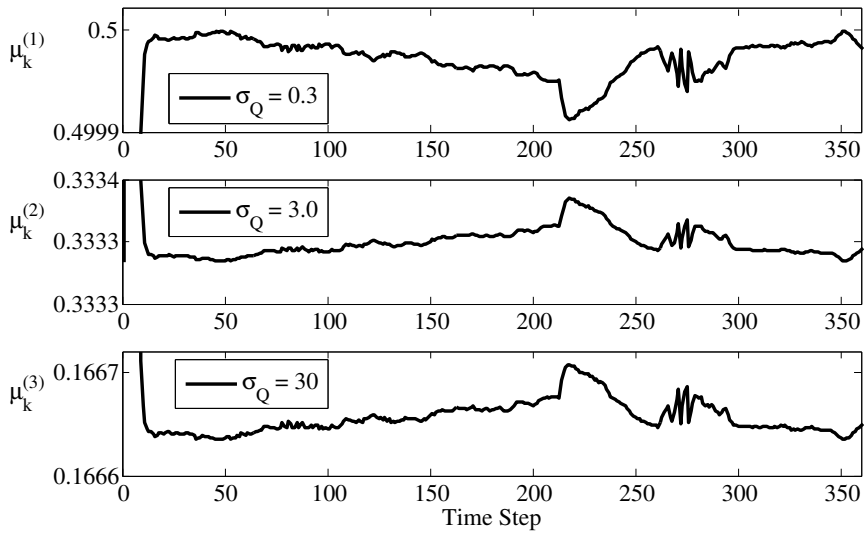

Figure 6.22: The mixing probabilities of the NCV-IMM filter while tracking a vehicle performing an unexpected right turn. Notice the subtle, but non-trivial, variation in the mixing probabilities when the maneuver occurs between time steps 200-250.

6.4 Robust Observer using R-RANSAC

Many autonomous systems are increasingly relying on sensor feedback to adaptively react to the environment. Sensor feedback can directly influence both low-level control and high-level logic. For example, accelerometers, gyros, altimeters, global positioning systems, and other sensors can be used to accurately estimate the vehicle states for control and stability. Cameras, radar, and other sensors can be used to avoid obstacles, plan paths, and adjust mission priorities. However, all sensors are prone to failure or other faults.

Sensor fault detection is the process of recognizing and robustly adapting to various sensor faults. Several of the possible types of sensor faults include spurious measurements, or clutter, corrupting the data set, change in noise distribution, sensor bias, sensor drift, etc. [24, 70]. Unless accounted for, these sensor faults could lead to instability and system failure, endangering the platform.

A classical method used to mitigate the effects of clutter is a *gated*-Kalman filter (KF). Gating is a heuristic used to distinguish between noisy measurements of the true signal and spurious measurements. Gating is a well-established technique that ignores measurements whose residual or weighted residual falls outside of some user-specified threshold [70,106,147,

152]. One advantage of the gated-KF is its low computational complexity. In scenarios when the probability of detection is high enough and the probability of failure is very low, then the gated-KF would be sufficient. However, the gated-KF is related to the nearest-neighbor approach in target tracking, which is known to perform poorly in high clutter. Similarly, the performance of the gated-KF significantly degrades as the probability of detection decreases. The gated-KF is effectively the nearest neighbor KF (NNKF) [12], but where only one measurement is received per time step.

Two alternatives to the gated-Kalman filter are the M-ary hypothesis testing and the KALMANSAC algorithms. Montgomery developed the M-ary hypothesis testing fault detection algorithm which maintains a bank of Kalman filters, each representing a different hypothesis of the state of the system [109]. A key difference is that in R-RANSAC, the bank of Kalman filters is not tuned a priori to predict the possible faults. Instead, R-RANSAC generates a new hypothesis for each new outlier, replacing the least-likely hypotheses.

The KALMANSAC algorithm in [161] is related to R-RANSAC in that both algorithms utilize the RANSAC paradigm to track a dynamic signal. However, a fundamental difference between R-RANSAC and KALMANSAC is that KALMANSAC relies on multiple iterations to compute a maximum a posterior estimate for each time step, while R-RANSAC stores multiple hypotheses in memory to allow subsequent inlier measurements to refine the current estimate. A basic summary of the KALMANSAC algorithm is to first select a random minimum subset of measurements to estimate the most likely states. Then, using the most likely states, it estimates the most likely inlier/outlier measurement labeling over the measurement set. These steps are iterated until a stopping criterion is satisfied. The resulting measurement labeling is used to seed the iteration of subsequent time steps. An advantage of R-RANSAC in MTT applications is its ability to track multiple signals [116], which is necessary in the third scenario considered in this section. However, in the case of a single signal, we expect the performance of the two algorithms to be similar.

We utilize the R-RANSAC algorithm as an observer to robustly estimate the states in a feedback control loop given the system inputs. In the remainder of this section, two types of sensor faults are analyzed: when faulty measurements are modeled as a Bernoulli distribution with parameter $p = \mathcal{P}_k(\omega = 1)$, and when the sensor variance increases half-way

through the simulation. It is shown that the gated-KF usually rejects outliers, but retains a non-trivial probability of divergence, while R-RANSAC is able to successfully estimate and track the desired altitude in all tested simulations. We also consider a scenario where the sensor can get 'stuck' in an undesired modality, which is modeled as a first-order Markov Chain. Only R-RANSAC is performed in this scenario since there are multiple true signals. This third scenario also shows that R-RANSAC successfully estimates the true states during short occlusions.

Simulation Parameters

In these scenarios, the objective is to control the altitude of an unmanned air system (UAS) to a desired height of 5 meters above the ground, where the altitude is measured directly using an altimeter. The parameters of the vehicle dynamics and observers for all simulations, unless otherwise stated, are specified as follows. The UAS altitude is modeled as a linear, constant velocity system as given in (6.1), with the dimension of the measurement vector $m = 1$, time step $\delta_k = 0.01$ s, process noise $\sigma_Q = 10$ and measurement noise variance ${\sigma_R}^2 = 0.1$. A proportional-derivative controller is implemented using the estimated states to control the altitude to be 5 meters above the terrain. The proportional gain is set to $k_\mathrm{p} = 1$, and the derivative gain is also set to $k_\mathrm{d} = 1$. The lateral velocity of the vehicle is 1 m/s, and the terrain can have a maximum slope of ± 1. Each simulation runs for 100 seconds, and 50 Monte Carlo simulations are performed at each parameter level. Due to the randomized initial conditions, we assume that good measurements, i.e. $\mathcal{P}_k(\omega = 1)$, are received for the first 5 seconds to arrive at a near steady-state condition before allowing false detections.

One difference from previous versions of R-RANSAC is that we define a second window length N_2 that is used to identify good hypotheses. The inlier ratio ρ is redefined as the ratio of the number of inliers in this second measurement window, $\rho_k^j = \frac{\left|\chi_k^j\right|}{N_2}$. Good hypotheses are identified as having an inlier ratio greater than $\rho_k^j > \tau_\rho$. When $N = N_2$, there is a trade-off between the variance of the inlier ratio and the memory requirement for storing measurements, where the variance of ρ is given in Theorem 3. Storing more measurements requires more memory, but significantly decreases the variance of ρ by averaging the number of inliers over a larger time window. By specifying two measurement windows, R-RANSAC

can reduce the memory requirement of storing while tracking inliers for a longer period of time. However, if N is too small, then there will be more outliers since the RANSAC estimate will be less accurate. Similarly, if N_2 is too large, then it will take longer to identify good hypotheses and will be slow to respond to signals varying significantly from their expected dynamic model.

Utilizing this slight modification, there are seven R-RANSAC parameters that need to be tuned. The number of stored R-RANSAC hypotheses is set to be $\mathcal{M} = 1$, since we know that there is no more than one true signal. Additional hypotheses are pruned such that only the hypothesis with the highest inlier ratio is kept. The measurement window is selected to store the previous $N = 75$ measurements and estimate the inlier ratio over $N_2 = 150$ measurements. The error threshold for both R-RANSAC and the gated-KF is set to $\tau_{\mathrm{R}} = 3\sigma_R$. A maximum of $\ell = 50$ RANSAC hypotheses are generated at each time step, and if a hypothesis is generated that contains more than γN inliers, where $\gamma = p$, we assume that the hypothesis is accurate and allow early termination. Similar hypotheses are combined if the estimated altitude is within 0.5 meters and the velocity is within 4 m/s.

Note that in the following, the R-RANSAC parameters could be optimized to improve computational efficiency for each particular scenario; however, we choose to use the same parameters for all simulations for better comparison. When the probability of detection is high enough, the window length N and the number of RANSAC iterations ℓ could be decreased to improve computational performance without degrading the tracking capabilities. Similarly, in scenarios with very low probability of detection, N and ℓ could be increased to improve performance at the expense of computational complexity. All simulations rely on synthetic data and were performed in MATLAB using a 64-bit 3.0 GHz Core 2 Duo processor, where it was observed that R-RANSAC is capable of real-time performance.

Gross Error Detection

The first sensor fault type under consideration is when the probability of detection p is varied. As described in (5.3), a valid detection is measured with additive zero-mean Gaussian noise with variance of ${\sigma_R}^2 = 0.1$ meters. However, a faulty measurement is received with probability $1 - p$; without loss of generality, the gross error noise is additive and drawn

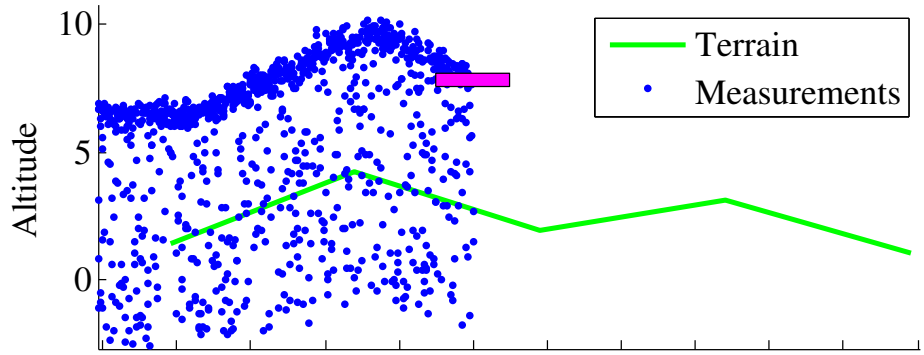

Figure 6.23: Example simulation where gross errors are observed 50% of the time and the variance of the measurement noise is ${\sigma_R}^2 = 0.1$ meters.

from a uniform distribution $\mathcal{U}(\underline{\varsigma}, \overline{\varsigma})$, where $\underline{\varsigma} = \mathbf{x}[k] - 10$ and $\overline{\varsigma} = \mathbf{x}[k]$. Note that the gross errors are biased and will therefore bias traditional observers like the KF. The probability of detections considered are

$$p_{\mathrm{D}} \in \big\{1.0,\ 0.99,\ 0.95,\ 0.9,\ 0.8,\ 0.7,\ 0.6,\ 0.5,\ 0.4,\ 0.3,\ 0.2\big\}.$$

A screen shot of the simulation during a representative example is shown in Fig. 6.23, with probability of a good measurement $p = 0.5$.

The results are shown in Fig. 6.24. We observe that the gated-KF works well when the probability of detection remains fairly high, $p > 0.6$. However, we claim that the gated-KF is not provably robust when $p < 1$. This is because there is no mechanism to 'catch' when the states begin to diverge. We find that R-RANSAC is able to successfully estimate the states in lower probability of detection, allowing the vehicle to be accurately controlled to the desired altitude. We find that the KF gradually diverges from estimating the true states and tracking the desired altitude. The reason that the KF does not experience instability is that the gross errors introduce a constant bias, and so their effect gradually takes over the zero-mean effect of the true measurements, causing the vehicle to believe it is tracking well, when in fact it is biased due to the gross errors. As seen in Fig. 6.24, errors in the KF estimate and resulting control are immediately observed when $p < 1$, as the mean RMSE increases as the probability of detection decreases.

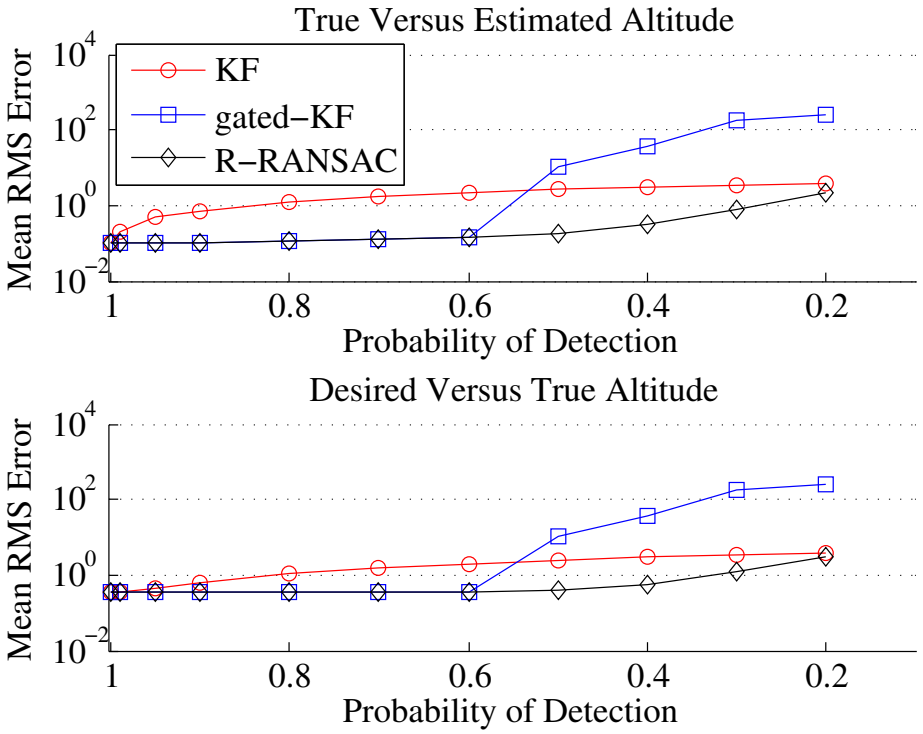

Figure 6.24: RMSE of the Kalman filter, gated-Kalman filter, and R-RANSAC when decreasing the probability of detection.

Sensor Noise Change

The second sensor fault type under consideration is when the measurement variance ${\sigma_R}^2$ deviates from the expected value halfway through the simulation. In this scenario, the probability of detection is set to $p = 1.0$ to remove the effect of gross errors. For the first half of the simulation the measurement noise variance is ${\sigma_R}^2 = 0.1$ meters as above. At time 50 seconds, we multiply ${\sigma_R}^2$ by a scalar α_R, such that the new variance for the duration of the simulation is $\alpha_R {\sigma_R}^2$. Figure 6.25 shows the results of the Monte Carlo simulations, where

$$\alpha_R \in \{0.1,\ 0.5,\ 1,\ 1.5,\ 2,\ 2.5,\ 3,\ 3.5,\ 4,\ 5,\ 6,\ 7,\ 8,\ 9,\ 10\}.$$

As before, we find that the gated-KF eventually diverges as measurements are received in such a way to cause the mean of the gate region of the measurements to drift away from the true mean. At $\alpha_R = 3.5$, the gated-KF probability of diverging is large enough such that it begins to diverge, though no divergence occurs during the Monte Carlo simulations

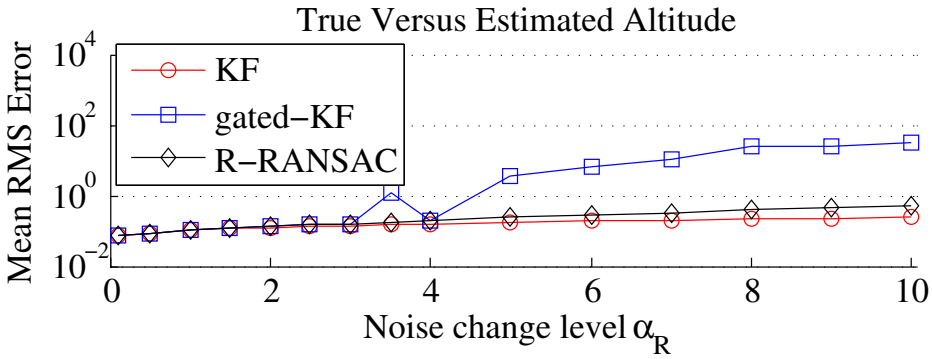

Figure 6.25: RMSE of the Kalman filter, gated-Kalman filter, and R-RANSAC when the variance of the measurement noise changes from ${\sigma_R}^2 = 0.1$ to ${\sigma_R}^2 = \alpha_R 0.1$ half-way through the simulation.

at $\alpha_R = 4$. We find the standard KF to be quite accurate since the measurements are zero-mean Gaussian about the true signal. However, the covariance becomes inconsistent as the KF does not account for the change in the sensor statistics. Due to missed measurements outside of the inlier region, the covariance of the gated-KF and R-RANSAC both increase after the sensor change. The results are summarized in Fig. 6.25, where the desired vs. true altitude plot is removed for brevity as it closely tracks the true vs. estimated states as in Fig. 6.24. Note that if $p < 1$, as in the previous section, R-RANSAC is the only algorithm that can robustly compensate for both types of sensor faults simultaneously.

UAS Flying Along a Cliff Edge

The final simulation under consideration is when a UAS flies along a cliff edge, where the terrain at the top and bottom of the cliff is assumed to be flat. Due to errors in the lateral dynamics of the UAS or the uneven edges of the cliff, we expect that the altimeter will operate in one of two modalities and report either the altitude to the top or bottom of the cliff. A screen shot of the simulation is shown in Fig. 6.26, where the top of the cliff has a higher altitude but results in smaller measurements.

The modality of the sensor at the current time step is not independent of the sensor modality from the previous time step, so we model the random process $\mathcal{P}_k(\omega)$ as a first-order Markov chain, as opposed to a Bernoulli process, since it is expected that both sensor modalities will persist for a time as the sensor switches between two different regimes. The

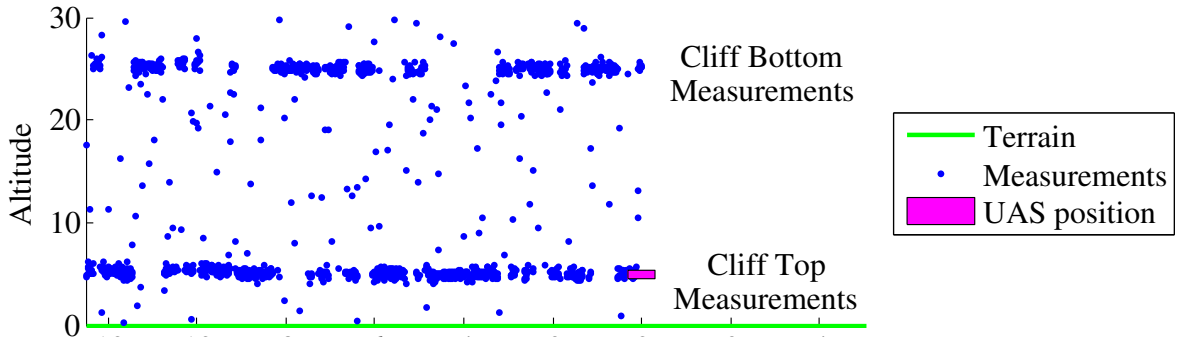

Figure 6.26: Example simulation where a UAS is to fly 5 meters above the top of the cliff (denoted at 0 meters), but measures either the top or bottom of the cliff according to a first-order Markov chain.

Markov chain transition matrix is assumed to be symmetric with transition probability $q \in [0, 1]$. In other words, if the sensor returns a valid measurement at time $k-1$, the sensor will continue to operate in that modality with probability $q = 0.96$ and transition to a faulty measurement with probability $1-q$, and vice versa.

In this case, we model the faulty measurement distribution as $\Theta \sim \mathcal{N}(\mathbf{x}[k] + c, {\sigma_R}^2)$, where $c = 20$ meters is the cliff height. The objective is to maintain 5 meters altitude above the top of the cliff. In essence, this scenario explores sensor occlusions, where for some period of time the true signal is not measured. In addition, we also include a Bernoulli process to describe faulty measurements, or that with probability $p = 0.9$ a valid measurement is observed and with probability $1 - p$ a faulty measurement is observed, regardless of the modality in which the sensor is operating. We assume the faulty measurements are uniformly distributed between $\underline{\varsigma} = 0$ meters and $\overline{\varsigma} = c + 10$ meters.

In the following simulation, we explore the effects of the measurement window length N_2, which is allowed to vary between

$$N_2 = \{3,\ 4,\ 4.5,\ 5,\ 5.5,\ 6,\ 7,\ 8,\ 9,\ 10,\ 11,\ 12\} \times 10^2$$

time steps. The good track threshold is set to $\tau_\rho = 0.1$. Since there are now two true signals to track, we set the number of stored hypotheses to $\mathcal{M} = 2$. Instead of using the track with the highest inlier ratio, as before, we now use the track with the lowest estimated altitude

among the set of tracks with $\rho_k^j > \tau_\rho$ to prevent ground collisions. If there are no valid tracks, we simply use the track with the lowest estimated altitude among the stored hypotheses.

One final modification was made after some preliminary analysis of the results. While not included, it was observed that when the process noise exceeded a certain value, then R-RANSAC diverged during longer occlusions. Obviously, if there is a time window with fewer than $p\tau_\rho N_2$ valid measurements, then we expect to lose track of the signal. Therefore, the window length should be sufficiently long to account for the expected occlusion length. However, we found that as the process noise increases, then even if N_2 is sufficiently large, the drift during the occlusion can cause R-RANSAC to lose track of the signal. Even if $\mathcal{M} > 2$, since N is large, it may take a long time for it to drop below the good hypothesis threshold, during which time significant divergence can be expected. For this reason, the process noise was reduced to $\sigma_Q = 3$.

A Monte Carlo simulation demonstrates that when N_2 is too small or large, the R-RANSAC estimate may diverge due to tracking the wrong signal or incorrectly tracking a low probability gross error signal, respectively. Results are shown in Fig. 6.27, where the true vs. desired altitude is not included as it closely tracks the true vs. estimated states. It is anticipated that if there was only a single true signal, then the good hypothesis logic can be modified to reacquire drifting signals after long occlusions. This will be the subject of future work.

The states of a single run are shown in Fig. 6.28, when $N_2 = 500$. The top sub-figure shows which modality the sensor is operating in, not including the gross errors. The bottom figure shows the unsaturated force commands sent to the system. The remaining sub-figures show the desired, true, and estimates of both the vehicle height and velocity. As shown in subplot of the sensor modality in Fig. 6.28, the sensor switches between modalities frequently, and the approximate maximum time in a single modality is about 1 second (100 time steps). The R-RANSAC algorithm is able to track through these 'occlusions' and we see that the altitude and altitude velocity relative to both the top and the bottom of the cliff are accurately estimated. There is some initial velocity error of the hypothesis of the states relative to the bottom of the cliff as new measurements are first received at time 5 seconds, but these errors quickly settle to the correct values.

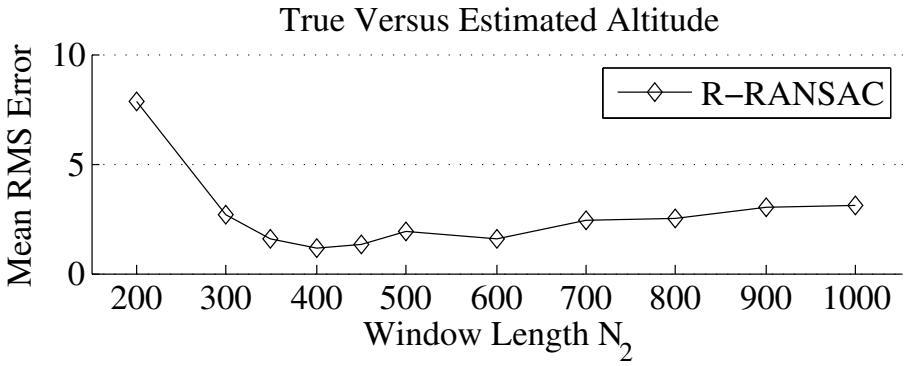

Figure 6.27: Monte Carlo simulation varying the R-RANSAC window length N_2 where a UAS measures either the top or bottom of a cliff according to a first order Markov chain. We observe that R-RANSAC successfully accounts for the occlusions and drift when the window length is not too small or large.

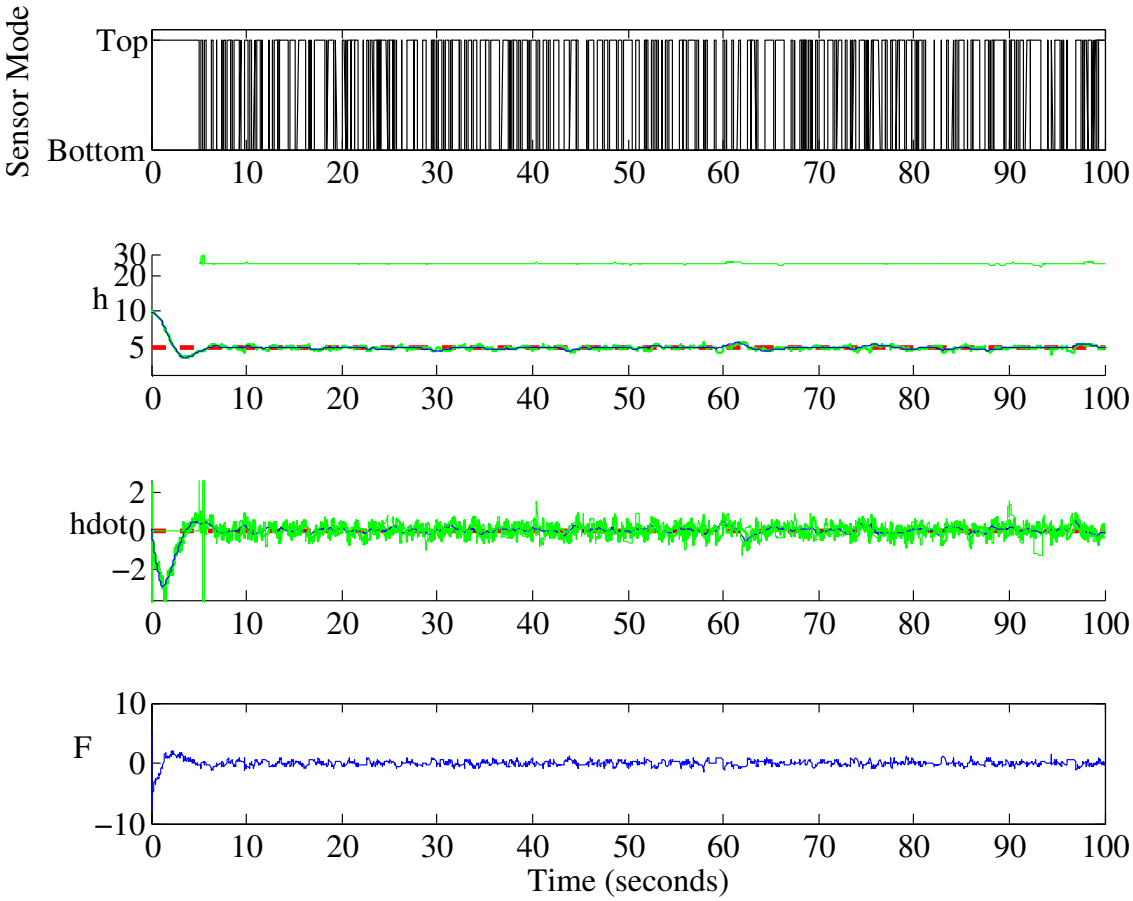

Figure 6.28: UAS states when flying along edge of a cliff and measuring both the top and bottom. Both estimates generated by the R-RANSAC algorithm are shown, which generally track both the top and bottom of the cliff.

Chapter 7. Conclusion

7.1 Summary

The main contribution of this dissertation is the development of a novel multiple target tracking (MTT) algorithm called recursive-RANSAC (R-RANSAC). The R-RANSAC algorithm is related to the track-oriented MHT filter. While the track-oriented MHT filter stores a set of tracks between frames, the hypothesis tracks are formed by exhaustively enumerating the possible measurement assignments. This is a brute-force search of the data association space. On the other hand, R-RANSAC uses the current set of measurements to either update existing tracks or generate new tracks that best characterize unassigned measurements. New tracks are formed using RANSAC with the past N measurement scans to estimate the data association. The resulting tracks are merged and pruned to keep only the best $\mathcal{M}$ tracks between time steps. Good tracks are formed when the inlier ratio given by (2.10) is greater than a threshold.

Some of the supporting contributions in the development of R-RANSAC are to prove in Theorem 1 that the worst-case complexity of R-RANSAC is proportional to the number of measurements squared and the number of stored tracks squared, $\mathcal{O}\Big(\ell\, N\, {\psi_k}^2 + \mathcal{M}^2\Big)$. We also show in Theorem 2 that the R-RANSAC state estimates converge in mean to the true target states for well-separated targets. In Chapter 3, we develop the R-RANSAC algorithm to recursively estimate the parameters of multiple static signals in clutter, and we show in Chapter 4 that R-RANSAC is more robust than existing single signal parameter estimation algorithms such as sequential RANSAC and gated recursive least-squares (see Fig. 4.1). We also show in Chapter 4 that R-RANSAC is able to robustly track multiple ground reflectors using synthetic aperture radar (SAR) returns (see Fig. 4.16) that can be employed in GPS-denied radar odometry-aided navigation; in this test, the execution rate

of R-RANSAC is almost 20 times faster than the Hough transform. In Chapter 5, we develop the R-RANSAC algorithm to recursively estimate the states of multiple dynamic targets in clutter. As part of that development, we derive a maximum likelihood RANSAC-based technique to estimate the initial conditions of a hypothesis trajectory from a set of measurements (see (5.24)), we account for possibly known inputs when R-RANSAC is used within a feedback loop (see (5.32)), and we also develop a maximum likelihood RANSAC-based technique to simultaneously estimate the initial conditions and a piecewise-constant input vector estimating the control inputs of a maneuvering target (see (5.37)). In Chapter 6, R-RANSAC is compared against five existing MTT algorithms: the GNN, JPDA, MHT, GM-PHD, and MCMCDA filters. We find that R-RANSAC achieves excellent track continuity compared to the optimal MHT filter and other algorithms (see Fig. 6.8) and is also accurate and computationally efficient. Also in Chapter 6, R-RANSAC is applied to video-based tracking examples and is shown to achieve much better track continuity than the GM-PHD filter (see Figs. 6.11-6.14).

While we do not claim that this algorithm should supplant the many tracking algorithms found within the literature for every situation, we do believe that R-RANSAC should be considered for a wide range of tracking scenarios. Some advantages of R-RANSAC are that it is intuitive and simple, which facilitates implementation and allows for efficient computation and autonomous track management. While originally developed in MATLAB, we find that a direct implementation of R-RANSAC in C++ is very efficient. Tracking 5-8 targets on an 64-bit 3.4 GHz Core i7-4770 processor, using a nearly-constant jerk IMM R-RANSAC algorithm with PDA measurement weights, has an execution rate of over 270 Hz. We have also observed very good track continuity of detected tracks. Disadvantages include its reliance on heuristics and user-specified parameter settings; one possible improvement to R-RANSAC would be to incorporate adaptive thresholds and parameter settings to account for variations in the estimated, but unknown, environmental parameters. Additionally, the computational complexity of R-RANSAC increases as the ratio of clutter returns to the number of valid measurements increases. However, the framework of R-RANSAC is flexible enough to incorporate additional heuristics to maintain real-time performance in complex

scenarios. Some of these unexplored improvements include guided sampling, enforcing minimum velocity constraints, and parallel processing during the hypothesis generation step.

7.2 Future Work

There remain several viable avenues of future research with regard to further development of the R-RANSAC algorithm. This section describes some of the major avenues of research that this dissertation does not consider, but that should be considered in future work. This section is obviously not an exhaustive enumeration of the possible extensions, but lists some of the specific opportunities for possible extensions and related applications.

Nonlinear Dynamics

One of the more obvious extensions is to apply R-RANSAC to a system with nonlinear dynamics. We emphasize, as discussed in Chapter 2, that there is nothing with regard to the R-RANSAC framework that precludes the use of nonlinear filtering techniques such as the extended Kalman filter, unscented Kalman filter, or others. Instead of the linear update and prediction steps in Algorithm 2, the appropriate nonlinear propagation and update steps should be used.

The only unresolved issue is how to appropriately initialize a new trajectory using RANSAC. With linear systems theory, the maximum likelihood technique developed in Section 5.2.1 is a natural way to estimate the initial states of hypothesis trajectories using minimum subsets of measurements. However, to our knowledge, there is no such technique for a general nonlinear system. We have examined using specific parameterizations for particular problems, but as of yet we have not developed a solution for the general problem. One of the possible solutions is to use nonlinear smoothing plus filtering to generate RANSAC hypotheses [63, 98].

Moving Reference Frame

With several colleagues at United Technologies Research Center, we compared R-RANSAC to the cardenalized PHD filter in the context of video-based tracking from an aerial

platform [64]. We used a standard implementation of R-RANSAC modeling constant velocity dynamics and used an all-neighbors weighting function of the measurements. R-RANSAC performed as expected when the video was stable and smooth. However, whenever there was a sudden motion of the camera due to turbulence or gimbaling, R-RANSAC generated many false tracks and generally lost track of the target. We found that since we were tracking in the camera frame, without accounting for the ego motion of the camera, the previous measurements in the measurement window do not accurately reflect the trajectory of the target. If the homography between the two frames was known exactly, then this transformation could be applied to all of the measurements within the window. However, estimating the homography is not trivial when the image is not purely stationary with dynamic objects.

Feature-Aided Recursive-RANSAC

Another method that can be incorporated into the R-RANSAC algorithm is to utilize features to enhance the data association process. In images, some types of features include color, texture, and size; in radar, intensity or aspect ratio may be used. Some examples of existing trackers that use this technique include [14, 65, 169]. Developing a feature-aided R-RANSAC algorithm would require modifying the measurement weights w_{ij} based on the likelihood of a measurement matching the tracked target.

Feature-aided tracking is one heuristic that may help with a broader class of problems related to unresolved targets. Essentially, unresolved targets occur when the targets are too closely spaced to be effectively differentiated [14]. In these circumstances, one or no measurements may occur, and measurements that are detected may have non-Gaussian noise. In short, unresolved measurements are a manifestation of a break-down of the standard assumption that the sensor detection step in Fig. 1.1 is independent from the data association and filtering steps.

Distributed Tracking

In wide-area surveillance for a single tracking system, there is a trade-off in the resolution of the imagery and the volume of the surveillance region. However, by utilizing a distributed network of sensors, the maximum volume of the surveillance region is only limited by the number of sensors. R-RANSAC lends itself to distributed tracking since it maintains a set of tracks that can be shared between neighboring agents. In this way, track 'hand-offs' can be achieved to maintain track continuity of single targets. Additionally, this framework would facilitate the fusion of heterogeneous sensor data.

Hierarchical tracking is related to distributed tracking. In these scenarios, a high level tracking algorithm is in charge of track maintenance. Under the direction of a higher level tasking algorithm, lower level agents are in charge of performing a task with regard to one or more of the tracked targets. Transferring information from the high level tracker and the low lever tracker is called a hand-off, e.g. where the tasked agent must correctly track the object identified by the higher level tracker. We expect that a similar structure could be formulated for R-RANSAC, where a high-level tracker manages the majority of the track initiation and validation and hands-off validated tracks to lower-level agents for specific tasking.

R-RANSAC SLAM

Recently, Mullane et al. developed a feature-based simultaneous localization and mapping (SLAM) algorithm that uses the GM-PHD filter to track a set of features in the environment [111]. The joint state and map distribution is solved using a technique called Rao-Blackwellization, where the vehicle states are estimated using a particle filter with an associated feature map, while the feature maps, conditioned on the vehicle state, are estimated using the GM-PHD filter. Besides showing improved performance over the more traditional extended KF-based FastSLAM approaches [108], Mullane et al. suggest that the PHD filter is a more natural solution for map estimation since it simultaneously estimates both the feature location and number of features; i.e., there is no need for a feature (or track) management system as with the extended KF. Since the PHD-SLAM problem is partitioned

using Rao-Blackwellization, we expect that the R-RANSAC algorithm could be applied in a similar manner to estimate the feature locations in the trajectory conditioned maps.

7.3 Discussion

In this final section, we discuss R-RANSAC in a high-level context; specifically, which tracking scenarios are well-suited to the R-RANSAC approach. One challenge of delving into the MTT literature is the breadth and depth of a field that has matured over the past 60+ years by approaching the MTT and data association problems using a number of different techniques. That being said, MTT is still an active area of research and many new tracking algorithms continue to be developed. Obviously, we have not been able to test R-RANSAC against every possible algorithm in the wide-range of tracking scenarios that have been considered. However, by comparing it to some of the standard algorithms in the literature, we feel like we have some intuition of the capabilities of R-RANSAC.

The main take-away about R-RANSAC is that it is, at worst, an excellent starting point for developing an efficient and effective tracking system. While tracking and surveillance is a goal in and of itself, frequently tracking is only a means to an end, where a simple, robust, and autonomous solution is desired in order to perform higher level tasks. Some of these higher level tasks that are being considered in the BYU MAGICC Lab and the National Science Foundation's Center for Unmanned Air Systems (C-UAS) include

- Sense and avoidance algorithms to help integrate UAS into the national airspace.
- Autonomous landing detection for autonomous vehicles in an urban environment.
- Feature tracking in a GPS-denied environment, using both camera and radar senors, to estimate sensor odometry.
- State estimation in feedback loops when there exist sensor faults in GPS or other sensors.
- Distributed tracking and localization by a team of robots.

In these and many other applications in robotics, we desire a 'plug-and-play' tracking algorithm that requires minimal coding effort, computational requirements, and reliable tracking.

While MHT and MCMCDA provide very accurate tracking results, they are very difficult to code and they require significant computational resources. The JPDA is simple and efficient for sparse targets, but requires more computational complexity in a dense target environment and still requires the implementation of a separate track management algorithm to initialize tracks. We believe that this explains why some researchers opt to use the simple and efficient GNN algorithm with a track management algorithm to perform autonomous tracking [90], even though GNN is known to diverge in clutter. R-RANSAC can be used as a building block within an embedded system to perform tracking.

Depending on the data association method selected, we have found that R-RANSAC tracks tend to coalesce when targets remain close together, as also seen in the PDA and JPDA algorithms. This suggests that R-RANSAC, without additional development, is ill-suited for tracking clustered targets where the targets are closely spaced and travel for extended periods of times together. We expect that one or more tracks may be generated to describe the cluster, but that R-RANSAC will underestimate the number of targets. Another area where R-RANSAC may struggle is when the clutter rate is extremely high. In these scenarios, the window length N and the number of RANSAC iterations ℓ needed to identify the targets may substantially increase the computational complexity of R-RANSAC.

One key concept behind both the RANSAC and R-RANSAC algorithms is the minimal subset of measurement data, which is used to generate hypotheses of the track of interest. The expected convergence rate of both algorithms is dependent on the cardinality of the minimum subset: the larger the minimum subset, the more hypotheses must be generated to find a minimum subset of all true measurements of the signal of interest. In cases where s is very large, the number of hypotheses may become unwieldy. Thankfully, the hypothesis generation step of RANSAC is parallelizable, but the fact remains that very large s may render R-RANSAC too computationally intensive.

An anonymous reviewer for a paper that we submitted correctly noted that R-RANSAC will have trouble identifying short duration signals. This is true and also a challenge for many of the MTT algorithms that rely on an M/N-like detection scheme. For example, during pedestrian tracking it would be difficult to detect a pedestrian that only briefly passes through the corner of the camera field of view before exiting. In our opinion,

at least in that particular example, this is mostly a problem with insufficient sensor coverage of the environment. A related scenario is when the same target is measured only for brief intervals, for example, due to occlusions. We expect R-RANSAC to successfully track these targets if the measurement window is large enough and the target dynamics do not change too much during the occlusions.

This leads us to another open problem in the MTT community, which is tracking maneuvering targets through occlusions. Throughout this dissertation, we have used a constant velocity model to track maneuvering targets. When a target is occluded and is maneuvering in such a way that its trajectory does not fit the underlying model, R-RANSAC usually initializes a new track after the occlusion. There is a trade-off between the size of the measurement window and the frequency of target maneuvers and occlusions. As we more fully incorporate the interacting multiple model (IMM) filter into the R-RANSAC framework, we have seen some improvement in this regard, but we expect that the IMM R-RANSAC filter also has its limits, which we have not fully explored.

A final open topic is how sensitive the R-RANSAC algorithm is to environmental parameters. In Chapter 6, we look at the sensor fault condition when the measurement noise variance suddenly increases. In this particular scenario, R-RANSAC handles the switch gracefully, but what about other conditions? Scenarios where the process noise is different, or the probability of detection is unknown, are interesting future case studies. One benefit of the R-RANSAC algorithm is that the clutter rate does not need to be known, unless the underlying measurement weighing function relies on it. Lastly, it is difficult to tune the R-RANSAC good track threshold to detect targets with differing levels of probability of detection. R-RANSAC can either track the low p_{D} targets without ever declaring it a track, or R-RANSAC can validate low p_{D} target tracks at the expense of an increased false alarm rate.

In conclusion, we remind the reader of the analogy that motivated the development of R-RANSAC: RANSAC is to least-squares (LS) as R-RANSAC is to recursive LS (or the Kalman filter). In regression analysis, the LS algorithm achieves the minimum mean-squared error solution when there are no spurious measurements or clutter in the data set. RANSAC robustly and efficiently estimates the parameters of a single signal from a cluttered

data set by forming many hypotheses and selecting the hypothesis with the best support. RANSAC has since become nearly ubiquitous in the computer vision community due to its simplicity and robustness. When there are sequential measurements, recursive LS converges in mean to the LS solution [69] but also diverges in clutter. The R-RANSAC algorithm extends RANSAC to sequentially track signals by storing multiple hypotheses between time steps, which naturally allows for MTT. Through its simplicity, R-RANSAC is an attractive alternative for researchers seeking an algorithm to robustly estimate the states of multiple signals for a wide range of applications.

Appendix A. Weighted Measurement Kalman Filter

Under the assumption that the system dynamics and measurements are linear time-invariant, then the evolution of the states $\mathbf{x} \in \mathbb{R}^n$ and measurements $\mathbf{y} \in \mathbb{R}^m$ can be written in matrix form

$$\mathbf{x}_k = A\mathbf{x}_{k-1} + \mathbf{w}, \tag{A.1}$$

$$\mathbf{y}_k = C\mathbf{x}_{k-1} + \mathbf{v}, \tag{A.2}$$

where $\mathbf{w}$ and $\mathbf{v}$ are zero-mean Gaussian processes with covariance Q and R, respectively. It is well-known that the minimum mean-squared estimator of linear time-invariant systems is the Kalman filter [80, 110]. In the presence of multiple measurements, the Kalman filter is propagated as normal; given an initial condition $\mathbf{x}_{k-1|k-1}$ and initial covariance $P_{k-1|k-1}$, the propagated states and covariance are given by

$$\hat{\mathbf{x}}_{k|k-1} = A\,\hat{\mathbf{x}}_{k-1|k-1}, \tag{A.3}$$

$$P_{k|k-1} = A\,P_{k-1|k-1}\,A^\top + Q. \tag{A.4}$$

When multiple weighted measurements are to be included in the measurement update, we fuse the measurements into a single pseudo-measurement and update the states and covariance using a modified Kalman filter, as done in the probabilistic data association (PDA) filter [13]. Following the derivation in [19], the steps of the update are given in the following. First, we assume that the propagated states $\hat{\mathbf{x}}_{k|k-1}$ and covariance $P_{k|k-1}$ are available, along with a set of ψ_k measurements $\mathcal{Z}_k = \{\mathbf{z}_k^1, \ldots, \mathbf{z}_k^{\psi_k}\}$ and associated weights w_j for $j = 0, \ldots, \psi_k$; note that the weight w_0 represents the probability of a missed detection. The

next step is to fuse the weighted measurements into a single pseudo-measurement

$$\tilde{\mathbf{z}}_k = \sum_{j=1}^{\psi_k} w_j \big(\mathbf{z}_k^j - C\,\hat{\mathbf{x}}_{k|k-1}\big). \tag{A.5}$$

The measurement $\tilde{\mathbf{z}}_k$ is then used to update the estimated states as normal

$$\hat{\mathbf{x}}_{k|k} = \hat{\mathbf{x}}_{k|k-1} + L_k\,\tilde{\mathbf{z}}_k, \tag{A.6}$$

where $L_k = P_{k|k-1}C^\top \left(C\,P_{k|k-1}C^\top + R\right)^{-1}$ is the Kalman gain. The covariance update has some additional terms to account for possible missed detections and association uncertainty,

$$P_{k|k} = (1 - w_0)P^*_{k|k} + w_0\,P_{k|k-1} + dP_k, \tag{A.7}$$

where

$$\begin{aligned} P^*_{k|k} &= \Big(I_m - L_k\,C\Big)P_{k|k-1}, \\ dP_k &= L_k \left[\sum_{j=1}^{\psi_k} w_j \left(\mathbf{z}_k^j - C\,\hat{\mathbf{x}}_{k|k-1}\right)\left(\mathbf{z}_k^j - C\,\hat{\mathbf{x}}_{k|k-1}\right)^\top - \tilde{\mathbf{z}}_k\tilde{\mathbf{z}}_k^\top\right] L_k^\top, \end{aligned}$$

where we note that $P^*_{k|k}$ is the nominal Kalman filter updated covariance matrix.

Appendix B. Maximum Likelihood Derivation

We desire to find the maximum likelihood (ML) estimate of the general linear equation

$$Y = \mathcal{O}\mathbf{x} + \xi.$$

The maximization problem can be posed as

$$\hat{\mathbf{x}} = \max_{\mathbf{x}} p(Y \mid \mathbf{x}), \tag{B.1}$$

or in other words finding $\mathbf{x}$ that maximizes the probability of receiving the measurement Y given $\mathbf{x}$. Due to the Gaussian distribution of the noise, we simply take the derivative of the log-likelihood $\Lambda(\mathbf{x})$, set it equal to zero, and solve for $\mathbf{x}$.

The distribution $p(Y \mid \mathbf{x})$ is given by

$$p(Y \mid \mathbf{x}) = \frac{1}{\sqrt{(2\pi)^n \left|\Xi\right|}} e^{-\frac{1}{2}(Y-\mathcal{O}\mathbf{x})^\top \Xi^{-1}(Y-\mathcal{O}\mathbf{x})}. \tag{B.2}$$

The log-likelihood of this function is given by

$$\begin{aligned} \Lambda(\mathbf{x}) &= \log p(Y \mid \mathbf{x}) \\ &= -\frac{1}{2}\log\left(2\pi\right)^n \left|\Xi\right| - \frac{1}{2}(Y - \mathcal{O}\mathbf{x})^\top \Xi^{-1}(Y - \mathcal{O}\mathbf{x}). \end{aligned} \tag{B.3}$$

The derivative of the $\Lambda(\mathbf{x})$ is given by

$$\begin{aligned}\frac{\partial \Lambda(\mathbf{x})}{\partial \mathbf{x}} &= 0 - \frac{1}{2}\frac{\partial}{\partial \mathbf{x}}\left((Y - \mathcal{O}\mathbf{x})^\top \Xi^{-1}(Y - \mathcal{O}\mathbf{x})\right) \\ &= -\frac{1}{2}\frac{\partial}{\partial \mathbf{x}}\Big[Y^\top \Xi^{-1} Y + \mathbf{x}^\top \mathcal{O}^\top \Xi^{-1}\mathcal{O}\mathbf{x} - Y^\top \Xi^{-1}\mathcal{O}\mathbf{x} - (Y^\top \Xi^{-1}\mathcal{O})\mathbf{x}\Big] \\ &= -\frac{1}{2}\Big[2\mathcal{O}^\top \Xi^{-1}\mathcal{O}\mathbf{x} - 2\mathcal{O}^\top \Xi^{-1} Y\Big] \\ &= \mathcal{O}^\top \Xi^{-1} Y - \mathcal{O}^\top \Xi^{-1}\mathcal{O}\mathbf{x}. \end{aligned} \tag{B.4}$$

The maximum value is attained when $\frac{\partial \Lambda(\mathbf{x})}{\partial \mathbf{x}} = 0$. Therefore, the ML estimate is given by

$$\hat{\mathbf{x}} = \left(\mathcal{O}^\top \Xi^{-1}\mathcal{O}\right)^{-1} \mathcal{O}^\top \Xi^{-1} Y, \tag{B.5}$$

with covariance

$$P = \left(\mathcal{O}^\top \Xi^{-1}\mathcal{O}\right)^{-1}. \tag{B.6}$$

Bibliography

[1] "Detecting Cars Using Gaussian Mixture Models - MATLAB & Simulink Example." [Online]. Available: http://www.mathworks.com/help/vision/examples/detecting-cars-using-gaussian-mixture-models.html 130

[2] P. Addesso, R. Conte, M. Longo, R. Restaino, and G. Vivone, "MAP-MRF Cloud Detection Based on PHD Filtering," *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, vol. 5, no. 3, pp. 919–929, 2012. 1

[3] I. Ahmed, "Dolphin Whistle Frequency Estimation using Gaussian Mixture Probability Hypothesis Density Filter," *International Journal of Computer Vision and Signal Processing*, vol. 1, no. 1, pp. 9–14, 2012. 1

[4] K. Al-Mutib, E. Mattar, M. AlSulaiman, H. Ramdane, and M. Emaduddin, "Mobile Robot Floor Navigation using RGB+D and Stereo Cameras," *International Journal of Soft Computing and Engineering*, vol. 3, no. 3, pp. 286–292, 2013. 5

[5] K. O. Arras, S. Grzonka, M. Luber, and W. Burgard, "Efficient People Tracking in Laser Range Data Using a Multi-Hypothesis Leg-Tracker with Adaptive Occlusion Probabilities," in *IEEE International Conference on Robotics and Automation*, 2008, pp. 1710–1715. 1

[6] S. Avidan, "Ensemble Tracking." *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 29, no. 2, pp. 261–71, 2007. 14

[7] D. Avitzour, "Stochastic Simulation Bayesian Approach to Multitarget Tracking," *IEE Proceedings - Radar, Sonar and Navigation*, vol. 142, no. 2, p. 41, 1995. 12

[8] A. Bal and M. Alam, "Automatic Target Tracking in FLIR Image Sequences Using Intensity Variation," *IEEE Transactions on Instrumentation and Measurement*, vol. 54, no. 5, pp. 1846–1852, 2005. 14

[9] D. H. Ballard, "Generalizing the Hough Transform to Detect Arbitrary Shapes," *Pattern Recognition*, vol. 13, no. 2, pp. 111–122, 1981. 44, 64

[10] Y. Bar-Shalom, "Tracking Methods in a Multitarget Environment," *IEEE Transactions on Automatic Control*, vol. 23, no. 4, pp. 618–626, 1978. 7

[11] Y. Bar-Shalom, F. Daum, and J. Huang, "The Probabilistic Data Association Filter," *IEEE Control Systems Magazine*, vol. 29, no. 6, pp. 82–100, 2009. 1, 8, 10

[12] Y. Bar-Shalom and T. Fortmann, *Tracking and Data Association.* Academic Press, 1988, vol. 179. 7, 43, 146

[13] Y. Bar-Shalom and E. Tse, "Tracking in a Cluttered Environment with Probabilistic Data Association," *Automatica*, vol. 11, no. 5, pp. 451–460, 1975. 8, 57, 117, 121, 164

[14] Y. Bar-Shalom, P. Willett, and X. Tian, *Tracking and Data Fusion: A Handbook of Algorithms.* YBS Publishing, 2011. 19, 32, 78, 94, 103, 110, 142, 158

[15] M. Betke, D. Hirsh, A. Bagchi, N. Hristov, N. Makris, and T. Kunz, "Tracking Large Variable Numbers of Objects in Clutter," in *IEEE Conference on Computer Vision and Pattern Recognition*, Jun. 2007. 1

[16] S. Bhatia and S. Chalup, "Segmenting Salient Objects in 3D Point Clouds of Indoor Scenes Using Geodesic Distances," *Journal of Signal and Information Processing*, vol. 4, no. 3B, pp. 102–108, 2013. 5

[17] S. Blackman, "Multiple Hypothesis Tracking for Multiple Target Tracking," *Aerospace and Electronic Systems Magazine, IEEE*, vol. 19, no. 1, pp. 5–18, 2004. 9, 10

[18] S. Blackman, R. Dempster, and R. Reed, "Demonstration of Multiple Hypothesis Tracking (MHT) Practical Real-Time Implementation Feasibility," in *Proceedings SPIE Confrence on Signal and Data Processing of Small Targets*, 2001, pp. 470–475. 10

[19] S. Blackman and R. Popoli, *Design and Analysis of Modern Tracking Sytstems*. Norwood, MA: Artech House, 1999. 7, 8, 9, 12, 19, 35, 36, 64, 78, 99, 116, 117, 118, 164

[20] W. Blanding, P. Willett, and Y. Bar-Shalom, "ML-PDA: Advances and a New Multitarget Approach," *EURASIP Journal on Advances in Signal Processing*, 2008. 9

[21] H. Blom, "An Efficient Filter for Abruptly Changing Systems," in *The 23rd IEEE Conference on Decision and Control*, 1984, pp. 656–658. 26, 89, 94, 95, 136

[22] J. Boškovic, J. Jackson, and R. Mehra, "Sensor and Tracker Requirements Development for Sense and Avoid Systems for Unmanned Aerial Vehicles," in *Proceedings of the Guidance, Navigation, and Control Conference*, 2013. 1

[23] R. Brown, *Introduction to Random Signal Analysis and Kalman Filtering*. John Wiley & Sons, Inc., 1983. 105

[24] B. Cannon, R. Leishman, T. McLain, J. Jackson, and J. Boškovic, "Non-Redundant Sensor Fault Detection for Autonomous Rotorcraft using an Improved Dynamic Model," in *Proceedings of the AIAA Guidance Navigation and Control Conference*, 2013, p. 4776. 145

[25] Z. Chao, H. Hua-sheng, B. Wei-min, and Z. Luo-ping, "Robust Recursive Estimation of Auto-Regressive Updating Model Parameters for Real-Time Flood Forecasting," *Journal of Hydrology*, vol. 349, no. 3-4, Feb. 2008. 43

[26] Y. Cheng, "Mean Shift, Mode Seeking, and Clustering," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 17, no. 8, pp. 790–799, 1995. 14

[27] N. Chenouard, I. Bloch, and J. Olivo-Marin, "Multiple Hypothesis Tracking in Microscopy Images," in *IEEE International Symposium on Biomedical Imaging: From Nano to Macro*, 2009, pp. 1346–1349. 1

[28] T. Chin, H. Wang, and D. Suter, "Robust Fitting of Multiple Structures: The Statistical Learning Approach," in *IEEE 12th International Conference on Computer Vision*, Sep. 2009, pp. 413–420. 5

[29] S. Cho, S. Huh, H. Choi, and D. Shim, "A Vision-Based Detection and Tracking of Airborne Obstacles in Cluttered Environment," *Journal of Intelligent & Robotic Systems*, vol. 69, no. 1-4, pp. 475–488, 2013. 1

[30] S. Choi, T. Kim, and W. Yu, "Performance Evaluation of RANSAC Family," in *Proceedings of the British Machine Vision Conference*. British Machine Vision Association, 2009. 4, 21

[31] D. Clark, A. Cemgil, P. Peeling, and S. Godsill, "Multi-Object Tracking of Sinusoidal Components in Audio with the Gaussian Mixture Probability Hypothesis Density Filter," in *IEEE Workshop on Applications of Signal Processing to Audio and Acoustics*, 2007, pp. 339–342. 1

[32] D. Clark, K. Panta, and B. Vo, "The GM-PHD Filter Multiple Target Tracker," in *9th International Conference on Information Fusion*, 2006. 119, 132

[33] S. Cong, L. Hong, and D. Wicker, "Markov-Chain Monte-Carlo Approach for Association Probability Evaluation," in *Proceedings of Control Theory and Applications*, 2004, pp. 185–193. 11

[34] G. Conte, M. Hempel, P. Rudol, D. Lundström, S. Duranti, M. Wzorek, and P. Doherty, "High Accuracy Ground Target Geo-Location using Autonomous Micro Aerial Vehicle Platforms," in *Proceedings of the AIAA Guidance, Navigation and Control Conference*, 2008. 58

[35] R. Cooperman, "Tactical Ballistic Missile Tracking using the Interacting Multiple Model Algorithm," in *5th International Conference on Information Fusion*, vol. 2, 2002, pp. 824–831. 1

[36] S. Coraluppi, D. Grimmett, and P. de Theije, "Benchmark Evaluation of Multistatic Trackers," in *9th International Conference on Information Fusion*, 2006. 28, 29

[37] I. Cox, "Downloads: Source Code: Multiple Hypothesis Tracking Code." [Online]. Available: http://mediafutures.cs.ucl.ac.uk/people/IngemarCox/downloads/ 116, 118

[38] ——, "A Review of Statistical Data Association Techniques for Motion Correspondence," *International Journal of Computer Vision*, vol. 10, no. 1, pp. 53–66, 1993. 7

[39] I. Cox and S. 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*, vol. 18, no. 2, pp. 138–150, 1996. 10, 116, 118

[40] I. Cox and M. Miller, "On Finding Ranked Assignments With Application to Multi-Target Tracking and Motion Correspondence," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 32, no. 1, pp. 486–489, 1995. 10, 116, 118

[41] D. Crouse, M. Guerriero, and P. Willett, "A Critical Look at the PMHT," *Journal of Advances in Information Fusion*, vol. 4, no. 2, pp. 93–116, 2009. 11

[42] D. Crouse, P. Willett, Y. Bar-Shalom, and L. Svensson, "Aspects of MMOSPA Estimation," in *50th IEEE Conference on Decision and Control and European Control Conference*, no. 1, 2011, pp. 6001–6006. 28

[43] F. Daum and J. Huang, "Curse of Dimensionality and Particle Filters," in *Proceedings of the IEEE Aerospace Conference*, vol. 4, 2003, pp. 1979–1993. 12

[44] G. Demos, R. Ribas, T. Broida, and S. Blackman, "Applications of MHT to Dim Moving Targets," in *Proceedings SPIE 1305, Signal and Data Processing of Small Targets*, 1990, pp. 297–309. 9

[45] A. Dempster, N. Laird, and D. Rubin, "Maximum Likelihood from Incomplete Data Via the EM Algorithm," *Journal of the Royal Statistical Society*, vol. 39, no. 1, pp. 1–38, 1977. 11

[46] A. Dick, P. Torr, and R. Cipolla, "Automatic 3D Modelling of Architecture," in *Procedings of the British Machine Vision Conference*, 2000. 5

[47] N. R. Draper, H. Smith, and E. Pownell, *Applied Regression Analysis*, 3rd ed. New York: Wiley, 1966. 44

[48] U. Dubin, "Probabilistic Data Association Filters (PDAF) - a tracking demo," 2011. [Online]. Available: http://www.mathworks.com/matlabcentral/fileexchange/34146-probabilistic-data-association-filters-pdaf-a-tracking-demo 57

[49] R. O. Duda and P. E. Hart, "Use of the Hough transformation to detect lines and curves in pictures," *Communications of the ACM*, vol. 15, no. 1, pp. 11–15, 1972. 44

[50] D. Dunham, R. Dempster, and S. Blackman, "Tracking Algorithm Speed Comparisons Between MHT and PMHT," in *Proceedings of the Fifth International Conference on Information Fusion*, vol. 2, 2002, pp. 846–851. 11

[51] O. Erdinc, P. Willett, and Y. Bar-Shalom, "Probability Hypothesis Density Filter for Multitarget Multisensor Tracking," in *7th International Conference on Information Fusion*, 2005, pp. 146–153. 14, 127

[52] O. Erdinc, P. Willett, and S. Coraluppi, "The Gaussian Mixture Cardinalized PHD Tracker on MSTWG and SEABAR'07 Datasets," in *11th International Conference on Information Fusion*, 2008, pp. 1791–1798. 1, 16

[53] M. Fischler and R. Bolles, "Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography," *Communications of the ACM*, vol. 24, no. 6, pp. 381–395, 1981. 3, 4, 21, 22, 23, 43, 77, 82, 101

[54] R. Fitzgerald, "Development of Practical PDA Logic for Multitarget Tracking by Microprocessor," in *American Control Conference*, 1986, pp. 889–898. 9, 10

[55] A. Fitzgibbon, M. Pilu, and R. Fisher, "Direct Least Squares Fitting of Ellipses," *Pattern Analysis and Machine Intelligence*, vol. 21, no. 5, pp. 476–480, 1996. 68, 73

[56] T. Fortmann, Y. Bar-Shalom, and M. Scheffé, "Multi-Target Tracking Using Joint Probabilistic Data Association," in *9th IEEE Conference on Decision and Control including the Symposium on Adaptive Processes*, vol. 19, 1980, pp. 807–812. 7, 9, 62, 117

[57] S. Foucher, G. Bénié, and J. Boucher, "Multiscale MAP Filtering of SAR Images." *IEEE Transactions on Image Processing*, vol. 10, no. 1, pp. 49–60, 2001. 64

[58] B. Fridling and O. Drummond, "Performance Evaluation Methods for Multiple Target Tracking Algorithms," in *Proceedings of SPIE 1481, Signal and Data Processing of Small Targets*, 1991, pp. 371–383. 28

[59] G. Gallego, C. Cuevas, R. Mohedano, and N. Garcia, "On the Mahalanobis Distance Classification Criterion for Multidimensional Normal Distributions," *IEEE Transactions on Signal Processing*, vol. 61, no. 17, pp. 4387–4396, Sep. 2013. 100, 104

[60] W. Gander, "Least Squares with a Quadratic Constraint," *Numerische Mathematik*, vol. 36, no. 3, pp. 291–307, 1981. 69

[61] D. Garmatyuk, K. Kauffman, J. Schuerger, and S. Spalding, "Wideband OFDM System for Radar and Communications," in *IEEE Radar Conference*, 2009. 64

[62] D. Garmatyuk, "Simulated Imaging Performance of UWB SAR Based on OFDM," in *IEEE International Conference on Ultra-Wideband*, 2006, pp. 237–242. 64

[63] A. Gelb, Ed., *Applied Optimal Estimation.* MIT press, 1974. 157

[64] R. Georgescu, P. Niedfeldt, S. Zhang, A. Speranzon, and O. Erdinc, "The CPHD and R-RANSAC Trackers Applied to the VIVID Dataset," in *Proceedings of SPIE 9092, Signal and Data Processing of Small Targets*, 2014, p. 90920E. 16, 158

[65] R. Georgescu and P. Willett, "Classification Aided Cardinalized Probability Hypothesis Density Filter," in *Proceedings of SPIE 8392, Signal Processing, Sensor Fusion, and Target Recognition XXI*, vol. 8392, 2012, p. 83920F. 158

[66] A. Gorji, R. Tharmarasa, and T. Kirubarajan, "Performance Measures for Multiple Target Tracking Problems," in *Proceedings of the 14th International Conference on Information Fusion*, 2011, pp. 1560–1567. 28

[67] R. Halí and J. Flusser, "Numerically Stable Direct Least Squares Fitting of Ellipses," in *International Conference in Central Europe on Computer Graphics*, 1998, pp. 125–132. 67, 69

[68] D. Hall and J. Llinas, Eds., *Handbook of Multisensor Data Fusion: Theory and Practice.* New York: CRC Press, 2001. 1, 77

[69] S. Haykin, *Adaptive Filter Theory*, 4th ed. Upper Saddle River, NJ: Prentice Hall, 2002. 43, 163

[70] G. Heredia, A. Ollero, M. Bejar, and R. Mahtani, "Sensor and Actuator Fault Detection in Small Autonomous Helicopters," *Mechatronics*, vol. 18, no. 2, pp. 90–99, 2008. 145

[71] R. Hess and A. Fern, "Discriminatively Trained Particle Filters for Complex Multi-Object Tracking," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2009, pp. 240–247. 1

[72] P. Hough, "Method and Means for Recognizing Complex Patterns," *US Patent 3,069,654*, 1962. 5, 44, 50, 64

[73] J. Huang, W. Hu, and L. Zhang, "Bayesian Group Tracking Method For Space Debris," in *5th European Conference on Space Debris*, 2013. 1

[74] T. Huang and S. Russell, "Object Identification in a Bayesian Context," in *15th International Conference on Artificial Intelligence*, 1997, pp. 1276–1283. 9

[75] P. J. Huber, *Robust Statistical Procedures*, 2nd ed. Philadelphia: SIAM, 1996. 44

[76] C. Hue, J. Le Cadre, and P. Pérez, "Sequential Monte Carlo Methods for Multiple Target Tracking and Data Fusion," *IEEE Transactions on Signal Processing*, vol. 50, no. 2, pp. 309–325, 2002. 12

[77] ——, "Tracking Multiple Objects with Particle Filtering," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 38, no. 3, pp. 791–812, 2002. 12

[78] IMSAR, "NanoSARC," 2013. [Online]. Available: http://www.imsar.com/uploads/files/46_NanoSAR_C_Data_Sheet.pdf 64, 74

[79] K. Jung, N. Kim, S. Lee, and J. Paik, "Dual-Layer Particle Filtering for Simultaneous Multiple Objects Detecting and Tracking," in *IEEE Conference on Acoustics, Speech and Signal Processing*, 2013, pp. 2307–2311. 12

[80] R. Kalman, "A New Approach to Linear Filtering and Prediction Problems," *Journal of Basic Engineering*, vol. 82, no. 1, pp. 35–45, 1960. 164

[81] B. Kalyan, K. Lee, S. Wijesoma, and N. Patrikalakis, "A Clutter Rejection Filter for Sonar Feature Based Navigation in Marine Environments," in *OCEANS*, 2011. 1

[82] Y. Kanazawa and H. Kawakami, "Detection of Planar Regions with Uncalibrated Stereo using Distributions of Feature Points," in *Procedings of the British Machine Vision Conference*, 2004, pp. 247–256. 4, 8

[83] T. Kangsheng, "Multi-Target Tracking Algorithm of Boost-Phase Ballistic missile defense," *Journal of Systems Engineering and Electronics*, vol. 24, no. 1, pp. 90–100, 2013. 1

[84] K. Kauffman, "Fast Target Tracking Technique for Synthetic Aperture Radars," Master's thesis, Miami University, 2009. 64

[85] ——, "Radar Based Navigation in Unknown Terrain," Ph.D. dissertation, Air Force Institute of Technology, 2012. 64

[86] K. Kauffman, D. Garmatyuk, and J. Morton, "Efficient Sparse Target Tracking Algorithm for Navigation with UWB-OFDM Radar Sensors," in *National Aerospace & Electronics Conference*, 2009, pp. 14–17. 64

[87] K. Kauffman, J. . Raquet, Y. Morton, and D. Garmatyuk, "Experimental Study of UWB-OFDM SAR for Indoor Navigation with INS Integration," in *25th International Technical Meeting of the Satellite Division of the Insitute of Navigation*, 2012, pp. 3847–3852. 64

[88] K. Kauffman, J. Raquet, Y. Morton, and D. Garmatyuk, "Simulation Study of UWB-OFDM SAR for Dead-Reckoning Navigation," in *International Technical Meeting of The Institute of Navigation*, 2010, pp. 153–160. 64

[89] ——, "Simulation Study of UWB-OFDM SAR for Navigation Using an Extended Kalman Filter," in *23rd International Technical Meeting of the Satellite Division of The Institute of Navigation*, 2010, pp. 2443–2451. 64, 65

[90] ——, "Enhanced Feature Detection and Tracking Algorithm for UWB-OFDM SAR Navigation," in *National Aerospace & Electronics Conference*, 2011, pp. 261–269. 64, 65, 161

[91] ——, "Simulation Study of UWB-OFDM SAR for Navigation with INS Integration," in *International Technical Meeting of The Institute of Navigation*, 2011, pp. 184–191. 64

[92] C. Kimme, D. Ballard, and J. Sklansky, "Finding Circles by an Array of Accumulators," *Communications of the ACM*, vol. 18, no. 2, pp. 2–4, 1975. 44

[93] T. Kirubarajan and Y. Bar-shalom, "Tracking Evasive Move-Stop-Move Targets with a GMTI Radar Using a VS-IMM Estimator," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 39, no. 3, pp. 1098–1103, 2003. 136

[94] G. Kladis, J. Economou, K. Knowles, J. Lauber, and T. Guerra, "Energy Conservation Based Fuzzy Tracking for Unmanned Aerial Vehicle Missions Under a priori Known Wind Information," *Engineering Applications of Artificial Intelligence*, vol. 24, no. 2, pp. 278–294, 2011. 65

[95] J. Kuchar and L. Yang, "A Review of Conflict Detection and Resolution Modeling Methods," *IEEE Transactions on Intelligent Transportation Systems*, vol. 1, no. 4, pp. 179–189, 2000. 1

[96] T. Kurien, A. Blitz, R. Washburn, and A. Willsky, "Optimal Maneuver Detection and Estimation in Multiobject Tracking," in *6th MIT/ONR Workshop on Command, Control, and Communication Systems*, vol. 1, 1983, pp. 164–171. 9, 10, 118

[97] J. Leonard, B. Moran, I. Cox, and M. Miller, "Underwater Sonar Data Fusion Using an Efficient Multiple Hypothesis Algorithm," in *IEEE International Conference on Robotics and Automation*, 1995, pp. 2995–3002. 116

[98] C. Leondes, J. Peller, and E. Stear, "Nonlinear Smoothing Theory," *IEEE Transactions on Systems Science and Cybernetics*, vol. 6, no. 1, pp. 63–71, 1970. 157

[99] X. Li and V. Jilkov, "Survey of Maneuvering Target Tracking. Part I: Dynamic Models," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 39, no. 4, pp. 1333–1364, 2003. 32

[100] B. Lucas and T. Kanade, "An Iterative Image Registration Technique with an Application to Stereo Vision," in *7th International Joint Conference on Artificial Intelligence*, 1981, pp. 674–679. 14

[101] R. Mahler, "A Theoretical Foundation for the Stein-Winter" Probability Hypothesis Density (PHD)" Multitarget Tracking Approach," Lockheed Martin, Tech. Rep., 2000. 13, 111

[102] ——, "Multitarget Bayes Filtering via First-Order Multitarget Moments," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 39, no. 4, pp. 1152–1178, 2003. 13

[103] ——, "A Survey of PHD Filter and CPHD Filter Implementations," in *Proceedings of SPIE 6567, Signal Processing, Sensor Fusion, and Target Recognition XVI*, 2007. 14, 111

[104] ——, "PHD Filters of Higher Order in Target Number," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 43, no. 4, pp. 1523–1543, 2007. 14, 127

[105] ——, *Statistical Multisource-Multitarget Information Fusion*. Norwood: Artech House, 2007. 12, 13, 112, 126

[106] R. Mehra and J. Peschon, "An Innovations Approach to Fault Detection and Diagnosis," *Automatica*, vol. 7, pp. 637–640, 1971. 145

[107] K. Mehrotra and P. Mahapatra, "A Jerk Model for Tracking Highly Maneuvering Targets," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 33, no. 4, pp. 1094 – 1105, 1997. 78

[108] M. Montemerlo, S. Thrun, D. Koller, and B. Wegbreit, "FastSLAM: A Factored Solution to the Simultaneous Localization and Mapping Problem," in *American Association for Artificial Intelligence*, 2002, pp. 593–598. 1, 159

[109] R. Montgomery and D. Price, "Management of Analytical Redundancy in Digital Flight Control Systems for Aircraft," in *Mechanics and Control of Flight Conference*, 1974. 146

[110] T. Moon and W. Stirling, *Mathematical Methods and Algorithms for Signal Processing*. Upper Saddle River, NJ: Prentice Hall, 2000. 16, 43, 47, 49, 164

[111] J. Mullane, B. Vo, M. Adams, and B. Vo, "A Random-Finite-Set Approach to Bayesian SLAM," *IEEE Transactions on Robotics*, vol. 27, no. 2, pp. 268–282, 2011. 1, 159

[112] J. Munkres, "Algorithms for the Assignment and Transportation Problems," *Journal of the Society for Industrial & Applied Mathmatics*, vol. 5, no. 1, pp. 32–38, 1957. 8, 29, 116

[113] K. Murty, "An Algorithm for Ranking all the Assignments in Order of Increasing Cost," *Operations Research*, vol. 16, no. 3, pp. 682–687, 1968. 10, 118

[114] W. Ng, J. Li, and S. Godsill, "Online Multisensor-Multitarget Detection and Tracking," in *IEEE Aerospace Conference*, 2006. 12

[115] P. Niedfeldt and R. Beard, "Recursive RANSAC: Multiple Signal Estimation with Outliers," in *9th IFAC Symposium on Nonlinear Control Systems*, 2013. 16, 45, 50

[116] ——, "Multiple Target Tracking using Recursive RANSAC," in *To Appear in the Proceedings of the 2014 American Control Conference*, 2014. 16, 78, 104, 109, 146

[117] P. Niedfeldt, E. Quist, and R. Beard, "Characterizing Range Progression of SAR Reflectors with Recursive RANSAC," in *To Appear in the IEEE Radar Conference*, 2014. 16, 45, 50

[118] H. Niknejad, A. Takeuchi, S. Mita, and D. McAllester, "On-Road Multivehicle Tracking using Deformable Object Model and Particle Filter With Improved Likelihood Estimation," *IEEE Transactions on Intelligent Transportation Systems*, vol. 13, no. 2, pp. 748–758, 2012. 1

[119] S. Oh, S. Russell, and S. Sastry, "Multi-Scan MCMCDA (MATLAB)." [Online]. Available: http://cpslab.snu.ac.kr/software/multi-scan-mcmcda 116, 120

[120] S. Oh, "Multiple Target Tracking for Surveillance," Master's thesis, University of California, Berkeley, 2003. 8

[121] S. Oh, S. Russell, and S. Sastry, "Markov Chain Monte Carlo Data Association for Multi-Target Tracking," School of Engineering, University of California, Merced, SoE TR 2008001, Tech. Rep., 2008. 11

[122] S. Oh, S. Russell, S. Sastry, A. This, and M. Carlo, "Markov Chain Monte Carlo Data Association for Multi-Target Tracking," *IEEE Transactions on Automatic Control*, vol. 54, no. 3, pp. 481–497, 2009. 8, 11, 12, 19, 116, 120

[123] P. O'Leary and P. Zsombor-Murray, "Direct and Specific Least-Square Fitting of Hyperbolæand Ellipses," *Journal of Electronic Imaging*, vol. 13, no. 3, pp. 492–503, 2004. 67

[124] U. Orguner, "JPDA Association Probability Calculator (MATLAB)." [Online]. Available: http://www.control.isy.liu.se/student/graduate/TargetTracking/JPDA_ProbCalc.m 118

[125] H. Pasula, S. Russell, M. Ostland, and Y. Ritov, "Tracking Many Objects with Many Sensors," in *International Joint Conference on Artificial Intelligence*, 1999, pp. 1160–1171. 11

[126] R. Philip, S. Ram, X. Gao, and J. Rodríguez, "A Comparison of Tracking Algorithm Performance for Objects in Wide Area Imagery," in *IEEE Southwest Symposium on Image Analysis and Interpretation*, 2014, pp. 109–112. 14

[127] R. Pitre, V. Jilkov, and X. Li, "A Comparative Study of Multiple-Model Algorithms for Maneuvering Target Tracking," in *Proceedings of SPIE 5809, Signal Processing, Sensor Fusion, and Target Recognition XIV*, 2005, p. 549. 3, 94, 95

[128] E. Pollard, B. Pannetier, and M. Rombaut, "Convoy Detection Processing by using the Hybrid Algorithm (GMCPHD/VS-IMMC-MHT) and Dynamic Bayesian Networks," in *12th International Conference on Information Fusion*, 2009, pp. 907–914. 1

[129] G. Pulford, "Taxonomy of Multiple Target Tracking Methods," in *IEE Radar, Sonar and Navigation*, 2005, pp. 291–304. 7, 14

[130] H. Quevedo, S. Blackman, T. Nichols, R. Dempster, and R. Wenski, "Reducing MHT Computational Requirements through use of Cheap JPDA Methods," in *Proceedings of SPIE 4473, Signal and Data Processing of Small Targets*, 2001, p. 246. 10, 11

[131] E. Quist and R. Beard, "Radar Odometry on Small Unmanned Aircraft," in *AIAA Guidance, Navigation and Control Conference*, 2013, p. 4698. 64, 65, 66

[132] D. Reid, "An Algorithm for Tracking Multiple Targets," *IEEE Transactions on Automatic Control*, vol. 24, no. 6, pp. 843–854, 1979. 7, 9, 32

[133] M. Richards, *Fundamentals of Radar Signal Processing.* Tata McGraw-Hill Education, 2005. 64

[134] B. Ristic, J. Sherra, and A. García-Fernández, "Performance Evaluation of Random Set Based Pedestrian Tracking Algorithms," in *8th IEEE International Conference on Intelligent Sensors, Sensor Networks and Information Processing*, 2012. 1

[135] B. Ristic, B. Vo, D. Clark, and B. Vo, "A Metric for Performance Evaluation of Multi-Target Tracking Algorithms," *IEEE Transactions on Signal Processing*, vol. 59, no. 7, pp. 3452–3457, 2011. 28

[136] J. Roecker, "A Class of Near Optimal JPDA Algorithms," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 30, no. 2, pp. 504–510, 1994. 9

[137] J. Roecker and G. Phillis, "Suboptimal Joint Probabilistic Data Association," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 29, no. 2, pp. 510–517, 1993. 9

[138] P. J. Rousseeuw and A. M. Leroy, *Robust Regression and Outlier Detection.* New York: John Wiley & Sons, Inc., 1987. 44

[139] S. Särkkä, A. Vehtari, and J. Lampinen, "Rao-Blackwellized Monte Carlo Data Association for Multiple Target Tracking," in *7th International Conference on Information Fusion*, 2004, pp. 583–590. 12

[140] R. Schnabel, R. Wahl, and R. Klein, "Efficient RANSAC for Point-Cloud Shape Detection," *Computer Graphics Forum*, vol. 26, no. 2, pp. 214–226, 2007. 3, 21

[141] D. Schuhmacher, B. Vo, and B. Vo, "A Consistent Metric for Performance Evaluation of Multi-Object Filters," *IEEE Transactions on Signal Processing*, vol. 56, no. 8, pp. 3447–3457, Aug. 2008. 28

[142] D. Schulz, W. Burgard, D. Fox, and A. Cremers, "Tracking Multiple Moving Targets with a Mobile Robot using Particle Filters and Statistical Data Association," in *IEEE International Conference on Robotics and Automation*, 2001, pp. 1665–1670. 12

[143] J. Sherman and W. J. Morrison, "Adjustment of an Inverse Matrix Corresponding to a Change in One Element of a Given Matrix," *The Annals of Mathematical Statistics*, vol. 21, no. 1, pp. 124–127, 1950. 43, 49

[144] H. Sidenbladh, "Multi-Target Particle Filtering for the Probability Hypothesis Density," in *6th International Conference on Information Fusion*, 2003, pp. 800–806. 13

[145] B. Sinopoli, L. Schenato, M. Franceschetti, K. Poolla, M. Jordan, and S. Sastry, "Kalman Filtering with Intermittent Observations," *IEEE Transactions on Automatic Control*, vol. 49, no. 9, pp. 1453–1464, 2004. 104, 105, 106

[146] R. Sittler, "An Optimal Data Association Problem in Surveillance Theory," *IEEE Transactions on Military Electronics*, vol. 8, no. 2, pp. 125–139, 1964. 1, 77

[147] S. Spehn, "Noise Adaptation and Correlated Maneuver Gating of an Extended Kalman Filter," Master's thesis, Naval Postgraduate School, 1990. 145

[148] D. Stowell, S. Muševič, J. Bonada, and M. Plumbley, "Improved Multiple Birdsong Tracking with Distribution Derivative Method and Markov Renewal Process Clustering," in *IEEE International Conference on Acoustics, Speech and Signal Processing*, 2013, pp. 468–472. 1

[149] P. Strandmark and I. Gu, "Joint Random Sample Consensus and Multiple Motion Models for Robust Video Tracking," in *Image Analysis*, A. Salberg, J. Hardeberg, and R. Jenssen, Eds. Springer, 2009, vol. 5575, pp. 450–459. 6

[150] R. Streit and T. Luginbuhl, "A Probabilistic Multi-Hypothesis Tracking Algorithm Without Enumeration and Pruning," in *6th Joint Service Data Fusion Symposium*, 1993, pp. 1015–1024. 10

[151] ——, "Probabilistic Multi-Hypothesis Tracking," Naval Undersea Warfare Center, Newport, RI, Tech. Rep. February, 1995. 10

[152] C. Taylor, "Improved Fusion of Visual Measurements Through Explicit Modeling of Outliers," in *IEEE/ION Position Location and Navigation Symposium*, 2012, pp. 512–517. 45, 145

[153] G. Thomaidis, L. Spinoulas, P. Lytrivis, M. Ahrholdt, G. Grubb, and A. Amditis, "Multiple Hypothesis Tracking for Automated Vehicle Perception," in *IEEE Intelligent Vehicles Symposium*, 2010, pp. 1122–1127. 1

[154] R. Toldo and A. Fusiello, "Robust Multiple Structures Estimation with J-Linkage," in *Proceedings of the European Conference on Computer Vision*, 2008, pp. 537–547. 5

[155] C. Tomasi and T. Kanade, "Detection and Tracking of Point Features," Carnegie Mellon University, Tech. Rep. April, 1991. 14

[156] B. Tordoff and D. Murray, "Guided Sampling and Consensus for Motion Estimation," in *Computer Vision—ECCV*, A. Heyden, Ed. Springer, 2002, vol. 2350, pp. 82–96. 4, 23, 101

[157] P. Torr and C. Davidson, "IMPSAC: Synthesis of Importance Sampling and Random Sample Consensus," *IEEE Transactions on Patern Analysis and Machine Intelligence*, vol. 25, no. 3, pp. 354–364, 2003. 4

[158] P. Torr and D. Murray, "The Development and Comparison of Robust Methods for Estimating the Fundamental Matrix," *International Journal of Computer Vision*, vol. 24, no. 3, pp. 271–300, 1997. 3, 45

[159] N. Tsokas and K. Kyriakopoulos, "Multi-Robot Multiple Hypothesis Tracking for Pedestrian Tracking," *Autonomous Robots*, vol. 32, no. 1, pp. 63–79, 2011. 1

[160] F. Ulaby, F. Kouyate, B. Brisco, and T. Williams, "Textural Information in SAR Images," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 24, no. 2, pp. 235–245, 1986. 64

[161] A. Vedaldi, H. Jin, P. Favaro, and S. Soatto, "KALMANSAC: Robust Filtering by Consensus," in *10th IEEE International Conference on Computer Vision*, 2005, pp. 633–640. 6, 21, 146

[162] H. Veeraraghavan and N. Papanikolopoulos, "Combining Multiple Tracking Modalities for Vehicle Tracking at Traffic Intersections," in *IEEE International Conference on Robotics and Automation*, vol. 3, 2004, pp. 2303–2308. 1

[163] J. Vermaak, S. Godsill, and P. Pérez, "Monte Carlo Filtering for Multi-Target Tracking and Data Association," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 41, no. 1, pp. 309–332, 2005. 12

[164] E. Vincent and R. Laganière, "Detecting Planar Homographies in an Image Pair," in *2nd International Symposium on Image and Signal Processing and Analysis*, 2001, pp. 182–187. 4, 8

[165] B. Vo and W. Ma, "A Closed-Form Solution for the Probability Hypothesis Density Filter," in *7th International Conference on Information Fusion*, vol. 2, 2005. 13

[166] ——, "The Gaussian Mixture Probability Hypothesis Density Filter," *IEEE Transactions on Signal Processing*, vol. 54, no. 11, pp. 4091–4104, 2006. 13, 18, 19, 26, 111, 112, 116, 119, 120, 132

[167] B. Vo, S. Singh, and A. Doucet, "Sequential Monte Carlo Implementation of the PHD Filter for Multi-Target Tracking," in *6th International Conference of Information Fusion*, 2003, pp. 792–799. 13

[168] B. Vo, B. Vo, and A. Cantoni, "Analytic Implementations of the Cardinalized Probability Hypothesis Density Filter," *IEEE Transactions on Signal Processing*, vol. 55, no. 7, pp. 3553–3567, Jul. 2007. 16

[169] X. Wang, B. Moran, and S. Challa, "Efficient Feature Aided Multi-Object Tracking in Video Surveillance," in *3rd International Congress on Image and Signal Processing*, 2010, pp. 114–118. 158

[170] P. Willett and S. Coraluppi, "MLPDA and MLPMHT Aplied to Some MSTWG Data," in *9th International Conference on Information Fusion*, 2006, pp. 1–8. 11

[171] T. Wood, C. Yates, D. Wilkinson, and G. Rosser, "Simplified Multitarget Tracking Using the PHD Filter for Microscopic Video Data," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 22, no. 5, pp. 702–713, 2012. 1

[172] J. Xing, H. Ai, L. Liu, and S. Lao, "Multiple Player Tracking in Sports Video: A Dual-Mode Two-Way Bayesian Inference Approach with Progressive Observation Modeling." *IEEE Transactions on Image Processing*, vol. 20, no. 6, pp. 1652–67, 2011. 1

[173] L. Xu and E. Oja, "Randomized Hough Transform (RHT): Basic Mechanisms, Algorithms, and Computational Complexities," *CVGIP: Image understanding*, vol. 57, no. 2, pp. 131–154, 1993. 5

[174] L. Xu, E. Oja, and P. Kultanen, "A New Curve Detection Method: Randomized Hough Transform (RHT)," *Pattern Recognition Letters*, vol. 11, no. 5, pp. 331–338, 1990. 5

[175] A. Yilmaz, O. Javed, and M. Shah, "Object Tracking: A Survey," *ACM Computing Surveys*, vol. 38, no. 4, 2006. 7, 14

[176] T. Zajic, R. Ravichandran, R. Mahler, R. Mehra, and M. Noviskey, "Joint Tracking and Identification with Robustness against Unmodeled Targets," in *Proceedings of SPIE 5096, Signal Processing, Sensor Fusion, and Target Recognition*, vol. 5096, 2003, pp. 279–290. 13

[177] E. Zaugg, D. Hudson, and D. Long, "The BYU SAR: A Small, Student-Built SAR for UAV Operation," in *IEEE International Conference on Geoscience and Remote Sensing Symposium*, 2006, pp. 411–414. 64

[178] W. Zhang and J. Ksecká, "Nonparametric Estimation of Multiple Structures with Outliers," in *Dynamical Vision*, Y. Vidal, René and Heyden, Anders and Ma, Ed. Springer, 2007, vol. 4358, pp. 60–74. 5

[179] M. Zuliani, C. Kenney, and B. Manjunath, "The multiRANSAC Algorithm and its Application to Detect Planar Homographies," in *IEEE International Conference on Image Processing*, vol. III, 2005, pp. 153–156. 5, 8

[180] M. Zuliani, "RANSAC for Dummies," 2011. [Online]. Available: http://vision.ece.ucsb.edu/~zuliani/Research/RANSAC/docs/RANSAC4Dummies.pdf 28, 45