SANDIA REPORT

SAND2015-0209
Unlimited Release
Printed January 2015

Large Scale Tracking Algorithms

Ross L. Hansen, Joshua A. Love, David K. Melgaard,
David B. Karelitz, Todd A. Pitts, Joshua D. Zollweg,
Dylan Z. Anderson, Prabal Nandy, Gary L. Whitlow,
Daniel A. Bender and Raymond H. Byrne

Prepared by
Sandia National Laboratories
Albuquerque, New Mexico 87185 and Livermore, California 94550

Sandia National Laboratories is a multi-program laboratory managed and operated by Sandia Corporation,
a wholly owned subsidiary of Lockheed Martin Corporation, for the U.S. Department of Energy's
National Nuclear Security Administration under contract DE-AC04-94AL85000.

Approved for public release; further dissemination unlimited.

Sandia National Laboratories

Issued by Sandia National Laboratories, operated for the United States Department of Energy by Sandia Corporation.

NOTICE: This report was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government, nor any agency thereof, nor any of their employees, nor any of their contractors, subcontractors, or their employees, make any warranty, express or implied, or assume any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represent that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise, does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government, any agency thereof, or any of their contractors or subcontractors. The views and opinions expressed herein do not necessarily state or reflect those of the United States Government, any agency thereof, or any of their contractors.

Printed in the United States of America. This report has been reproduced directly from the best available copy.

Available to DOE and DOE contractors from
U.S. Department of Energy
Office of Scientific and Technical Information
P.O. Box 62
Oak Ridge, TN 37831

Telephone: (865) 576-8401
Facsimile: (865) 576-5728
E-Mail: reports@adonis.osti.gov
Online ordering: http://www.osti.gov/bridge

Available to the public from
U.S. Department of Commerce
National Technical Information Service
5285 Port Royal Rd
Springfield, VA 22161

Telephone: (800) 553-6847
Facsimile: (703) 605-6900
E-Mail: orders@ntis.fedworld.gov
Online ordering: http://www.ntis.gov/help/ordermethods.asp?loc=7-4-0#online

SAND2015-0209
Unlimited Release
Printed January 2015

Large Scale Tracking Algorithms

Ross L. Hansen, Joshua A. Love, David K. Melgaard, David B. Karelitz, Todd A. Pitts, Joshua D. Zollweg, Dylan Z. Anderson, Prabal Nandy, Gary L. Whitlow, Daniel A. Bender and Raymond H. Byrne

Sandia National Laboratories
MS 0576, P.O. Box 5800
Albuquerque, NM 87185-0576

Abstract

Low signal-to-noise data processing algorithms for improved detection, tracking, discrimination and situational threat assessment are a key research challenge. As sensor technologies progress, the number of pixels will increase significantly. This will result in increased resolution, which could improve object discrimination, but unfortunately, will also result in a significant increase in the number of potential targets to track. Many tracking techniques, like multi-hypothesis trackers, suffer from a combinatorial explosion as the number of potential targets increase. As the resolution increases, the phenomenology applied towards detection algorithms also changes. For low resolution sensors, "blob" tracking is the norm. For higher resolution data, additional information may be employed in the detection and classification steps. The most challenging scenarios are those where the targets cannot be fully resolved, yet must be tracked and distinguished for neighboring closely spaced objects. Tracking vehicles in an urban environment is an example of such a challenging scenario. This report evaluates several potential tracking algorithms for large-scale tracking in an urban environment.

Acknowledgments

This research was funded by the Laboratory Directed Research and Development (LDRD) program at Sandia National Laboratories. Sandia National Laboratories is a multi-program laboratory managed and operated by Sandia Corporation, a wholly owned subsidiary of Lockheed Martin Corporation, for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-AC04-94AL85000.

© 2012 The MathWorks, Inc. MATLAB® and Simulink® are registered trademarks of The MathWorks, Inc. See www.mathworks.com/trademarks for a list of additional trademarks. Other product or brand names may be trademarks or registered trademarks of their respective holders.

Contents

Abstract 3

Acknowledgments 4

Table of Contents 5

List of Figures 7

List of Tables 9

1 Introduction 11

2 Tracking Evaluation Data Sets 12
- 2.1 Simulated Urban Traffic . . . 12
- 2.2 Vehicle Traffic Data from Albuquerque, New Mexico . . . 13
- 2.3 AFRL Wright Patterson AFB Flyover Dataset . . . 16
  - 2.3.1 Region of Interest 1 . . . 16
  - 2.3.2 Region of Interest 2 . . . 16
  - 2.3.3 Region of Interest 3 . . . 17

3 Evaluation Metrics 17

4 Probabilistic Multi Hypothesis Tracking (PMHT) 18

5 Random Sample Consensus (RANSAC) 21
- 5.1 Introduction . . . 21
- 5.2 Algorithm . . . 21
  - 5.2.1 RANSAC . . . 21
  - 5.2.2 s-RANSAC . . . 23
  - 5.2.3 ms-RANSAC . . . 25
- 5.3 Implementation . . . 27
- 5.4 Results . . . 28
  - 5.4.1 AFRL ROI 1 . . . 28
  - 5.4.2 AFRL ROI 2 . . . 29
  - 5.4.3 AFRL ROI 3 . . . 29
  - 5.4.4 Crest . . . 29
  - 5.4.5 Socorro . . . 30
  - 5.4.6 Design Parameter Sensitivity . . . 30
- 5.5 Future Work . . . 31
  - 5.5.1 Additional Analysis . . . 31
  - 5.5.2 Improved Performance . . . 32
- 5.6 Conclusions . . . 33

6 Markov Chain Monte Carlo Data Association (MCMCDA) 33
- 6.1 MCMC Methods . . . 34
- 6.2 Application of MCMC to Data Association . . . 34
- 6.3 Modeling the Posterior . . . 35
  - 6.3.1 Development of the Prior . . . 35

6.3.2 Development of the Likelihood . . . 36
6.4 MCMCDA Algorithm . . . 36
6.5 Results . . . 36

7 Tracklet Inference from Factor Graphs 39
7.1 Algorithm Overview . . . 39
7.2 Bayesian Network and Factor Graph Formulation . . . 39
7.2.1 Motion-only Approximation . . . 40
7.2.2 Virtual Detections . . . 41
7.2.3 Motion Parents . . . 41
7.2.4 Converting Bayesian Networks to Factor Graphs . . . 41
7.3 Most Probably Explanation and Inference on Factor Graphs . . . 42
7.4 Implementation of Bayesian Network, Factor Graph, and MPE . . . 43
7.5 Tracklet Inference from MPE Result . . . 43
7.6 Forming Tracks from Tracklets . . . 43
7.7 Results . . . 44
7.8 Future Work and Algorithm Benefits and Drawbacks . . . 46

8 Proximity Tracking Algorithm 48
8.1 Basic Algorithm . . . 48
8.2 Data Sets Results . . . 49
8.3 Comments . . . 53

9 Algorithm Comparison 55

10 Imaging Resolution Study Using DIRSIG 56
10.1 Generating Synthetic Imagery with DIRSIG . . . 56
10.2 Scene Simulation . . . 56
10.2.1 Geometry Setup . . . 56
10.2.2 Peterbilt 359 Semi Truck . . . 57
10.2.3 Audi A3 Car . . . 57
10.3 Resolution and Pixel Phasing . . . 57
10.4 Results . . . 61
10.4.1 Peterbilt 359 Semi-Truck . . . 61
10.4.2 Audi A3 Car . . . 70
10.5 General Trends . . . 70

11 Summary and Conclusions 83

References 84

List of Figures

1 Detection and tracking system block diagram [1]. 11
2 Intensity as a function of heading for simulated night traffic data. 13
3 Map of Socorro, New Mexico. 13
4 Sample traffic data from Albuquerque, New Mexico. 15
5 Sample image from AFRL WPAFB dataset and plot of all tracks. 16
6 Locations of Regions of Interest in first 70 frames of track data. 17
7 Sample image and all tracks in Region of Interest 1. 17
8 Sample image and all tracks in Region of Interest 2. 18
9 Sample image and all tracks in Region of Interest 3. 18
10 PMHT Bayesian Inference Network (BIN) [2]. 20
11 PMHT example results: proper convergence (left), no convergence (center), incorrect convergence (right). 20
12 Batch of detections input to RANSAC. 22
13 Random sample with only 2 inliers. 23
14 Random sample with 10 inliers. 23
15 Batch of detections input to s-RANSAC. 24
16 First acceptable consensus set. 24
17 Remaining Detections. 24
18 Second acceptable consensus set. 25
19 Remaining Detections. 25
20 Batch of detections input to ms-RANSAC. 26
21 Subbatch formed from scans 1-10. 27
22 Consensus sets for scans 1-10. 27
23 Subbatch formed for scans 4-13. 27
24 Consensus sets for scans 4-13. 27
25 Consensus sets for both subbatches. 28
26 Merged Tracks for scans 1-13. 28
27 ms-RANSAC Results for AFRL ROI 1. 29
28 ms-RANSAC Results for AFRL ROI 2. 30
29 ms-RANSAC Results for AFRL ROI 3. 31
30 ms-RANSAC Results for Crest. 32
31 ms-RANSAC Results for Socorro Frames 100 through 199. 33
32 Examples of the five valid MCMC moves [3]. 35
33 Evolution of MOTA and MT performance as MCMC sampling progresses in the AFRL WPAFB ROI 2 dataset. 37
34 Evolution of MOTA and MT performance as MCMC sampling progresses in the Crest dataset. 38
35 Construction of Bayesian networks in a sliding window. 40
36 Example Bayesian network. 42
37 Result of MPE query on Factor Graph, resulting in a set of detections consistent with the root node. 42
38 Sample tracking of two vehicles in AFRL WPAFB ROI 2 dataset. The faster vehicle, Truth 1, is covered by two track segments while the slower vehicle, Truth 2, is completely tracked by a single segment. 45
39 A histogram of the number of vehicles that are broken into multiple track segments in the ROI 2 dataset. 46

40 Crest data true tracks. 49
41 Broken due to gaps in the detections. 51
42 True tracks for Socorro simulated data. 52
43 Socorro tracks that were mostly lost 52
44 Socorro area of interest tracks. 52
45 WPAFB ROI1 true tracks. 53
46 WPAFB ROI2 true tracks. 53
47 WPAFB ROI3 true tracks. 53
48 Simulated semi-truck target imagery setup 58
49 Simulated semi-truck target dimensions 58
50 Simulated car target imagery setup 59
51 Simulated car target dimensions 59
52 Simulation imagery background radiance 60
53 Pixel phasing illustration 61
54 Mean peak radiance above background for semi-truck target, sun at 90° elevation . . 63
55 Semi-truck mean peak radiance above background for nadir illumination, low elevation platform 64
56 Normalized standard deviation of peak radiance above background for semi-truck target, sun at 90° elevation 65
57 Low semi-truck standard deviation from nadir illumination, nadir platform 66
58 High semi-truck standard deviation from nadir illumination, nadir platform 67
59 Low semi-truck standard deviation from nadir illumination, low elevation platform . 68
60 High semi-truck standard deviation from nadir illumination, low elevation platform . 69
61 Mean peak radiance above background for semi-truck target, sun at 15° elevation . . 71
62 Normalized standard deviation of peak radiance above background for semi-truck target, sun at 15° elevation 72
63 Mean peak radiance above background for semi-truck target, sun at 165° elevation . 73
64 Normalized standard deviation of peak radiance above background for semi-truck target, sun at 165° elevation 74
65 Sun reflections on car target windshield 75
66 Mean peak radiance above background for car target, sun at 90° elevation 76
67 Normalized standard deviation of peak radiance above background for car target, sun at 90° elevation 77
68 Mean peak radiance above background for car target, sun at 15° elevation 78
69 Normalized standard deviation of peak radiance above background for car target, sun at 15° elevation 79
70 Mean peak radiance above background for car target, sun at 165° elevation 80
71 Normalized standard deviation of peak radiance above background for car target, sun at 165° elevation 81
72 Comparison of simulated target means 82
73 Comparison of simulated target standard deviations 82

List of Tables

1 Summary of vehicle parameters. 14
2 Distribution of world car colors [4]. 14
3 Summary of tracking metrics. 19
4 ms-RANSAC results. 28
5 ms-RANSAC results under smaller error tolerances. 31
6 Summary of MCMCDA results. 38
7 Conditional probability table between detections. 40
8 Conditional probability table for observation nodes. 40
9 Summary results for the factor graph inference algorithm. 44
10 Summary results for the factor graph inference algorithm with a more restrictive motion model. 46
11 Summary results for the factor graph inference algorithm for long tracks only. 47
12 Proximity algorithm parameters for Crest data (pixel units). 50
13 Proximity algorithm parameters for Socorro data (pixel units). 51
14 Proximity algorithm parameters for Wright Patterson Air Force Base data (pixel units). 53
15 Summary of proximity tracker results. 54
16 Comparison of Algorithm Results 55
17 Sun positions simulated in DIRSIG 56
18 Imaging platform positions simulated in DIRSIG 57
19 Spectral bands simulated in DIRSIG 57
20 Resolutions of simulated target imagery 60

This page intentionally left blank.

1 Introduction

The technical question that we are trying to answer is "what is the best approach for tracking large numbers of unresolved low signal-to-noise ratio (SNR) targets" and "what system parameters have the greatest impact on system performance?" Recent advances in computing have enabled the application of multi-processor GPU's (graphics processor units) for processing large blocks of data in parallel. An integral part of the research was an optimization of parameters that drive system performance and design when the goal is to track and classify large numbers of targets in unresolved image data.

Current approaches have not addressed this problem for several reasons. First, the application of GPU technology to tracking applications is relatively new (e.g. the first general purpose GPU's became available in 2006). Second, most tracking applications operate on either point source data or higher resolution data but not on resolutions in between these bounds. For this effort, we will focus on the intersection of these traditional approaches. This makes the research unique, and also opens up a wide range of potential strategies.

The task of tracking large numbers of vehicles in an urban environment using image data was employed to focus the large scale tracking algorithm research. The basic elements of a multi-target tracking system are illustrated in Figure 1. For an imaging sensor, the first processing step usually involves some sort of background estimation and removal, followed by a detection algorithm. Simple thresholding is shown in Figure 1, but a variety of algorithms may be applied [5]. The next step is typically observation-to-track association: does the detection belong to an existing track or should a new track be created. This is followed by a track maintenance step (initiation, confirmation, or deletion). For each track, some sort of prediction (typically a Kalman Filter) is often employed to estimate the approximate location of the next detection for gating. This gating is then used to reduce the number of possible associations after the next observation. This paper focuses on the tracking algorithm (which is presented with a list of detections) and the phenomenology of vehicles in an urban environment. Detection algorithms are not addressed.

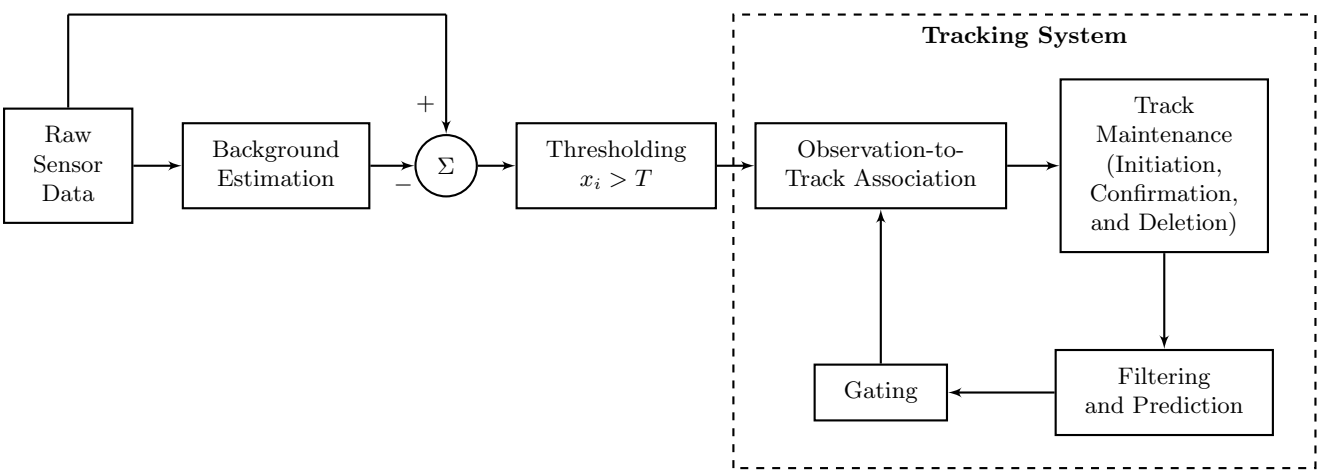

Figure 1: Detection and tracking system block diagram [1].

Previous research on tracking algorithms is extensive [6]. The design of a road-constrained particle filter for vehicle tracking is presented in [7, 8]. An approach for tracking vehicles in an urban environment using airborne image sequences is covered in [9]. They employ a particle filter to better handle the nonlinear dynamics of the target vehicles. In addition, information

about color and shape was incorporated to improve tracking reliability. A multi-model particle probability hypothesis density filter (PPHDF) is applied to the urban tracking problem in [10], which incorporates radar data rather than image data. The two motion models utilized are a nearly constant velocity (NCV) model as well as the coordinated turn (CT) model. An excellent general overview of tracking algorithms can be found in [6, 1, 11].

This report is organized as follows: Section 2 provides a description of the data sets employed to compare the performance of the tracking algorithms studied. Section 3 discusses the performance metrics to evaluate tracking algorithms. Section 4 provides results for the probabilistic multi-hypothesis tracker (PMHT). Section 5 reviews results for a Random Sample Consensus (RANSAC) based tracking algorithm. A Markov chain Monte Carlo data association is covered in Section 6. Tracklet inference from factor graphs is discussed in Section 7. A proximity tracker is described in Section 8. The relative performance of each tracking algorithm is discussed in Section 9. The phenomenology of vehicle targets is reviewed in Section 10. A summary appears in Section 11.

2 Tracking Evaluation Data Sets

In order to assess the performance of various tracking algorithms, vehicle tracks with "truth" data are required. Two common options are to manually verify detections and tracks in image data or to generate simulated data. The three data sets used in this research employed both approaches. For the first data set, simulated traffic data was generated using an open source simulator. The second and third data set were generated from image data and manual processing. A brief description of each of the data sets follows.

2.1 Simulated Urban Traffic

In order to create simulated urban vehicle tracks, we utilized the SUMO (Simulated Urban Mobility) open source traffic simulator [12]. A map of Socorro, NM, was downloaded from the OpenStreetMap project [13] and processed with the SUMO **netconvert** program to create a compatible netlist. The resulting netlist is shown in Figure 3. The **randomTrips.py** Python script was then employed to create random trips. The output of this script was modified with a MATLAB script to include additional trips (e.g. traffic on the interstate and some additional trips). Then, the SUMO **duarouter** was used to generate the xml file containing the vehicle routes. A MATLAB script then modified the route file to include a variety of vehicle types. The vehicle types are summarized in Table 1. Five percent of the vehicles are trucks, while the remaining are passenger vehicles with different characteristics. The vehicle type was randomly selected from the values in Table 1. The automobiles were also assigned a color drawn from the distribution of world automobile colors. All trucks were assumed to be grey. The distribution of car colors is summarized in Table 2. To allow for simulation of headlights seen in night data, an intensity value was assigned based on the heading of travel. The intensity as a function of heading is shown in Figure 2.

In order to simulate occlusions and various probabilities of detection, each vehicle was assigned a probability of detection between 0.5 and 1. Then, detections were sampled from the truth data using this probability of detection. Vehicle detections closer than 4 meters were combined into a single detection using a clustering algorithm (distance was the criterion for forming the clusters). Additive gaussian noise was added to the detection position ($\sigma = 0.1$ meter). Finally, false noise detections, with a density of 10 false detections/frame, were added to the detection list.

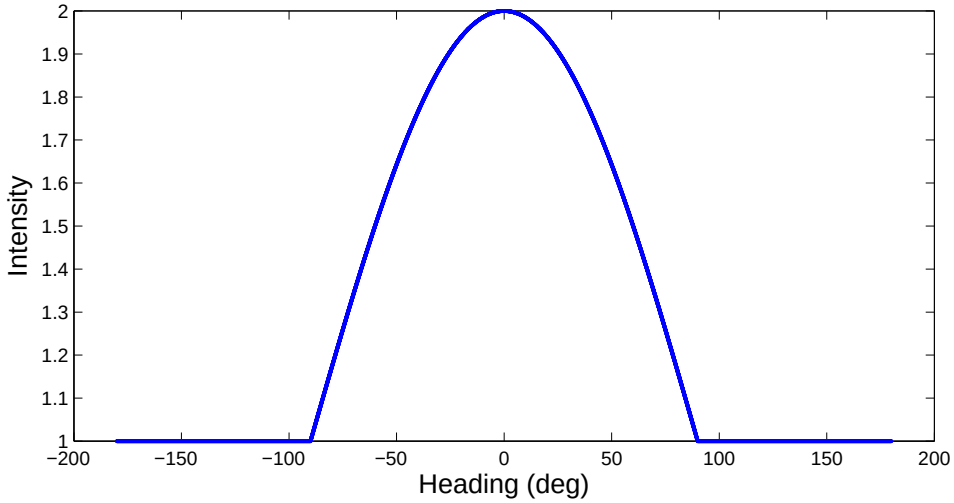

Figure 2: Intensity as a function of heading for simulated night traffic data.

Figure 3: Map of Socorro, New Mexico.

2.2 Vehicle Traffic Data from Albuquerque, New Mexico

Several low resolution data sets were collected of Albuquerque, NM, night traffic by placing a sensor on Sandia Peak, a 10,378 foot mountain adjacent to Albuquerque. A Red Mysterium X camera with an 18 to 85 mm adjustable lens mounted on a tripod was used to gather traffic data at 24 frames per second, with 3Kx3K resolution. The data were collected at .45, .9 and 1.8 meters/pixel in the foreground (e.g. bottom third of the frame). Frame to frame registration was applied to mitigate the effects of jitter. A typical processed frame is shown in Figure 4. Detections were identified using a simple thresholding scheme after background estimation and removal. Truth data was manually derived for a small section of the image in the foreground.

Table 1: Summary of vehicle parameters.

vehicle id	accel. (m/s^2)	decel. (m/s^2)	length (m)	minGap (m)	maxSpeed (m/s)	sigma (m/s)
CarA	3.0	6.0	5.0	2.5	40.0	0.5
CarB	2.0	6.0	7.5	2.5	33.0	0.5
CarC	1.0	5.0	5.0	2.5	33.0	0.5
CarD	1.0	5.0	7.5	2.5	25.0	0.5
CarE	3.0	6.0	5.0	2.5	40.0	0.5
CarF	2.0	6.0	7.5	2.5	33.0	0.5
CarG	1.0	5.0	5.0	2.5	33.0	0.5
CarH	1.0	5.0	7.5	2.5	25.0	0.5
CarI	1.0	5.0	7.5	2.5	40.0	0.5
CarJ	3.0	6.0	5.0	2.5	40.0	0.5
CarK	2.0	6.0	7.5	2.5	33.0	0.5
CarL	1.0	5.0	5.0	2.5	33.0	0.5
CarM	1.0	5.0	7.5	2.5	25.0	0.5
CarN	3.0	6.0	5.0	2.5	40.0	0.5
CarO	2.0	6.0	7.5	2.5	33.0	0.5
CarP	1.0	5.0	5.0	2.5	33.0	0.5
CarQ	1.0	5.0	7.5	2.5	25.0	0.5
CarR	1.0	5.0	7.5	2.5	40.0	0.5
CarS	1.0	5.0	7.5	2.5	40.0	0.5
TruckA	1.0	2.0	15	7.0	30.0	0.5

Table 2: Distribution of world car colors [4].

color	Percent
white	23%
black	21%
silver	18%
gray	14%
blue	6%
red	8%
brown/beige	6%
yellow/gold	1%
green	1%
other	2%

Figure 4: Sample traffic data from Albuquerque, New Mexico.

2.3 AFRL Wright Patterson AFB Flyover Dataset

An additional dataset provided by the Air Force Research Labs was used to evaluate tracking algorithms. The "WPAFB 2009 Dataset" is a publicly available imagery dataset of a video flyover of Wright Pattern Air Force Base [14]. The video was taken in 2009 and is 21 minutes long, with images taken at ∼1 Hz. Images are provided at six different resolutions in both JPEG and NITF format, and the first 14 minutes of imagery include manually annotated truth data for all vehicles in the scene.

For evaluation of the algorithms in this report, we created an artificial detection dataset from the truth data. A MATLAB script was written to parse the truth data files and generate a set of detections, as would be produced from an object detector on an image. The truth locations were converted from latitude and longitude coordinates to linear distances in meters. This resulted in a dataset similar to the simulated dataset discussed above; however, the source of the vehicle tracks in this dataset is provided by real truth tracks from the WPAFB flyover instead of the SUMO traffic simulation. A sample image from the dataset and a sample plot of seven minutes of track data are shown in Figure 5.

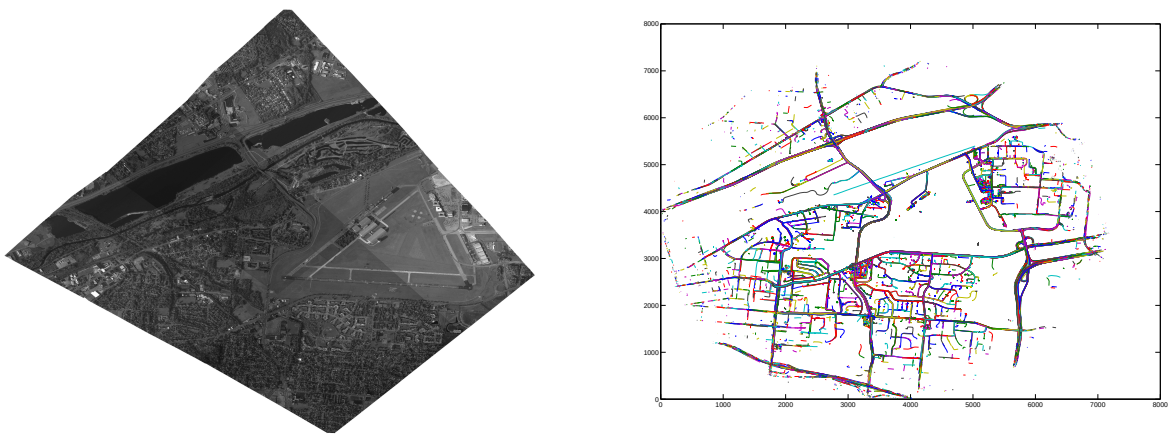

Figure 5: Sample image from AFRL WPAFB dataset and plot of all tracks.

From the full dataset, three Regions of Interest (ROI) were identified in the first 70 frames of image data. From these ROIs three datasets were obtained. The ROI locations as shown on top of the track data are illustrated in Figure 6.

2.3.1 Region of Interest 1

ROI 1 has a total of 68 vehicles and features a turning highway and stopping intersection. Many cars coming from the right side of the image stop at the intersection and all turn left at once, resulting in a large group of vehicles traveling towards the bottom of the video.

2.3.2 Region of Interest 2

ROI 2 is a stretch of 4-lane highway with a slight bend near the top. A total of 85 vehicles pass through the ROI in the dataset.

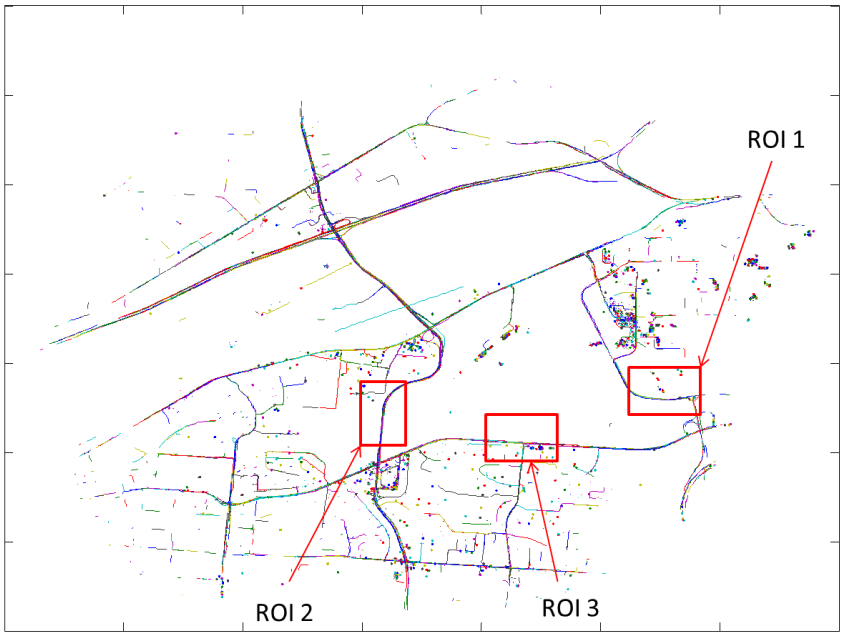

Figure 6: Locations of Regions of Interest in first 70 frames of track data.

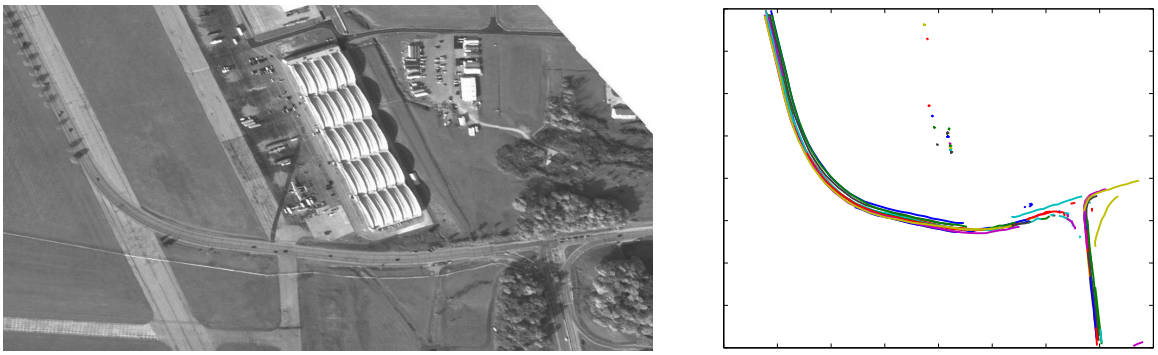

Figure 7: Sample image and all tracks in Region of Interest 1.

2.3.3 Region of Interest 3

ROI 3 contains a popular highway and stoplight intersection. Many cars travel on the highway as well as exit the major roadways and turn off into the surrounding residential neighborhood. In total 125 cars are contained in the ROI video.

3 Evaluation Metrics

A set of evaluation metrics were chosen to compare the performance of algorithms across the various datasets. The first metric, Multi-Object Tracking Accuracy (MOTA), was first defined in [15] and is a single value which incorporates false positives, false negatives, and identity switches. The mostly

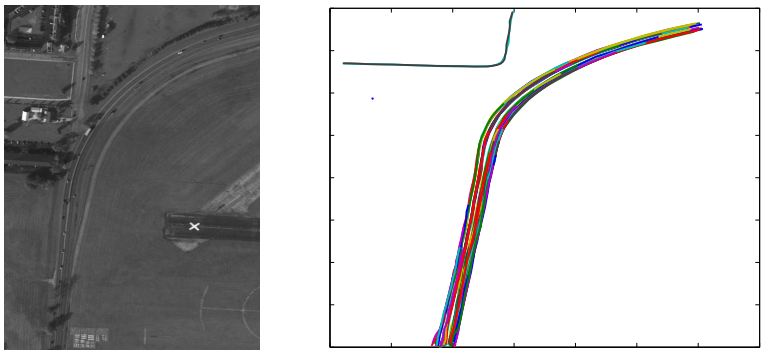

Figure 8: Sample image and all tracks in Region of Interest 2.

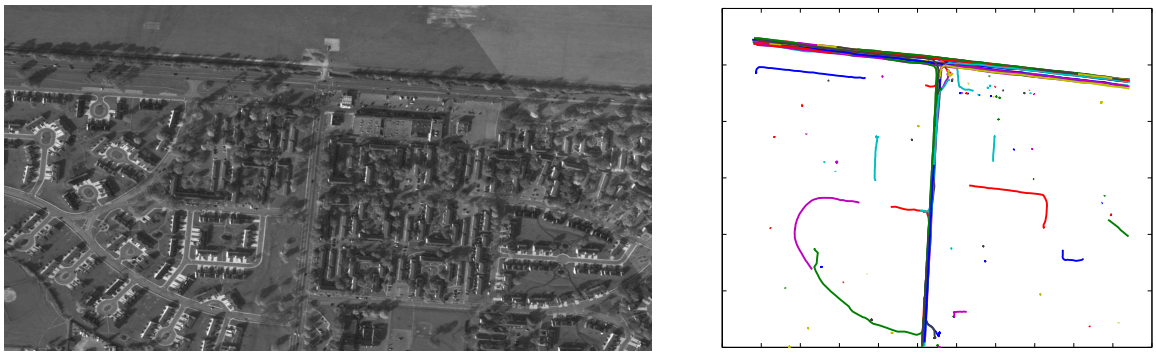

Figure 9: Sample image and all tracks in Region of Interest 3.

tracked (MT) and mostly lost metric (ML) are trajectory based and were first defined in [16]. The tracking metrics employed in this report are described in more detail in Table 3.

4 Probabilistic Multi Hypothesis Tracking (PMHT)

Probabilistic multi-hypothesis tracking, first described in [2], is a batch processing approach that uses expectation-maximization (EM) for data association. One promising aspect of the algorithm for large-scale tracking is that the computational scaling for the number of targets is linear (as opposed to combinatorial for the multi-hypothesis tracker) [6]. Another potential advantage of the PMHT is a "soft" measurement-to-track assignment. The Bayesian inference network (BIN) for the PMHT algorithm is illustrated in Figure 10. The PMHT algorithm estimates the observer state in three steps[2]:

Step 1 The batch joint probability density function (PDF) is marginalized (summed) over all measurement-to-track assignments.

Step 2 Estimate the target states x from the marginal PDF using the expectation maximization (EM) method.

Step 3 Use Bayes Theorem to compute the conditional density for the observer.

The PMHT algorithm, as described in [2], was implemented in MATLAB for evaluation. Typical results for a 3-target simulation are shown in Figure 11. The left plot shows proper convergence. The center plot, with the same tracks but different noise, illustrates the case of no convergence

Table 3: Summary of tracking metrics.

Multi-Object Tracking Accuracy (MOTA)

A single number that incorporates false positives, false negatives, and identity (ID) switches. Defined as

$$1 - \frac{\sum_t FP(t) + FN(t) + IDS(t)}{\sum_t N_{truth}(t)} \tag{1}$$

A MOTA of 1 indicates a tracker with no false associations or ID switches, while a tracker that has one error (either a false positive, false negative, or ID switch) for each ground truth target in each frame will have a MOTA of 0. A negative MOTA is possible for trackers that have more than one error per ground truth target per frame. [15]

Mostly Tracked (MT)

Percentage of targets that are tracked for more than 80% of its detections regardless of identity switches

Mostly Lost (ML)

Percentage of targets that are not tracked for more than 20% of its detections regardless of identity switches

Mostly Singly Tracked (MST)

Similar to MT, but accounting for identity switches (ie, 80% of detections are followed by a single track)

Mostly Singly Lost (MSL)

Similar to ML, but accounting for identity switches

False Positives (FP)

The number of tracked observations that were not true detections

False Negatives (FN)

The number of true detections that were not associated with a track

Identity Switches (IDS)

The number of times a track segment switches between two ground-truth targets

(number of iterations was limited to 100). The right plot gives an example of convergence to the wrong track. After initial investigations, we concluded that the PMHT is very sensitive to the initial conditions and often suffers from convergence problems. Subsequently, we found an article that discusses problems and recommends some solutions to improve the performance of the PMHT, which includes a discussion on initialization and convergence [17]. Based on our experience, and the comments in [17], we concluded that the PMHT algorithm was not well suited for the large-scale tracking problem.

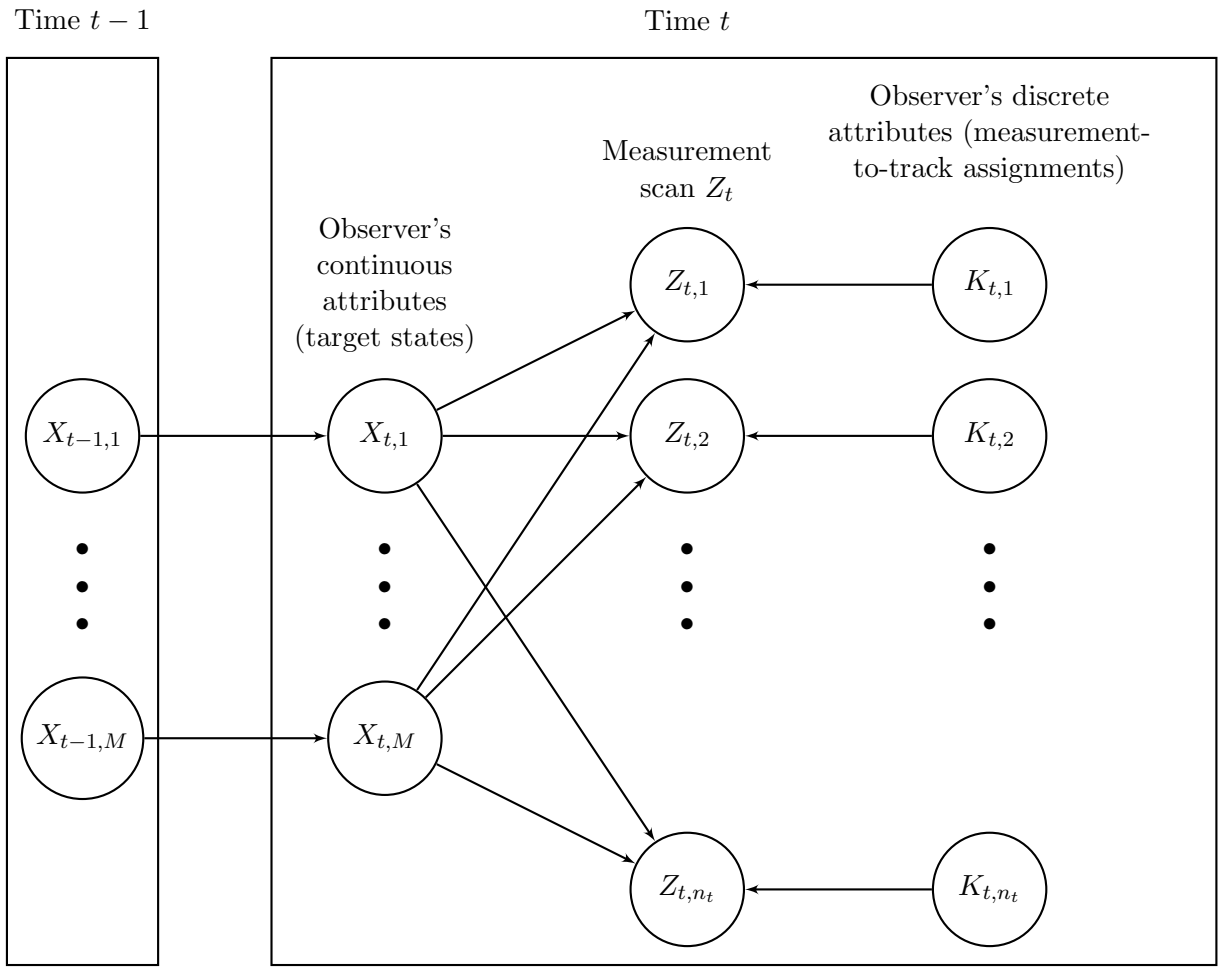

Figure 10: PMHT Bayesian Inference Network (BIN) [2].

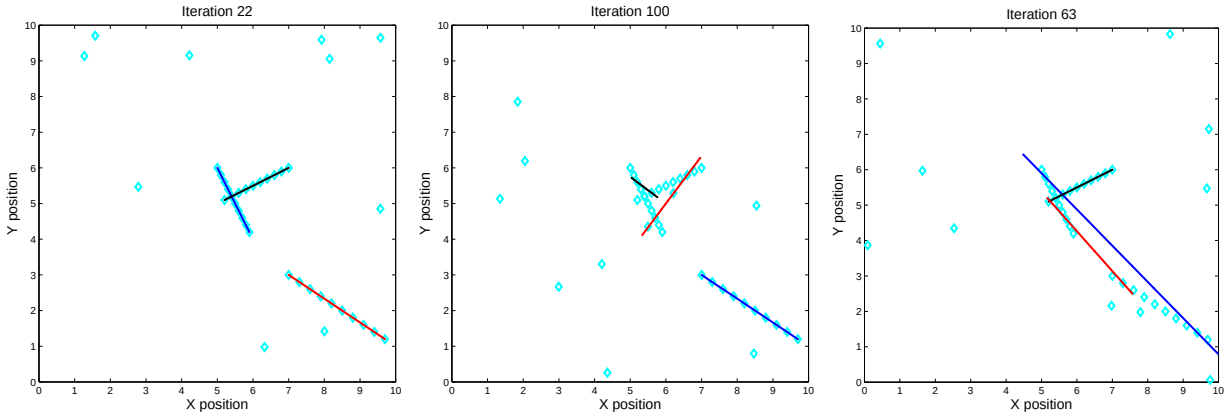

Figure 11: PMHT example results: proper convergence (left), no convergence (center), incorrect convergence (right).

5 Random Sample Consensus (RANSAC)

5.1 Introduction

In 1981 Fischler and Bolles introduced the Random Sample Consensus algorithm (RANSAC) to fit data to a model [18]. RANSAC's predecessors attempt to fit an entire data set to a single model (e.g. Least Squares). If the data set contains gross errors, those gross errors can fatally corrupt the predecessor's results. RANSAC's novelty is that it minimally samples the data set to gather exactly as many measurements as are needed to fit a model, no more and no less. This parameterized model is then used to calculate inliers (points that can be explained by the model and Gaussian measurement noise) and outliers (points that are attributed to gross errors). This revolutionary approach allows RANSAC to reliably fit data sets that contain gross errors [19].

The ability to fit data with gross errors is important for large-scale tracking. Gross errors commonly occur when detection algorithms are applied to find cars. These algorithms find cars, but also detect shadows or trees or sensor noise that momentarily appear to be in the shape of cars.

5.2 Algorithm

RANSAC is a model-based algorithm (see Algorithm 1). The data set d is fit to a specific model M. The model could be a straight line, a circle, a sine wave, etc.

For large-scale tracking of automobiles a constant velocity point-mass model was used. Over a short enough duration, cars act as approximately constant velocity. With two detections $d_1 = (x_1,\ y_1)$ and $d_2 = (x_2,\ y_2)$ at two different points in time t_1 and t_2, the velocity of the car can be estimated: $v_x = (x_2 - x_1)/(t_2 - t_1)$ and $v_y = (y_2 - y_1)/(t_2 - t_1)$. This provides a fully parametrized constant velocity model of the car that can be used to predict the position at any point in time.

$$x(t) = x_1 + v_x(t - t_1) \tag{2}$$

$$y(t) = y_1 + v_y(t - t_1) \tag{3}$$

5.2.1 RANSAC

RANSAC's inputs are: a data set d and a model M. It also had three design parameters: an error tolerance et, how many inliers are required ir, and the maximum number of attempts ma.

RANSAC randomly samples to find an acceptable consensus set, but there may not be an acceptable consensus set. This necessitates the outer *for* loop in Algorithm 1 to limit the number of random samples tried, preventing an infinite *while* loop if no signal is present. If too many samples have been attempted and no consensus set found, RANSAC assumes there is no consensus set and returns that nothing was found (line 10). The maximum number of attempts ma is a design parameter that is adjusted based upon the application.

At each for loop iteration, a minimal subset $S1$ is randomly selected from the full set of detections (line 3). This subset is used to fit the parameters for the model, instantiating $M1$ (line 4). This parametrized model is then used to predict where the constant-velocity car would be at each time step in the data set according to equation (3). Around each of those predicted locations a circle of radius et defines where 'inliers' occur. If a detection is within one of those circles at the correct instant in time, the small deviation is assumed to be due to Gaussian measurement noise. If a detection is not within one of those circles at the correct instant in time, it is an 'outlier' caused by gross errors. How large the design parameter et is depends on the application and the expected size of the Gaussian measurement noise. The set of all inliers in the data set is calculated to form

the consensus set $S1*$ in line 5. If there are as many inliers as are required (line 6), the consensus set $S1*$ is acceptable and RANSAC terminates (line 7). The design parameter ir depends on how many inliers are expected.

Algorithm 1 The Random Sample Consensus algorithm (RANSAC).

```
1:  function RANSAC(d, M, et, ir, ma)
2:      for i ← 1, ma do
3:          S1 ← random_minimal_subset(d, M)
4:          M1 ← instantiate_model(M, S1)
5:          S1* ← calculate_inliers(M1, et, d)
6:          if size(S1*) ≥ ir then
7:              return S1*
8:          end if
9:      end for
10:     return fail
11: end function
```

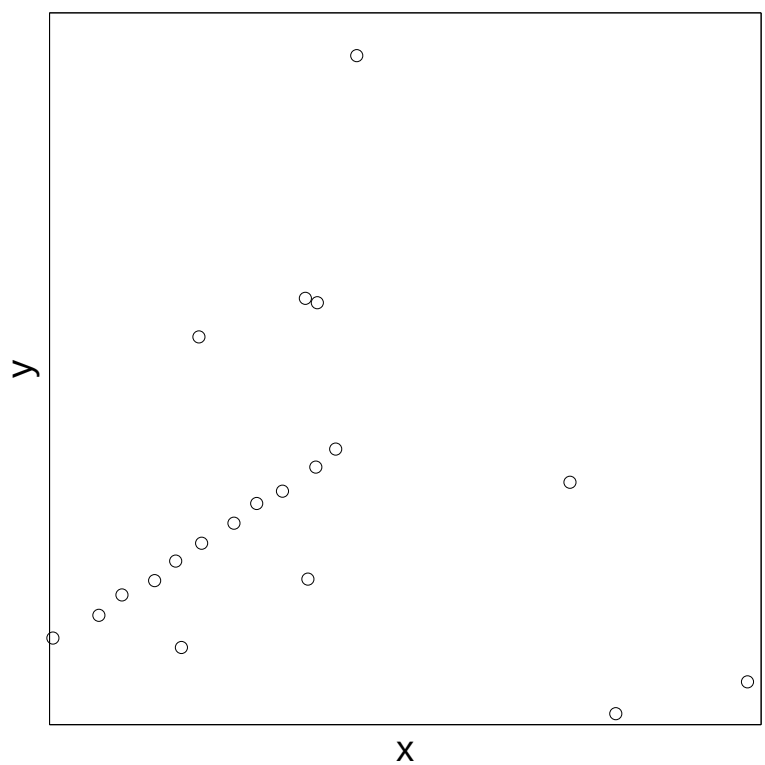

Figure 12: Batch of detections input to RANSAC.

Figure 12 shows an example data set containing a batch of car detections. The time index is printed next to the individual detections. There is quite obviously only one car in the batch.

Figure 13 shows an example of a 'bad' random sample. The parametrized model is blue. The consensus set is pink. The two randomly sampled detections are not both part of the actual signal. This causes only two inliers (the two points randomly sampled). This does not meet the number of inliers required to terminate, so the algorithm loops back to pick another random sample.

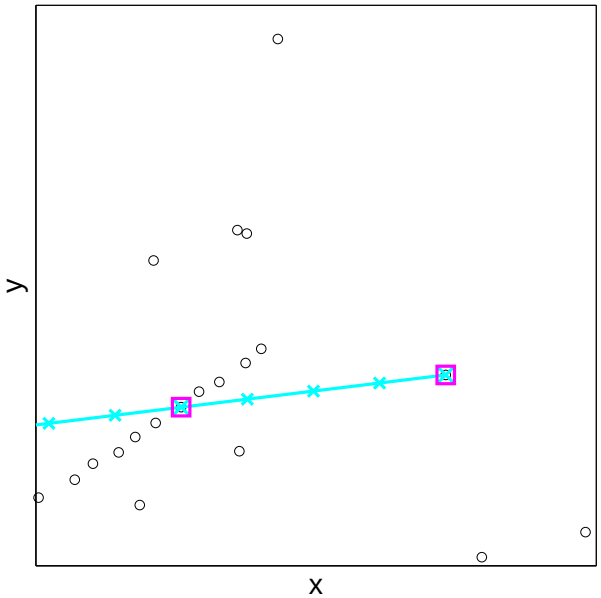

Figure 13: Random sample with only 2 inliers.

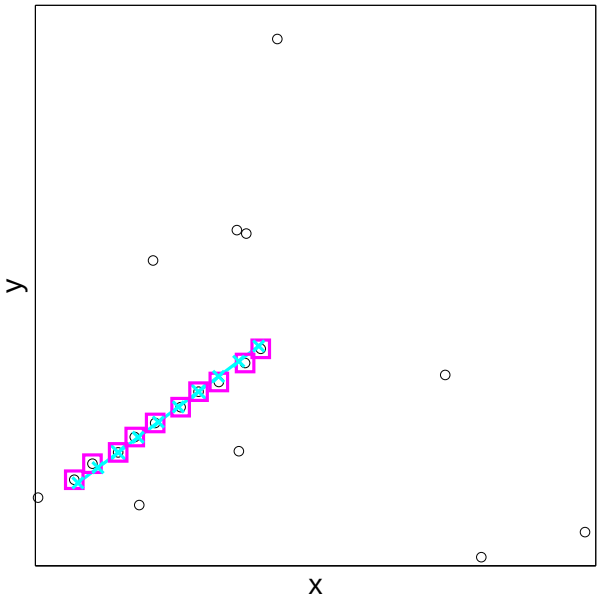

Figure 14: Random sample with 10 inliers.

Figure 14 shows an example of a 'good' random sample. The two randomly sampled detections at $t = 3$ and $t = 7$ are both part of the actual signal. This causes the parametrized model to closely reflect reality. All 10 of the predicted locations have detections that are inliers. This does meet the number of inliers required to terminate, and RANSAC successfully tracks the car.

5.2.2 s-RANSAC

RANSAC, in the traditional sense, looks for a single signal with Gaussian measurement noise in the presence of many additional gross errors. In some data sets, such as the detections of cars for large-scale tracking, there are multiple signals present. Several extensions of RANSAC have looked at finding these signals in parallel, such as multiRANSAC [20]. The parallel RANSAC extensions like multiRANSAC depend on a fixed number of models known in advance. They essentially define a single larger model of much higher dimension and use RANSAC on that combined model (e.g. combine two four-state cars into one eight-state model).

Algorithm 2 The Sequential RANSAC algorithm (s-RANSAC).

```
function s-RANSAC(d, M, et, ir, ma)
    CS ← ∅
    while RANSAC(d, M, et, ir, ma) ≠ fail do
        CS ← CS ∪ {S1*}
        d ← d \ S1*
    end while
    return CS
end function
```

In large-scale tracking the number of signals present typically is not known, but something to be determined. Sequential RANSAC (s-RANSAC) can be used to find an unspecified number of signals in the batch of detections, Algorithm 2.

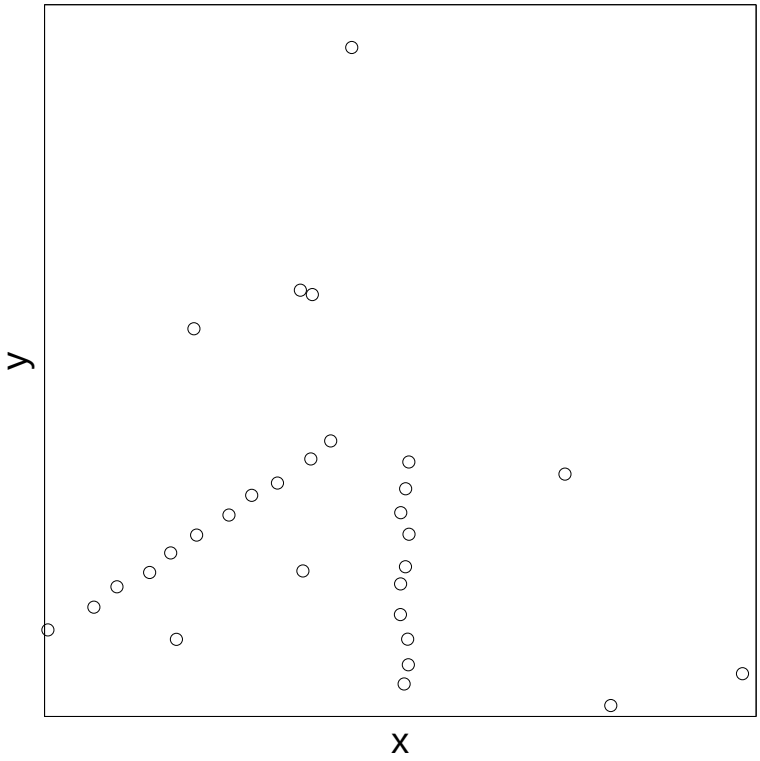

Figure 15: Batch of detections input to s-RANSAC.

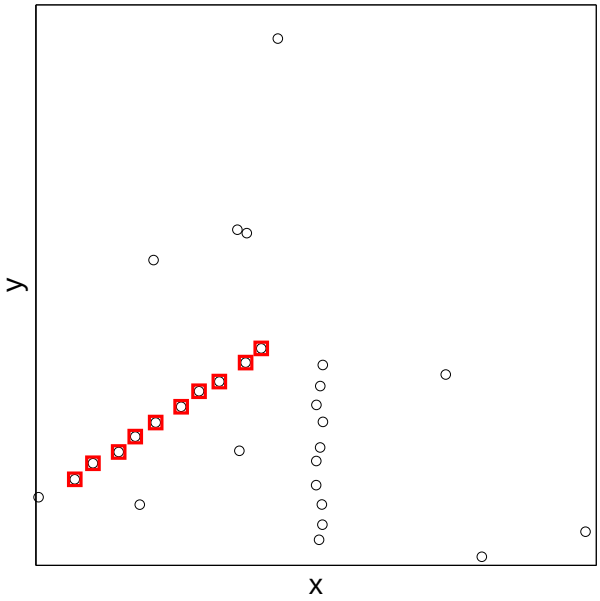

Figure 16: First acceptable consensus set.

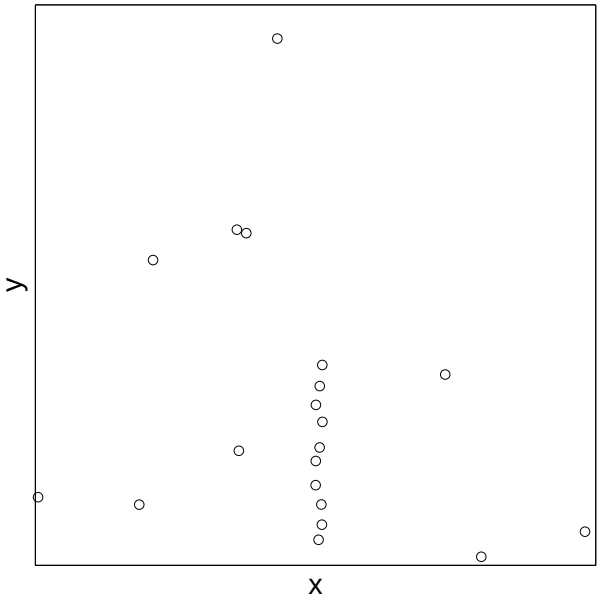

Figure 17: Remaining Detections.

Figure 15 shows the example data set now with a second car present. s-RANSAC starts with no acceptable consensus sets identified (line 2). There is always the possibility that $CS = \emptyset$ at termination if there are no signals present. s-RANSAC effectively runs RANSAC, waits for a single acceptable consensus set to be identified (Figure 16), appends that acceptable consensus set to the

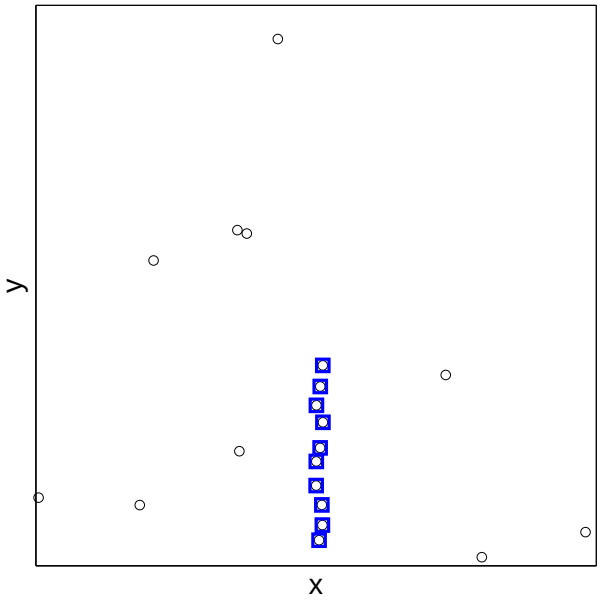

Figure 18: Second acceptable consensus set.

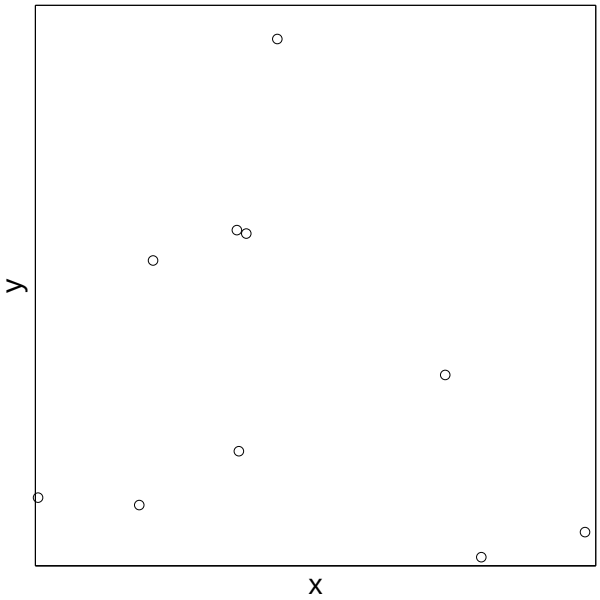

Figure 19: Remaining Detections.

list of conensus sets CS, removes the detections used in the previously found acceptable consensus set (Figure 17), and repeats until no more acceptable consensus sets are found (Figures 18 & 19).

5.2.3 ms-RANSAC

Sequential RANSAC can find multiple cars in short-duration batches of detections. The cars behave as approximately constant velocity. s-RANSAC cannot keep track of those cars over long-duration batches because cars are not actually constant velocity.

Drivers will start at zero velocity, accelerate to a desired speed, take a turn (changing the direction of the velocity vector), then maintain that speed until they have to speed up or slow down, and eventually come to a stop at a stop sign. This behavior is not constant speed, constant velocity, constant acceleration, or even constant jerk. The human driver is inserting unmodeled decisions as input into the system (the throttle position and steering wheel angle). For a model to accurately describe and predict a car and driver over a long duration the driver's desired destination, method of path planning, and driving habits must be known and incorporated into that model (or the steering wheel position and throttle angle directly measured).

RANSAC is not intended for models complicated enough to describe a driver's decisions during a crosstown trip. It can however be utilized over short durations because a car's physical limitations make it operate as approximately constant velocity during that period. s-RANSAC can be applied to short-duration subbatches of the overall data set to produce consensus sets for that short time window. These consensus sets can be easily matched to consensus sets for a successor time window, based on shared detections. This is the idea behind Matched Sequential RANSAC (ms-RANSAC), see Algorithm 3.

ms-RANSAC has three additional design parameters beyond RANSAC and s-RANSAC: the batch size bs, the batch overlap bo, and the matches required mr. When starting ms-RANSAC no tracks TR have yet been identified, and it is possible that the algorithm will terminate with no tracks being identified.

Figure 20 shows a batch of detections from 2 cars at 20 different time steps. If the batch size

Algorithm 3 The Matched Sequential RANSAC algorithm (ms-RANSAC).

```
1: function MS-RANSAC(d, M, et, ir, ma, bs, bo, mr)
2:     TR ← ∅
3:     n ← (number_frames(d) − bs + 1)
4:     for i ← 1, n do
5:         d* ← form_subbatch(d, i, bs, bo)
6:         CS ← s-RANSAC(d*, M, et, ir, ma)
7:         TR ← match_tracks(TR, CS, mr)
8:     end for
9:     return TR
10: end function
```

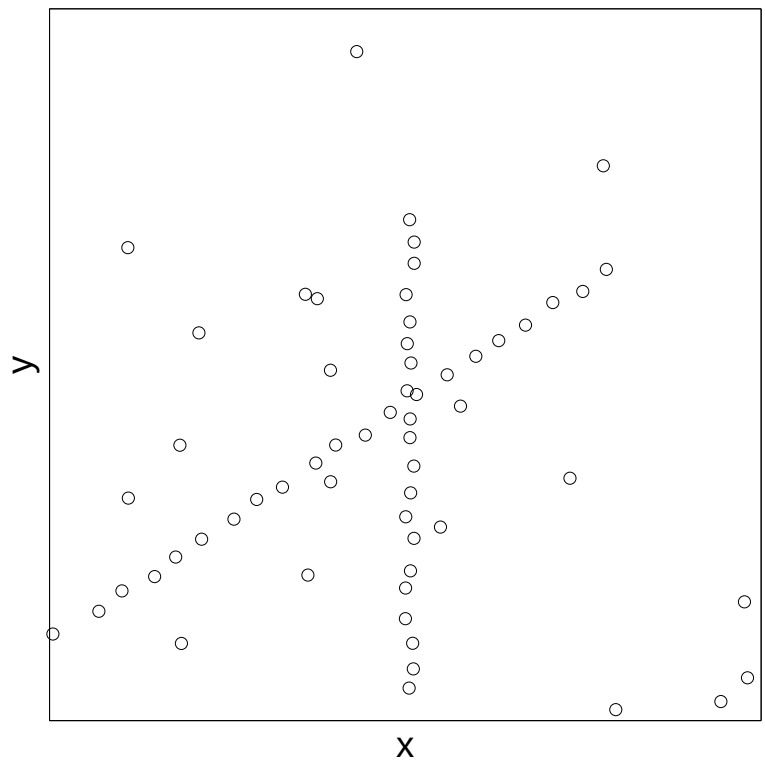

Figure 20: Batch of detections input to ms-RANSAC.

bs is set to 10 the first subbatch is shown in Figure 21. This subbatch is input into s-RANSAC which produces 2 consensus sets in Figure 22. Since there were no previous tracks, matching these 2 consensus sets to nothing generates 2 new tracks. If the batch overlap *bo* is 7 the next subbatch run will contain scans at time steps 4-13, Figure 23. This subbatch is input into s-RANSAC and also produces 2 consensus sets in Figure 24. Finally, the new consensus sets are matched against the existing tracks, Figure 25. Here the number of matches required *mr* is just 1. Since the yellow consensus set has at least 1 detection in common with the red track, the yellow consensus set is an extension of the red track. Similarly the green consensus set is an extension of the blue track. Figure 26 shows the matched tracks that result. This process is continued until there is not enough data left to form a complete new batch (line 3).

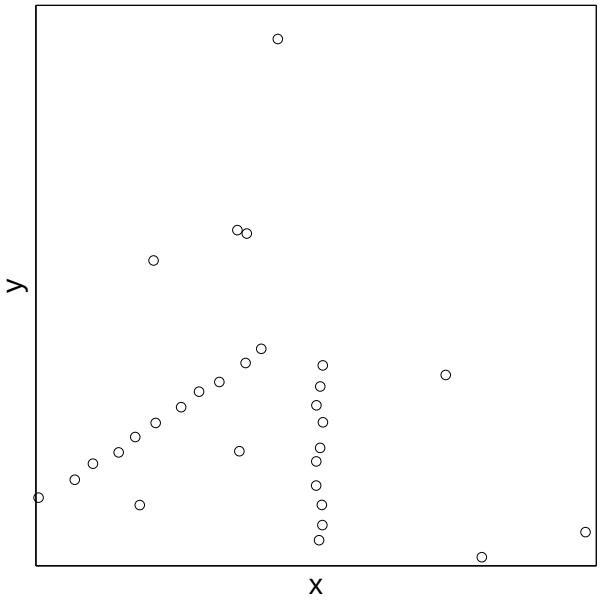

Figure 21: Subbatch formed from scans 1-10.

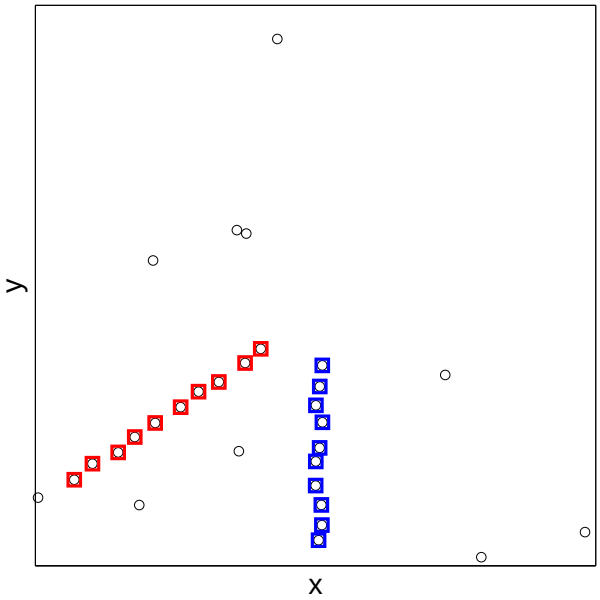

Figure 22: Consensus sets for scans 1-10.

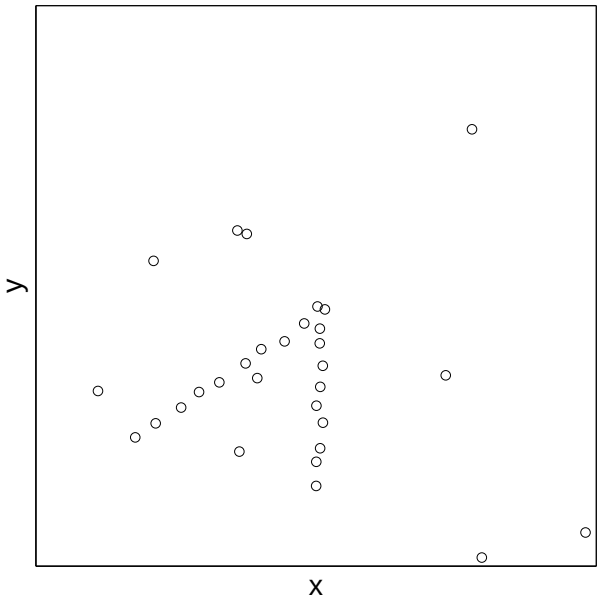

Figure 23: Subbatch formed for scans 4-13.

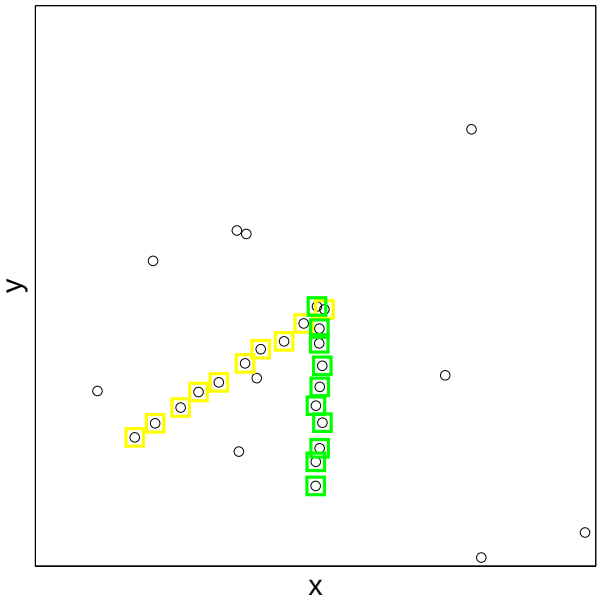

Figure 24: Consensus sets for scans 4-13.

5.3 Implementation

RANSAC, s-RANSAC, and ms-RANSAC were all implemented in MATLAB 2014a. The primary computer executing the code was an HP Z800 workstation with two 2.67 GHz Intel Xeon X5650 processors (12 total cores) and 22 GB of RAM.

The MATLAB license used did not have the Parallel Computing Toolbox, so none of the *for* loops could be converted to *parfor* loops to leverage the additional 11 processing cores. The ms-RANSAC algorithm has several loops that could potentially be parallelized, but the following results were generated by sequentially running the highly parallelizable ms-RANSAC algorithm.

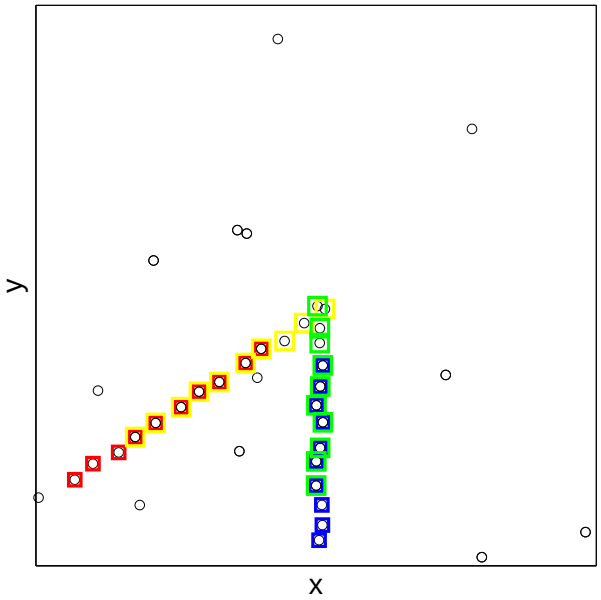

Figure 25: Consensus sets for both subbatches.

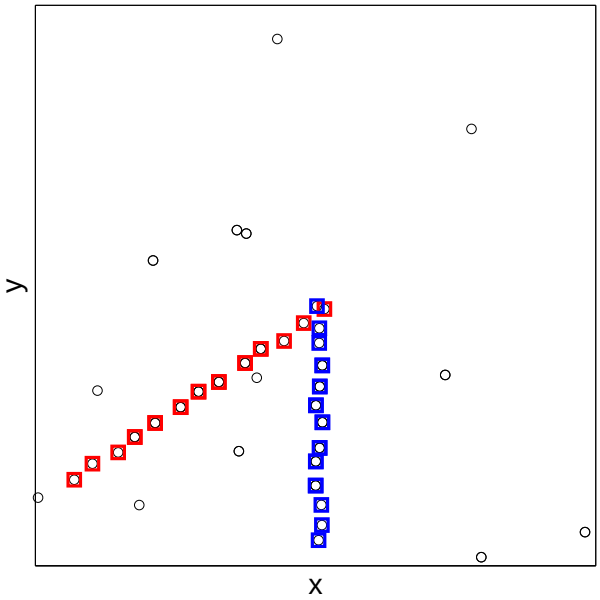

Figure 26: Merged Tracks for scans 1-13.

5.4 Results

Table 4 shows the tabulated results of running ms-RANSAC on all of the data sets.

Table 4: ms-RANSAC results.

dataset	**AFRL-1**	**AFRL-2**	**AFRL-3**	**Crest**	**Socorro**
MOTA	0.761	0.937	0.947	0.4335	N/A
MT	0.721	0.906	0.920	0.909	N/A
ML	0.147	0.024	0.032	0.091	N/A
MST	0.632	0.871	0.840	0.909	N/A
MSL	0.177	0.024	0.032	0.091	N/A
FP	0	0	0	219	N/A
FN	344	114	258	11	N/A
ID	78	50	100	0	N/A

5.4.1 AFRL ROI 1

Figure 27 shows the results of running ms-RANSAC on the AFRL ROI 1 data set. The detections at the extreme bottom right are not included in any tracks because there are too few. They only arrive in the data set at the conclusion and don't produce enough detections to cause a track.

Overall 72% of the actual vehicles are mostly tracked with 63% mostly singly tracked. There are no obvious gaps or flaws in the algorithm apparent from the figure.

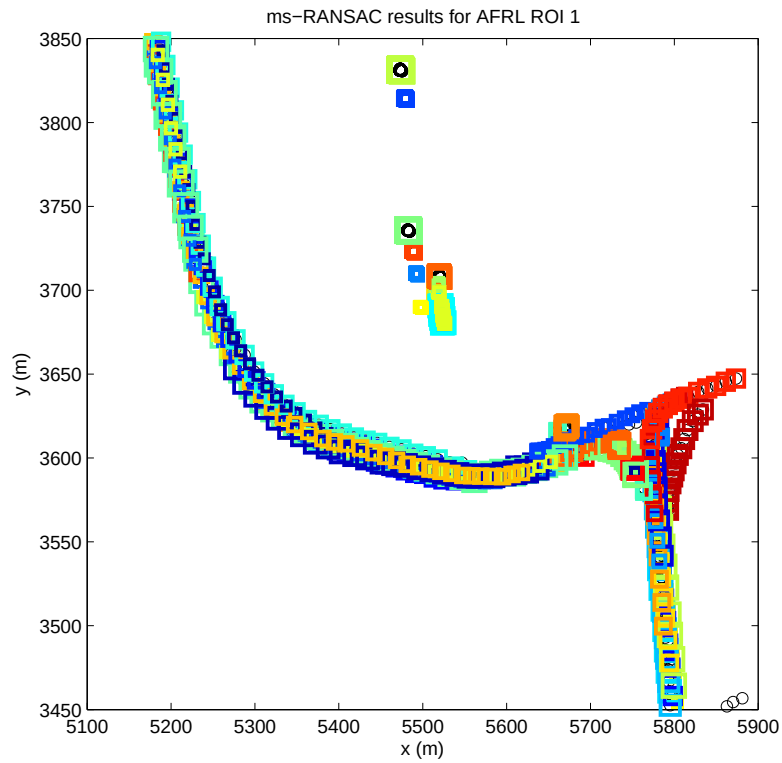

Figure 27: ms-RANSAC Results for AFRL ROI 1.

5.4.2 AFRL ROI 2

Figure 28 shows the results of running ms-RANSAC on the AFRL ROI2 data set. Around 90% of the actual vehicles are mostly tracked with an impressive 87% being mostly singly tracked. One flaw in the algorithm is apparent when looking at the sharp turn around $(3175, 3675)$. The data set is too sparse in time and there are very few detections as that turn is being made. Around 5 detections occur between going almost due South to due West. That appears to be too few for the algorithm to handle. The segments on both ends of the turn are as expected, but the turn is too fast (to few data points during the turn) for a constant velocity model to approximate.

5.4.3 AFRL ROI 3

Figure 29 shows the results of running ms-RANSAC on the AFRL ROI3 data set. Around 92% of the actual vehicles are mostly tracked. Several long tracks are visually obvious showing that the algorithm is handling long-duration tracking including vehicles making turns. This is also confirmed by 84% of the actual vehicles being mostly singly tracked.

5.4.4 Crest

Figure 30 shows the results for the Crest data set. Over 90% of the actual vehicles are mostly tracked and mostly singly tracked, but there are only 11 total vehicles, so this may not be truly representative of 'large-scale tracking'. However, the data is 'real' data that was truthed after collection and does show that ms-RANSAC is able to track vehicles during turns and non-constant velocity trajectories.

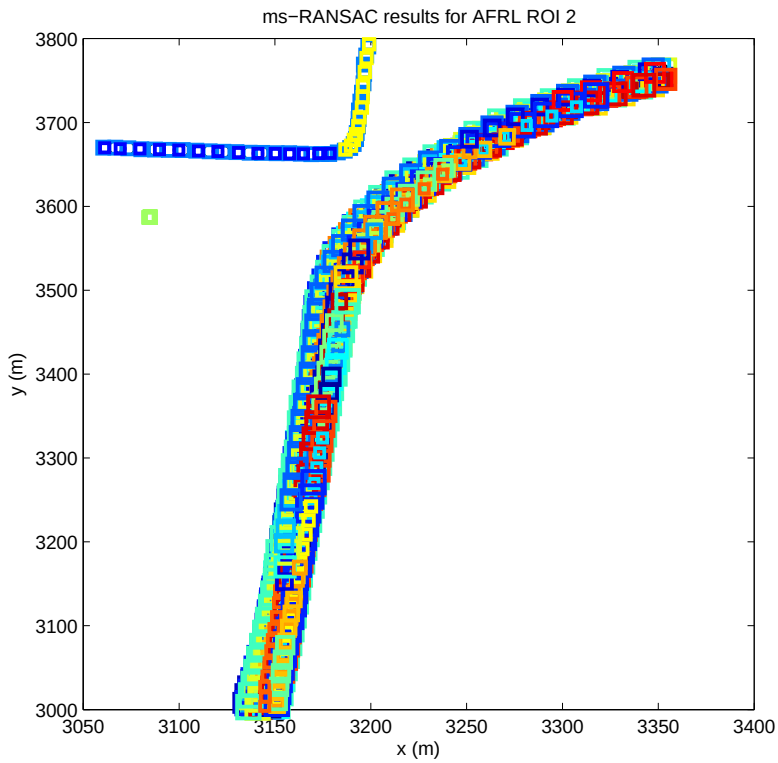

Figure 28: ms-RANSAC Results for AFRL ROI 2.

It is also debatable if some of the detections that were not classified as cars actually were cars (e.g. the strong cluster of detections around $(150, 120)$). These debatable detections are being tracked as stationary cars by ms-RANSAC and producing the 219 false positives.

5.4.5 Socorro

Figure 31 shows the results of running ms-RANSAC on a subset of the Socorro data set (frames 100-199).

5.4.6 Design Parameter Sensitivity

The results in Table 4 are reasonably good. They were generated from specific design parameter values that were manually tweaked to maximize the mostly tracked *mt* parameter. These design parameters can however be modified to yield different performances. Table 5 contains the results if the error tolerance is cut in half in comparison to Table 4. This modification makes ms-RANSAC more selective on what is an 'inlier'. This reduces the number of false positives and ID switches in exchange for also reducing the mostly tracked and mostly singly tracked metrics.

With truth data and sufficient computation time available, the design parameters could be optimized through a cost function composed of the different metrics being evaluated. This would produce the 'best' design parameters for the given cost function.

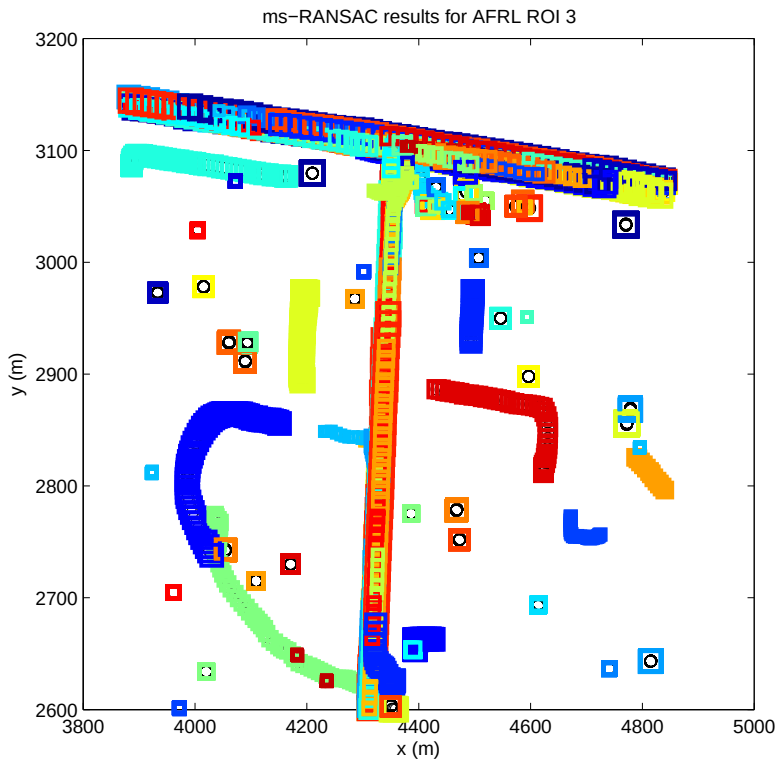

Figure 29: ms-RANSAC Results for AFRL ROI 3.

Table 5: ms-RANSAC results under smaller error tolerances.

dataset	AFRL-1	AFRL-2	AFRL-3	Crest	Socorro
MOTA	0.895	0.856	0.950	0.357	-
MT	0.529	0.706	0.912	0.364	-
ML	0.250	0.047	0.032	0.091	-
MST	0.382	0.329	0.680	0.273	-
MSL	0.265	0.047	0.040	0.091	-
FP	0	0	0	178	-
FN	181	368	289	83	-
ID	4	8	42	0	-

5.5 Future Work

There are multiple directions that future work could pursue. The two primary areas are additional analysis and improving ms-RANSAC's performance.

5.5.1 Additional Analysis

Additional data sets could be evaluated to provide more points of comparison between ms-RANSAC and other large-scale tracking algorithms. Each individual data set will require the ms-RANSAC design parameters to be appropriately tuned (or optimized) to realistically demonstrate the tracking

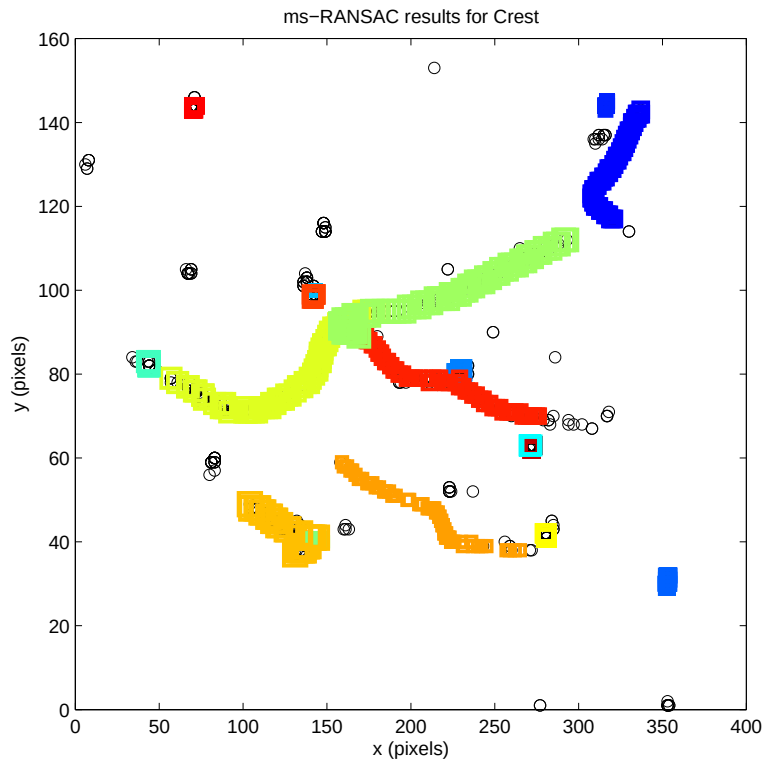

Figure 30: ms-RANSAC Results for Crest.

capability of ms-RANSAC.

Additional metrics could also be identified to characterize ms-RANSAC's performance relative to other large-scale tracking algorithms. The computational complexity of the competing algorithms and the amount of time required for execution are interesting metrics not currently evaluated. They could provide an interesting comparison for algorithms that can be parallelized. The parallelizable algorithms may have higher computational complexity (more total flops required) but may still run faster due to multiple cores operating.

5.5.2 Improved Performance

The performance of the algorithm also has several opportunities for improvement. This includes improving the ability to track cars and decreasing the amount of time required to run the algorithm.

One simple improvement would be the inclusion of color in the constant-velocity model; extending the model to be a constant-velocity constant-color model should not be difficult. This would allow two cars following each other, but of different colors, to be more easily discriminated. This should improve the mostly singly tracked metric. The color dimension would need a new error tolerance design parameter to allow for small deviations in color (due to changes in lighting conditions).

Similarly to including color, the model could be adapted to be an Interacting Multiple Model (IMM). This could include common driving maneuvers such as left/right turns, starting/stopping, as well as constant velocity driving. This should improve the mostly singly tracked metrics by following cars that make sharp/quick turns at intersections.

Finally, parallelization of the algorithm's implementation should allow it to run significantly

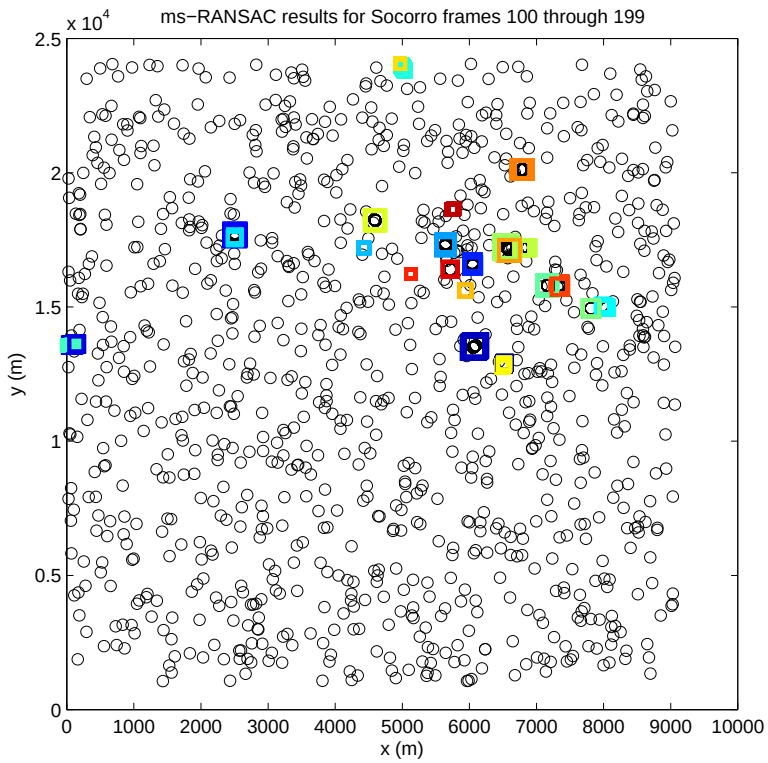

Figure 31: ms-RANSAC Results for Socorro Frames 100 through 199.

faster. The *for* loops in Algorithms 1 & 3 are prime candidates for parallelization since they are of known sizes. Algorithm 3 would need the matching of tracks to be moved outside of the *for* loop, but that could be turned into a post-processing routine run after each of the individual subbatch's consensus sets are found. Also, moving from MATLAB to C may drastically reduce the amount of time required for execution.

5.6 Conclusions

Overall ms-RANSAC performs fairly well. The tracking performance is comparable to the other algorithms considered and could easily be improved by the inclusion of colors and IMMs. The major drawback is that it currently takes a while to run on large data sets (e.g. the Socorro data set). For the Socorro data set it runs in a few weeks. For the other data sets it runs in a few minutes. This problem could be remedied through parallelization.

6 Markov Chain Monte Carlo Data Association (MCMCDA)

Markov chain Monte Carlo data association (MCMCDA) is an algorithm that applies the well-known Markov chain Monte Carlo (MCMC) methods of sampling probability distributions to the problem of data association for multiple target tracking [3, 21]. By applying MCMC techniques to data association, MCMCDA achieves an approximation that approaches the optimal Bayesian estimate of the track association given all previous observations. This section will describe the basics of the MCMCDA algorithm and the results of this algorithm on our datasets.

The central idea of MCMCDA is to use MCMC methods to estimate the posterior distribution

of the data association, $P(\omega|Y)$, where ω is the track associations and Y is the observations. The optimal track association is the maximum of $P(\omega|Y)$, known as the maximum a posteriori (MAP) estimate. The MAP estimate for the track association problem cannot be obtained analytically but can be estimated through methods such as MCMC sampling.

Using Bayes's Rule, the posterior $P(\omega|Y)$ can be expressed in terms of a likelihood and prior,

$$P(\omega|Y) \propto P(Y|\omega)P(\omega) \tag{4}$$

where the constant of proportionality is $\frac{1}{P(Y)}$ which is constant across all ω; consequently it can be ignored in the problem of maximizing the posterior over ω.

6.1 MCMC Methods

The goal of MCMC methods are to generate samples from a distribution, which can then be evaluated for either a MAP estimation or to approximate an expectation over the distribution. Samples are generated by executing moves around the sample space Ω. A move from state ω to ω' is executed with probability $A(\omega, \omega')$.

$$A(\omega, \omega') = \min(1, \frac{\pi(\omega')q(\omega', \omega)}{\pi(\omega)q(\omega, \omega')}) \tag{5}$$

where $\pi(\omega)$ is the stationary distribution to sample from and $q(\omega, \omega')$ is the probability of a move from state ω to state ω'. $A(\omega, \omega')$ is known as the *acceptance probability*. The distribution is explored by this process of repeated proposal and acceptance of moves according to $A(\omega, \omega')$.

6.2 Application of MCMC to Data Association

To apply MCMC to our data association problem, we utilize the method of move acceptances proposed in Section 6.1. To do so, we must define $\pi(\omega)$, the set of valid moves in Ω, and the probabilities associated with those moves $q(\omega, \omega')$. For the stationary distribution $\pi(\omega)$ we use the posterior such that $\pi(\omega) = P(\omega|Y)$. For the set of moves we define five valid operations and their reverse equivalents in Ω-space:

1. *Birth and Death.* A new track is formed (Birth) or an existing track is deleted (Death).
2. *Split and Merge.* A track is split into two new tracks (Split) or two existing tracks are merged into one new track (Merge).
3. *Extension and Reduction.* A track is lengthened by associating new observations past the end of the track (Extension) or a track is shortened by disassociating observations from the end of the track (Reduction).
4. *Update.* Observations in a track are randomly reassigned.
5. *Switch.* Two tracks are split and recombined at a single point.

Graphical illustrations of each of these operations are shown in Figure 32. In each situation where new observations are assigned to a track (as in births, extensions, and updates), the newly assigned observation is selected at random from a pool of potential observations within a gated region based on a maximum velocity from the previous detection in the track (for details see [3]). In cases where tracks or points on a track are selected (such as choosing a track to eliminate in a

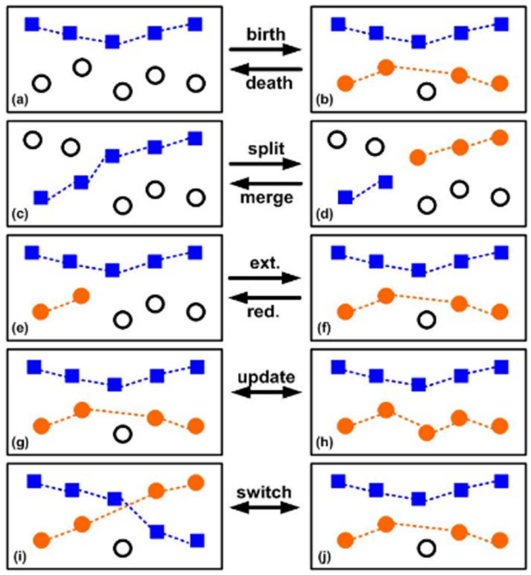

Figure 32: Examples of the five valid MCMC moves [3].

death move or the division point in a split move), the selection is made at random from all available options.

Finally, to complete the MCMC definition for data associations the probability of a particular move $q(\omega, \omega')$ is tracked as the move is selected; whenever a random decision is made in the selection of ω', a running probability is maintained and updated to yield the final move probability.

6.3 Modeling the Posterior

In Section 6, we introduced the posterior for the data association problem $P(\omega|Y)$, and its expression as a proportion to a likelihood and prior in equation (4). To complete the formulation, we must define the likelihood and prior.

6.3.1 Development of the Prior

The prior component of $P(\omega|Y)$, $P(\omega)$, assigns a probability to a set of associations based on prior assumptions about the objects in the scene. The scene characteristics that we will use in our prior formulation are described below:

- p_z - probability of a target terminating in a frame
- p_d - probability of detection
- λ_b - Poisson birth rate of new objects per unit time, volume
- λ_f - Poisson false alarm rate of detections per unit time, volume

The prior is then defined as

$$p(\omega) = \prod_{t=1}^{T} p_z^{z_t}(1-p_z)^{c_t} p_d^{d_t}(1-p_d)^{g_t} \lambda_b^{a_t} \lambda_f^{f_t} \tag{6}$$

where the quantities in the exponents are attributes of the particular association ω; z_t is the number of targets terminated at time t, c_t is the number of targets from time $t-1$ that have not terminated at time t, d_t is the number of detected targets at time t, g_t is the number of undetected targets, a_t is the number of new targets at time t, and f_t is the number of false alarms at time t. A detailed derivation and explanation of these quantities and the prior is shown in [3].

6.3.2 Development of the Likelihood

The likelihood component of $P(\omega|Y)$, $P(Y|\omega)$, reflects the probability that the observed detections could arise from the set of associations ω. This probability is determined from Kalman filters initialized on each track in ω. The process and measurement models used in the Kalman filters are:

$$x_t = Ax_{t-1} + w_k, w_k \sim N(0, Q) \tag{7}$$

$$z_t = Cx_t + v_k, v_k \sim N(0, R) \tag{8}$$

where $x = [x \quad v_x \quad y \quad v_y]^T$ is the state vector, $z = [x \quad y]^T$ is the observation vector, and $N(\mu, \Sigma)$ denotes a normal distribution with mean μ and covariance Σ.

We use a constant velocity model, so A and C are defined as follows for the time step t_Δ:

$$A = \begin{pmatrix} 1 & t_\Delta & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & t_\Delta \\ 0 & 0 & 0 & 1 \end{pmatrix}, \quad C = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \tag{9}$$

For each track τ and frame t we use the Kalman filters to get state estimate $\bar{x}_t(\tau)$ and covariance estimate $B_t(\tau)$, where $B_t(\tau) = C\bar{P}_t(\tau)C^T + R$. Then, we use those estimates along with the actual detections in a track to evaluate a normal distribution for each detection in each track, yielding the likelihood.

$$P(Y|\omega) = \prod_{\tau \in \omega} \prod_{i=1}^{|\tau|-1} N(\tau(t_{i+1})|C\bar{x}_{t_{i+1}}(\tau), B_{t_{i+1}}(\tau)) \tag{10}$$

For the tracking algorithm evaluation, we used the following values for Q and R :

$$Q = 4 \begin{pmatrix} \frac{1}{3} & \frac{1}{2} & 0 & 0 \\ \frac{1}{2} & 1 & 0 & 0 \\ 0 & 0 & \frac{1}{3} & \frac{1}{2} \\ 0 & 0 & \frac{1}{2} & 1 \end{pmatrix}, \quad R = \begin{pmatrix} 0.25 & 0 \\ 0 & 0.25 \end{pmatrix} \tag{11}$$

6.4 MCMCDA Algorithm

The algorithm to execute MCMCDA is shown in Algorithm 4, where $A(\omega, \omega')$ is as shown in equation (5).

This algorithm explores the posterior $P(\omega|Y)$ by proposing and accepting moves ω' according to MCMC sampling. The MAP estimate is maintained in ω_{map}, which is assigned whenever a new maximum posterior value is found.

6.5 Results

Overall, the MCMCDA algorithm produced average results - certainly no better than other comparable algorithms, and at similar or worse computational complexity. Because of the nature of how

Algorithm 4 MCMCDA Algorithm

$\omega \leftarrow \omega_{init}$

$\omega_{hat} \leftarrow \omega_{init}$

for $n = 1...n_{mc}$ **do**

  propose ω' based on ω

  sample uniform random variable $r \leftarrow unif(0, 1)$

  if $r < A(\omega, \omega')$ **then**

    $\omega \leftarrow \omega'$

  end if

  if $\frac{P(\omega'|Y)}{P(\omega_{map}|Y)} > 1$ **then**

    $\omega_{map} \leftarrow \omega$

  end if

end for

MCMC explores the posterior distribution with discrete moves, the more MCMC samples that are taken the better the performance is in general. This can be seen in the performance of the algorithm versus the sample number, as shown in Figure 33. As shown in the Figure, the MOTA and MT metrics generally improve over time; also, the most generic metric MOTA that incorporates FP, FN, and IDS, sees diminishing marginal returns. This agrees with our intuition, as we expect the algorithm to make large gains at the beginning as new tracks are added and extended for easily classifiable detections.

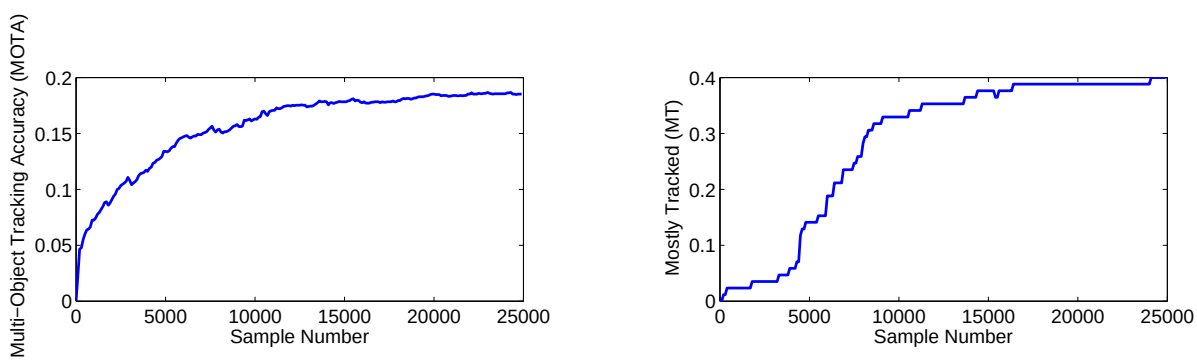

Figure 33: Evolution of MOTA and MT performance as MCMC sampling progresses in the AFRL WPAFB ROI 2 dataset.

For smaller datasets MCMCDA quickly attains optimal performance and plateaus - this can be seen in the Crest dataset of Figure 34. One might wonder why the algorithm's MOTA results occasionally degrade - particularly at the end of the 25,000 sample run. It is important to note that the MCMCDA algorithm does not optimize for MOTA or any other evaluation metric - it optimizes for the posterior probability.

The overall results for all datasets is presented in Table 6. All datasets were run for 25,000

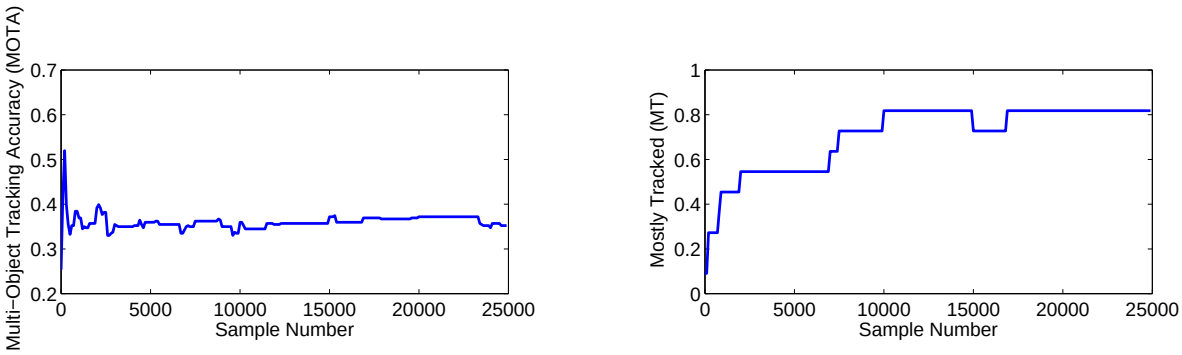

Figure 34: Evolution of MOTA and MT performance as MCMC sampling progresses in the Crest dataset.

Table 6: Summary of MCMCDA results.

dataset	**AFRL-1**	**AFRL-2**	**AFRL-3**	**Crest**	**Socorro**
MOTA	0.5637	0.1852	0.5375	0.3522	0.0111
MT	0.75	0.4	0.8960	0.8182	0
ML	0.0441	0.0471	0	0.0909	0.9987
MST	0.1176	0	0.1360	0.7273	0
MSL	0.3971	0.9059	0.2640	0.1818	0.9987
FP	0	0	0	233	0
FN	195	710	432	29	1472674
ID	576	1424	2662	1	173
SEG	9.6912	20.75	13.98	1.2727	1.4582

samples, except for the Socorro dataset which was run for 10,000 samples due to the size of the dataset. Even limited to 10,000 samples, the Socorro dataset took two days to run; smaller datasets such as Crest and the AFRL datasets took significantly less time (single-digit hours). This shows one of the major drawbacks to this approach: the significant time involved in running the large number of samples required for good tracking results.

MCMCDA performs decently for small datasets - on the Crest and AFRL datasets it achieves high MT, but track consistency is a big problem; few vehicles are singly tracked and there are a high number of identity switches. Additionally, MCMCDA fails to return any reasonable performance on the Socorro dataset - this is most likely due to the sheer complexity of the dataset being too large for the limited number of samples in the algorithm. This could possibly be remedied by using a larger number of samples, but the running time would be prohibitive in the algorithm's current implementation.

7 Tracklet Inference from Factor Graphs

Many methods in multi-target tracking form tracks by first identifying short tracklets, and then iteratively combining tracklets into longer persistent tracks. The crucial problems of these methods are how to identify the initial tracklets and how to combine the tracklets into tracks. We applied one of these tracklet-based algorithms which utilizes the powerful technique of factor graph probabilistic graph models to the problem of large-scale multi-object vehicle tracking [22, 23].

7.1 Algorithm Overview

The algorithm works by forming tracklets in a sliding window over the video frames. Once a set of tracklets are formed in a window, the new tracklets are merged and added to the tracklets from the previous windows to form larger tracks. Within each window the algorithm to form tracklets consists of these six steps:

1. Gather detections in window of length T
2. Form Bayesian networks rooted at each detection in the first frame of the window
3. Find MPE for all networks (get MAP estimate for each detection)
4. Discover tracklets from MPE of networks
5. Combine and prune tracklets within window
6. Combine tracklets from current window with previous windows

The key construction of this algorithm is the Bayesian network that is used for tracklet inference. The next section will go into detail about the formulation of these networks and their usefulness for discovering tracklets.

7.2 Bayesian Network and Factor Graph Formulation

Bayesian networks are a powerful graphical model to solve problems relating to causal relationships among stochastic variables. For multi-target tracking, we use Bayesian networks to determine which detections in a sequence of images are consistent with each other based on appearance and motion dynamics. This is done by constructing a Bayesian network rooted at each detection, where binary variables are placed for other nearby detections and represent whether those detections are consistent with the root based on appearance and motion dynamics. After placing the root detection, subsequent detection variables are added to the network in an iterative fashion by finding all nearby detections to the root in the next frame and connecting them as children to the root; likewise, finding detections in the third frame of the window that are nearby the recently added detections in the second frame, and adding them as children of their respective nearby parent in the second frame. An illustration of two Bayesian networks formed in this manner within a sliding window is shown in Figure 35

Once nodes are added to the network, probabilities must be assigned to the transitions between connected nodes - these probabilities are defined in conditional probability tables (CPTs). The CPTs between parent and child nodes are shown in Table 7. To encode the appearance and motion dynamics into the Bayesian networks, the CPTs incorporate appearance similarity metric $a(o_i^t, o_j^{t-1})$ and motion consistency metric $m(o_i^t)$. We used the same metrics as defined in [22]; in particular, we used a Kalman filter to evaluate the motion consistency of a path $m(o_i^t)$, using the same Kalman

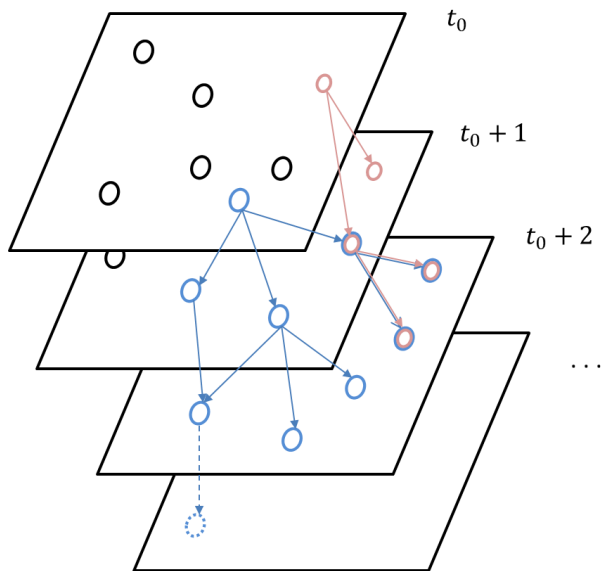

Figure 35: Construction of Bayesian networks in a sliding window.

filter as outlined in the MCMCDA Section 6.3.2. The metric $m(o_i^t)$ is high for paths from the root to the node o_i^t that follow the motion model well, while the metric is low for paths of detections that do not follow the model.

As shown in the CPT of Table 7, if the parent detection is False no information is used to inform the child. However, if the parent is True then the child's state is informed by the motion dynamics and appearance similarity to the parent and to the root node. In addition to the node-to-node CPTs, the appearance similarity to the root node itself is used directly as an evidence

Table 7: Conditional probability table between detections.

	$y_i^t = 0$	$y_i^t = 1$
$y_i^{t-1} = 0$	0.5	0.5
$y_i^{t-1} = 1$	$1 - a(o_i^t, o_j^{t-1})m(o_i^t)$	$a(o_i^t, o_j^{t-1})m(o_i^t)$

Table 8: Conditional probability table for observation nodes.

	$y_i^t = 0$	$y_i^t = 1$
$p(o_i^t \| y_i^t)$	$1 - a(o_i^t, o_0^{t_0})$	$a(o_i^t, o_0^{t_0})$

7.2.1 Motion-only Approximation

In our analysis and testing, some datasets did not include appearance information. In this situation, some modifications were made to the Bayesian network formulation - in addition to eliminating the $a(o_i^t, o_j^{t-1})$ from the transition CPTs, the behavior of the observation nodes was changed. If the motion probability between a parent and child was above a threshold (set to 0.7), the observation CPT of the child was set to have a valid probability of 1. Otherwise, it was set randomly between 0 and 1. This method, admittedly ad-hoc, was chosen to handle an unusual behavior observed in the

Bayesian networks when the appearance and observation factors were removed - the MPE of the networks was alternating frames of completely valid and completely invalid detections. It is believed that this happens because in a network of many detections, the vast majority are invalid and have near-zero motion consistencies $m(o_i^t)$. Additionally, these near-certain motion consistencies are only active when the parent is true (see CPT in Table 7). Because of this, the most likely instantiation of states accesses these near-certain motion consistencies by assigning a frame to be completely true and the subsequent frame to be completely false, resulting in the near-certain consistencies for the majority of the nodes in the false frame. The next frame would then be true, as the false frame does not contribute any information to it's subsequent frame according to the CPT, and the cycle of True-False-True-False frames repeats.

In the presence of appearance metrics and observation nodes as originally formulated this behavior does not arise because of the additional factors at work in the state assignments. But, when left purely to the motion factors this unusual and sub-optimal results occurs, and the ad-hoc assignment of random appearance and perfect observational nodes was used to rectify the situation.

7.2.2 Virtual Detections

One more detail must be added to the currently formulated Bayesian networks to account for occlusions and missed detections. In the process of building the Bayesian network, if a node does not have any nearby detections in the next frame a *virtual detection* is added to account for the possibility of an occlusion or missed detection. This virtual detection would then be treated like any other detection at the evaluation of the next frame, spawning further nearby child detections and building the rest of the network.

7.2.3 Motion Parents

In order to determine the motion consistency term in the CPTs for the network, a path down from the root must be defined which can be evaluated with a Kalman Filter. Because a node may have many children and many parents, multiple paths from any one node to the root can exist. To resolve ambiguities about the path that should be taken when there are multiple parents of a node, the concept of a *motion parent* is defined. The motion parent is chosen as the parent whose path to the child results in the closest estimate to the child's location, where "closest" is defined as the estimate with the highest likelihood given the Kalman filter initialized on the path to the root. When initializing Kalman filters to determine the motion consistency of a node, the motion parents are used to determine the path to the root node.

7.2.4 Converting Bayesian Networks to Factor Graphs

One final complication in the Bayesian Network formulation as currently stated is that it is frequently intractable for large-scale multi-target tracking. The size of the transition CPT in Table 7 for a given child node is exponential in the number of parents; in large-scale multi-target tracking problems there could be tens or hundreds of nearby detections, which makes the CPT impossibly large for many practical problems. To eliminate this problem, we make the following simplifying assumption:

$$p(y_i^t|y_1^{t-1}, y_2^{t-1}, ..., y_K^{t-1}) = \prod_{k=1}^{K} p(y_1^t|y_k^{t-1}) \tag{12}$$

This assumption allows us to treat the relationships to each parent independently, but breaks the assumptions of Bayesian networks. To model this new situation we must use a new tool, the

factor graph. Factor graphs are a probabilistic graph model that are more general than Bayesian networks and are able to handle this assumption. Once the Bayesian network is constructed, it is algorithmically converted to the more general factor graph with this simplifying assumption intact, and further computations are done on the factor graph representation.

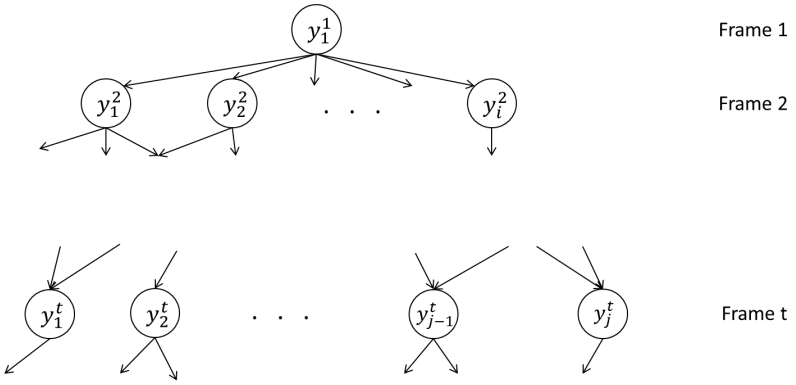

Figure 36: Example Bayesian network.

7.3 Most Probably Explanation and Inference on Factor Graphs

There are multiple algorithms that can be used to solve the MPE problem on a graphical model such as a factor graph. The most popular of these is the Max-Product algorithm, closely related to the Sum-Product algorithm and one of a general class of message passing belief propagation algorithms [24]. Other algorithms have been proposed, including LP-relaxation [25] and stochastic search [26]. In general, MPE is an NP-hard problem [27]. For some simple graphs MPE can be solved in linear time, but for most of the graph structures created by the tracklet algorithm the MPE solution cannot be trivially found.

The output of an MPE query on a factor graph will yield a set of detections that are consistent with the root node. This is demonstrated in Figure 37.

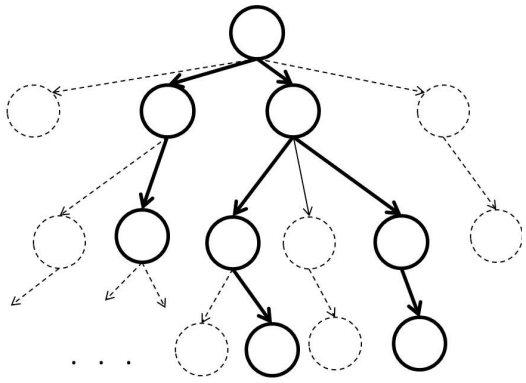

Figure 37: Result of MPE query on Factor Graph, resulting in a set of detections consistent with the root node.

7.4 Implementation of Bayesian Network, Factor Graph, and MPE

To simplify development, open-source packages were used for implementations of Bayesian networks, factor graphs, and an MPE solver. Although a Bayesian network form of the problem is not required, in the development of the algorithm a Bayesian network was used as a useful intermediate step in the formulation of the factor graphs. For these Bayesian networks, the Bayes Net Toolbox (BNT) was used [28].

To solve the MPE problem on our factor graphs, we use the publicly available SLS4MPE package [26]. SLS4MPE uses stochastic local search methods to find the MPE for an arbitrary factor graph. This package was chosen because of its C/C++ implementation and ready-made MATLAB MEX interface; other packages that were considered were either implemented fully in MATLAB (and thus too slow) or implemented in non-MATLAB languages without a MATLAB interface.

To solve each MPE problem, the detections were first organized into a Bayesian network with BNT and then transformed into a factor graph format readable by the SLS4MPE package. The motion model was used on each parent-child relationship of the factor graph to assign probabilities to the factors, and SLS4MPE was used to solve MPE. Within SLS4MPE the MiniBuckets algorithm was used to find an solution - if the solution was provably optimal (a flag returned from SLS4MPE), then it was used as-is. If the MiniBuckets solution was not provably optimal the GLS+ algorithm was run for a maximum of one second and the GLS+ result was used as the MPE solution. This method ensured optimal results for problems small enough to prove optimality by MiniBuckets, while maintaining speed for large problems by using the local search algorithm GLS+. More details about these using specific MPE solving algorithms and the SLS4MPE package can be found in the SLS4MPE readme.

7.5 Tracklet Inference from MPE Result

The MPE result on a factor graph will return the set of detections in the sliding window that are consistent with the root node. We harvest tracklets from each factor graph by starting at the bottom of the tree defined by the motion parents and following paths of valid detections up to the root of the tree. Each path from the root down to valid child detections forms a new initial tracklet.

After a set of initial tracklets is found from all of the factor graphs, the tracklets are pruned. Because of the restrictions on vehicle motion, we prune tracklets by three criteria informed by vehicle motion:

1. *Length* - the length must be longer than a threshold (set to one-half the window length).
2. *Smoothness* - the smoothness must be higher than a threshold (set to 0.1).
3. *Acceleration* - the acceleration must be lower than a threshold (set to 20).

For a given tracklet the calculation of smoothness and acceleration was carried out as described in [22]. Tracklets within a window were also combined with a similarity threshold in order to account for tracks that split and remerge within the same window. The similarity metric was defined as described in the *sim* function in [22] and the threshold was set to 0.5.

7.6 Forming Tracks from Tracklets

In order to form long tracks from short tracklets, tracklets from different windows were combined according to the *sim2* similarity metric in [22]. The merging threshold for this metric was set to 0.75. For datasets where appearance was not measured the appearance term in the *sim2* function was omitted.

7.7 Results

The algorithm was executed on the five datasets described in this report. All datasets were run with the motion-only model, and an additional model including color information was run for the Socorro dataset. The color similarity was defined as

$$a(o_i^t, o_j^{t+1}) = \frac{0.95}{3} \sum_{c \in \{r,g,b\}} e^{-|o_j^{t+1}(c) - o_i^t(c)|/100} \tag{13}$$

where $o_i^t(c), \quad c \in \{r, g, b\}$ is the color red, green, or blue content of the observation i at frame t on a scale from 0-255. The maximum possible similarity was set at 0.95 to prevent perfect confidence in the factor graph. For all datasets, tracks less than 4 seconds long (4 detections) were pruned from the tracker based on prior assumptions of the duration of vehicle tracks. The Noise parameters for each simulation are listed below.

$$Q = 8 \begin{pmatrix} \frac{1}{3} & \frac{1}{2} & 0 & 0 \\ \frac{1}{2} & 1 & 0 & 0 \\ 0 & 0 & \frac{1}{3} & \frac{1}{2} \\ 0 & 0 & \frac{1}{2} & 1 \end{pmatrix}, \quad R = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \tag{14}$$

A summary of the simulation results is found in Table 9. As seen in the table, the algorithm performs well across all datasets. Both Mostly Tracked and Mostly Singly Tracked metrics are high, indicating good coverage of vehicles and relatively few broken tracks. ID switches and false positives are low, resulting in a high MOTA. In comparing datasets it is interesting to note that the AFRL experiments contain many more vehicles and closer spacing than the Crest dataset, resulting in more identity switches and fewer singly tracked vehicles. The SEG metric is the average number of segments per track. A lower number is better, indicating that on average, the tracks have been broken into fewer segments.

Table 9: Summary results for the factor graph inference algorithm.

dataset	**AFRL-1**	**AFRL-2**	**AFRL-3**	**Crest**	**Socorro**	**Socorro w/ Color**
MOTA	0.8834	0.9546	0.9794	0.2980	0.8604	0.910
MT	0.7059	0.9294	0.9360	0.9091	0.6697	0.808
ML	0.2059	0.0235	0.0400	0.0909	0	0
MST	0.6324	0.6235	0.7840	0.8182	0.0666	0.120
MSL	0.2059	0.0235	0.0400	0.0909	0.6773	0.517
FP	0	0	0	275	1	0
FN	188	63	107	9	202,742	131,482
ID	18	56	31	1	5,219	2,101
SEG	1.1176	1.7882	1.3520	1.5455	71.49	49.37

One example of both good vehicle tracking and poor vehicle tracking is exhibited by one path in AFRL ROI 2. In the upper-left portion of the ROI there is a small road with a sharp turn that two cars pass through in the sequence of frames; one passes through faster at roughly 22 MPH while another passes through at around 17 MPH. The tracker is able to successfully track the slower

vehicle through the turn but fails to track the faster vehicle completely in one track, resulting in the faster vehicle being covered by two different tracks. An illustration of the truth detections and the track associations is shown in Figure 38.

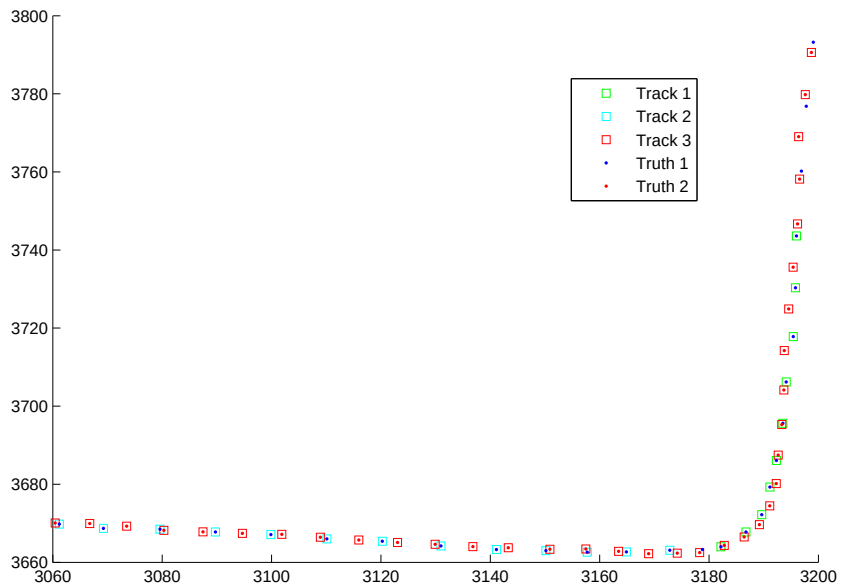

Figure 38: Sample tracking of two vehicles in AFRL WPAFB ROI 2 dataset. The faster vehicle, Truth 1, is covered by two track segments while the slower vehicle, Truth 2, is completely tracked by a single segment.

Another interesting analysis is the number of track segments that cover each vehicle. A histogram of this data for the AFRL ROI 2 dataset is shown in Figure 39. As seen in the histogram the vast majority of vehicles are covered by one or two track segments, which shows how the tracker does a good job of maintaining track consistency in the tightly grouped vehicles on the freeway of ROI 2. The worst tracking performance results in vehicles that are covered by up to five track segments - six vehicles are covered by five segments, which is 7% of the 85 total tracks. Two vehicles are covered by 0 track segments, which means that they are missed completely by the tracker.

Two other experiments were run to further investigate the algorithm's performance. First, the motion model was changed to have a more restrictive noise assumption, shown below.

$$\text{noise parameters } Q = 4 \begin{pmatrix} \frac{1}{3} & \frac{1}{2} & 0 & 0 \\ \frac{1}{2} & 1 & 0 & 0 \\ 0 & 0 & \frac{1}{3} & \frac{1}{2} \\ 0 & 0 & \frac{1}{2} & 1 \end{pmatrix}, \quad R = \begin{pmatrix} 0.25 & 0 \\ 0 & 0.25 \end{pmatrix} \tag{15}$$

The results of this model on the AFRL and Crest datasets are shown in Table 10. The more restrictive model resulted in a tradeoff of tracking performance - on some datasets such as AFRL-2 it improved tracking performance across all metrics. On other datasets, such as AFRL-1, it allowed the tracker to pick up more tracks but also resulted in more identity switches.

The next experiment was a result of noticing a peculiarity in some of the datasets - because of how the camera platform moves in the AFRL dataset, some cars go in and out of the field of view over the course of the video, resulting in very short tracks. This is especially present in AFRL-1,

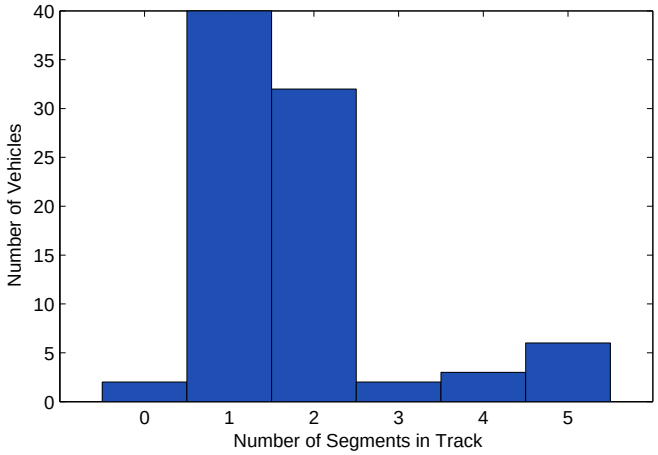

Figure 39: A histogram of the number of vehicles that are broken into multiple track segments in the ROI 2 dataset.

Table 10: Summary results for the factor graph inference algorithm with a more restrictive motion model.

dataset	**AFRL-1**	**AFRL-2**	**AFRL-3**	**Crest**
MOTA	0.9049	0.7637	0.9774	0.2512
MT	0.6324	0.9294	0.9280	0.9091
ML	0.2206	0.0235	0.0400	0.0909
MST	0.5147	0.4824	0.6800	0.6364
MSL	0.2353	0.0235	0.0400	0.0909
FP	0	0	0	283
FN	160	109	135	19
ID	8	36	16	1
SEG	1.0147	1.9176	1.59	1.6364

where 13% of the tracks are only one or two detections long. Because the factor graph algorithm assumes the length of the tracks is at least equal to a certain threshold (1/2 of the window length, or 3) this means that these tracks are never detected by this algorithm. So, an experiment was run on these datasets that only counted tracks that were three detections or longer, removing all of the one and two detection length tracks. The results are presented in Table 11. As seen in the table, this drastically improves the metrics for AFRL-1 and moderately improves metrics for the other AFRL datasets.

7.8 Future Work and Algorithm Benefits and Drawbacks

Overall, the factor graph tracklet inference algorithm exhibits very good performance on the datasets in this experiment. The algorithm has good coverage of vehicles with decently few identity switches and good track consistency. Future work on this algorithm could improve performance by refining the motion and appearance models used in the probability assignments - due to time

Table 11: Summary results for the factor graph inference algorithm for long tracks only.

dataset	**AFRL-1**	**AFRL-2**	**AFRL-3**
MOTA	0.8915	0.9518	0.9797
MT	0.8136	0.9518	0.9435
ML	0.0847	0	0.0323
MST	0.7288	0.6386	0.7903
MSL	0.0847	0	0.0323
FP	0	0	0
FN	172	60	105
ID	18	56	31
SEG	1.1176	1.7822	1.3520

constraints in this project, a thorough testing of the motion model assumptions against the datasets could not be performed, resulting in a model that is likely sub-optimal.

Follow-on work from the authors of the original algorithm added a second tracker based on regression that allowed the tracker to better follow vehicles through patches where the motion of the vehicle changed significantly - such as stops [29]. This is very recent work, and applying a similar concept with two parallel trackers is an interesting idea worth pursuing further.

Some of the advantages and disadvantages of this algorithm compared to other tracking algorithms are described below.

Algorithm advantages:

- *Parallelizable for Speed* - Many computationally intensive parts of the algorithm are independent of one another and can be computed in parallel. All of the factor graphs can be formed in parallel, including the motion and appearance models, and solving MPE on the factor graphs can be enhanced by using a message passing algorithm and parallelization.

- *Elegantly Handles Merged Detections* - the nature of the factor graph inference is such that multiple tracklets can be formed from one detection. This elegantly handles situations of occlusions or merged detections, where multiple tracks pass through the same detection.

- *Incorporates both Appearance and Motion* - the factor graph format is a convenient way to incorporate both appearance and motion in a single model. Although only the Socorro dataset had appearance features of the datasets looked at in this analysis, from the results comparing the Socorro dataset with color features to without it seems obvious that the additional color information is useful in tracking.

Algorithm drawbacks:

- *Requires Solving MPE Problem* - MPE on Bayesian networks and factor graphs is a well-studied problem, but the availability of free packages to solve MPE on factor graphs is limited. For this study the time investment involved with a custom implementation restricted our options and forced us to rely on a relatively obscure MPE solver compatible with MATLAB.

- *Requires Tuning of Multiple Parameters* - In addition to the parameters involved with the chosen motion model, the tracklet inference relies on smoothness, acceleration, and length thresholds that require some tuning.

- *Reliance on Motion Model* - As with many tracking algorithms, in many cases the tracklet inference is only as good as the motion model that is used as the base. If the motion model is poorly tuned or not a good match for the data, the performance of the tracker will suffer. From the results of this study it appears that the constant velocity Kalman filter performs adequately in the vehicle-tracking scenario.

8 Proximity Tracking Algorithm

Because of the nonlinear nature of traffic, we wanted to consider a tracking algorithm that did not utilize a long-term motion model to determine the vehicle tracks. The proximity tracking algorithm initiates the tracking process by finding detections that are close to each other without regard for their motion. These are then combined to form tracklets which are then merged to form the final tracks. The motion model for this combination is essentially a Markov chain, so the criteria for combining the pairs is based solely on the path of the current pairs. The algorithm is presented below. The algorithm parameters are identified by italic font. The values used in the data set analysis will be presented later in the discussion about the data sets.

8.1 Basic Algorithm

The processing steps for the proximity tracker algorithm are enumerated below:

1. First, detections are paired by finding the two closest detections in the next few subsequent frames. The number of following frames to consider is specified by the user (*Look Ahead*). That number is based on the detection probability to account for missing detections. If two detections are within a specified distance (*Max Position Diff*) that should be related to the maximum expected velocity, then the pair is saved for further processing.

2. Tracklets are then formed by linking pairs from one frame to the next that are close to each other (within *Max Position Diff*) and follow the same trajectory within the tolerances specified by several parameters. These parameters specify the velocity tolerance (*Velocity Limit*), the direction tolerance (*Angle Limit*) and the scaling at low velocities. At low velocities, the direction tolerance needs to be relaxed. For instance when a vehicle is stopped, the measured distance changes will solely be the result of noise so direction is irrelevant. Below a limit (*Slow Limit 1*), the direction is ignored and above a second limit (*Slow Limit 2*) , the path must meet the direction tolerance. However between those two tolerances a linear scale for 0 to 1 is used to gradually include the direction tolerance. The linking process starts with the closest point and ends either when a match is found or there are no more pairs in the next frame.

3. Tracklets are then merged to form tracks. In order to allow for the nonlinear maneuvering behavior of the vehicles, the allowed distance between track is increased from the link distance by multiplying *Max Position Diff* by a specified amount (*Merge Tolerance*). Also, the number of frames that can be spanned at the end of a tracklet and the beginning of the next is typically larger then *Look Ahead* and is specified (*Max Frame Diff*). The idea here is that the initial smaller tolerances will reduce the likelihood of including false detections in the tracklets.

Then the merging of the tracklet which are more likely track segments, is performed with wider tolerances.

4. The track is terminated if no detections have been associated with a track for a specified number of frames (*Track Termination*).

5. The final stage evaluates the consistency of the tracks. Because the Markov chain has such a short time component, it was possible for tracks to follow unrealistic paths. So the final stage was to evaluate the consistency of tracks, and to remove unrealistic links. If a step from one point to the next was greater than all the other steps by a specified ratio (*Break Ratio*), the track was separated into two tracks. Then any tracks less than 4 points are discarded.

8.2 Data Sets Results

Five data sets were used to evaluate the performance of the proximity tracks. Most of the proximity algorithm parameters remained the same. The initial values were determined from the Crest data set and most worked reasonably well for the other data sets. It also appeared that the results were not particularly sensitive to changes in the values with the exception of the *Max Position Diff* and *Look Ahead*. Since we were interested in evaluating the algorithm and not necessarily in finding the best results, there was not much effort made in tuning the parameters. We do have confidence however that the results do reasonably reflect the algorithm's performance.

The Crest data set was the smallest with only 11 tracks shown in Figure 40 in different colors. This was primarily due to the tedious process of determining the true detections and paths. However because the tracks are derived from an actual video taken off the crest of the Sandia mountains, the detection and noise characteristics are real, and therefore, this data set is important. The parameters used on the Crest data set are shown in Table 12. The proximity algorithm essentially found most of all but three of those tracks. The three it missed consisted of only 5 detections and failed to meet the minimum number of points. Of the other tracks, two of the tracks were broken due to the interference of other vehicles. With the small number of tracks, the statistics in the bench marking process were considerably influenced by those events.

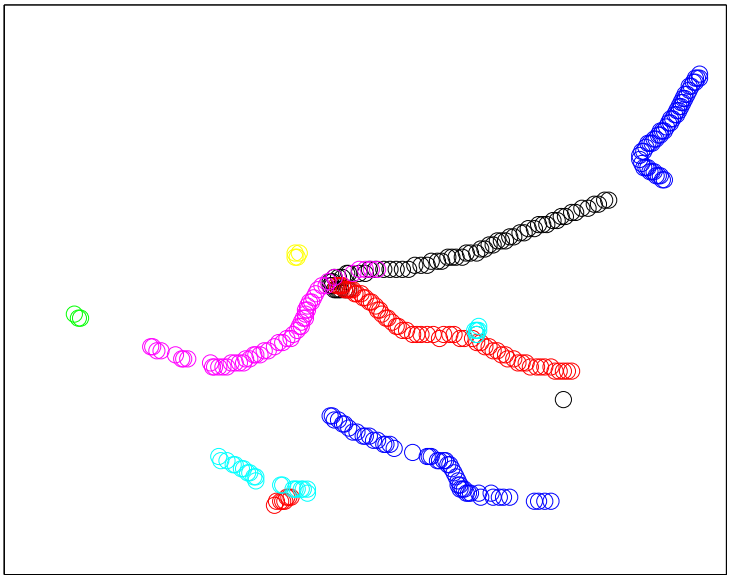

Figure 40: Crest data true tracks.

Table 12: Proximity algorithm parameters for Crest data (pixel units).

Parameter	**Value**
Look Ahead	3
Max Position Diff	5
Merge Tolerance	1.25
Track Termination	5
Max Frame Diff	8
Break Ratio	3.2
Velocity Limit	3
Angle Limit	1.6
Slow Limit 1	0.6
Slow Limit 2	0.8
Dup Velocity Limit	0.2

The simulated Socorro data had different characteristics. Some false detections were added but they had little impact on the tracking process. Most of the vehicles in this set were separated enough that there was very little mixing of tracks in the formation of tracklets with the exception of the interstate. Because the vehicle velocities differed from the Crest data, we used a smaller *Max Position Diff* for this data set as shown in Table 14 below. With those parameters the algorithm mostly tracked about 87% of the tracks. A significant issue with this data set was caused by the probability of detection on some of the tracks. At times there were significant gaps in the detections which broke tracks as shown in Figure 41. The blue, black and red symbols identify the three tracks generated by the proximity algorithm for this single true track. The figure shows a limited number of detections (green dots) between those tracks.

Because of the large number of detections and tracks in this data set, the full region was divided into regions with 512 rows and 514 columns with an overlap of 10 pixels and the regions were processed separately. Since the algorithm looks at all possible detections in the subsequent frame to find the best links, the separation significantly reduced the processing time as the search was conducted on a much smaller number of detections. The overlap which was based on the expected maximum position change, insured that tracks from the different regions could be merged. After all the regions were processed, tracks with identical points within the 10 pixel overlap areas were merged. Of the 781 tracks, 677 were tracked at least 80% of the time, 87 were tracked between 20% and 80% were tracked and 17 were tracked less then 20%. All the true tracks for the Socorro data set are shown in Figure 42. In the figure all the tracks are overlaid in different colors which provides the overall traffic pattern and provides evidence of the large number of tracks but with so many vehicles it is not possible to see the compete path of most of the vehicles. However the important feature is the series of tracks that form the longest path from the top to the bottom of the figure which stems from the simulated traffic on the interstate highway. In Figure 43 all the poorly tracked files are shown. That figure shows that most of the poorly tracked tracks lie along the interstate highway.

Since we divided this data set into subsets before the processing, we did not have the same comparison metrics that were available with the other data sets. Consequently, we also processed

a larger subset (Socorro AOI) which included the side streets and a portion of the interstate to capture the diversity of the full data set. The number of frames was reduced from 6000 to 506 in addition to reducing the area (see Figure 44). When we applied the algorithm to that subset, the bench marking results were comparable to the full data set. We then use those results to discuss the overall algorithm performance in the Comments subsection below.

Table 13: Proximity algorithm parameters for Socorro data (pixel units).

Parameter	**Value**
Look Ahead	5
Max Position Diff	3.5
Merge Tolerance	1.25
Track Termination	5
Max Frame Diff	8
Break Ratio	3.2
Velocity Limit	2.5
Angle Limit	1.6
Slow Limit 1	0.6
Slow Limit 2	0.8
Dup Velocity Limit	0.2

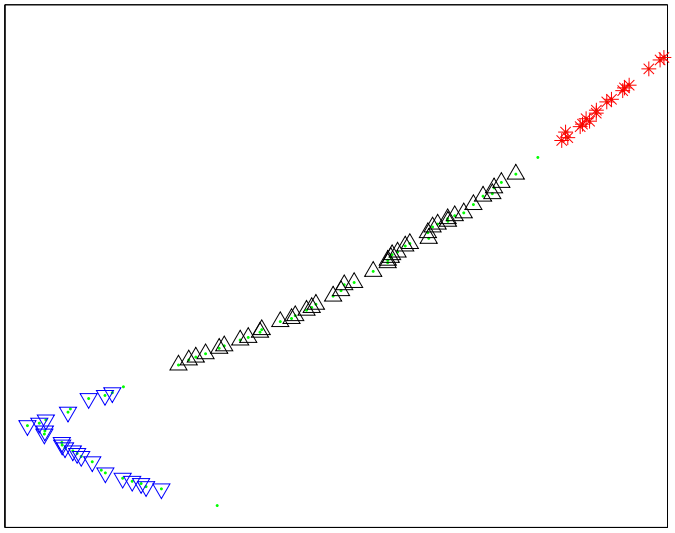

Figure 41: Broken due to gaps in the detections.

The remaining three data sets for the Wright Patterson Air Force Base (WPAFB) provided further insight into the strengths and weaknesses of the algorithm. The data was taken at a slower frame rate so the spacing between detections was much larger so *Max Position Diff* was significantly increased. Also the probability for the detections was higher and consequently *Look*

Figure 42: True tracks for Socorro simulated data.

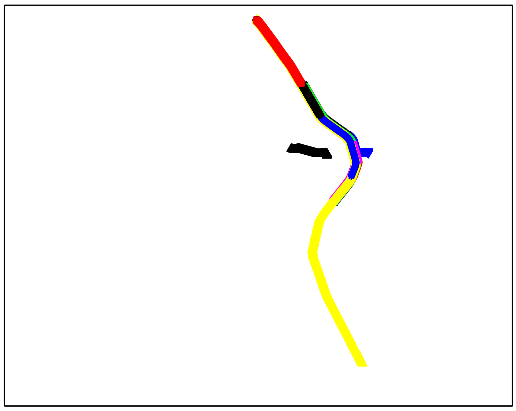

Figure 43: Socorro tracks that were mostly lost

Figure 44: Socorro area of interest tracks.

Ahead was reduced.

Figures 45, 46 and 47 show the true tracks for the three regions. The first data set primarily consisted of the intersection of a curved major road with a straight road with a few stopped or slow-moving vehicles. There were several vehicles in the region of the intersection. Of the 68 tracks, 47 were mostly tracked and 17 were mostly lost. Most of the lost tracks were in the region of the intersection where there were several vehicles in close proximity. A major four lane highway was the primary feature in the second region and that highway had almost all of the tracks. It was the most stressing scenario in all the data sets for the proximity algorithm, where there were multiple lanes containing many vehicles in close proximity moving at high rates of speed. For this data set of 85 tracks, only 12 were mostly tracked including all 4 that were not on the major highway while 19 were mostly lost. The third region had the most variety in traffic patterns. It had a major multiple lane road that intersected with a smaller road. In addition there were several isolated tracks as

Table 14: Proximity algorithm parameters for Wright Patterson Air Force Base data (pixel units).

Parameter	**Value**
Look Ahead	1
Max Position Diff	20
Merge Tolerance	1.25
Track Termination	5
Max Frame Diff	8
Break Ratio	3.2
Velocity Limit	3
Angle Limit	1.6
Slow Limit 1	0.6
Slow Limit 2	0.8
Dup Velocity Limit	0.2

well as many relatively stationary vehicles. The proximity algorithm performed well on the third data set, mostly tracking 99 of the 125 tracks while only mostly losing 4 short small tracks.

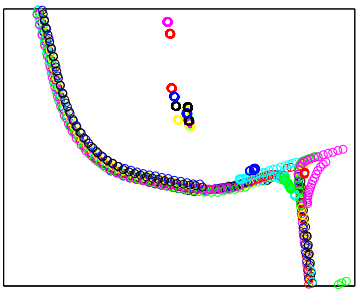

Figure 45: WPAFB ROI1 true tracks.

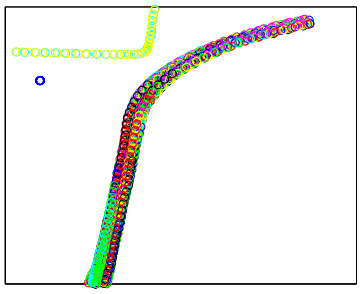

Figure 46: WPAFB ROI2 true tracks.

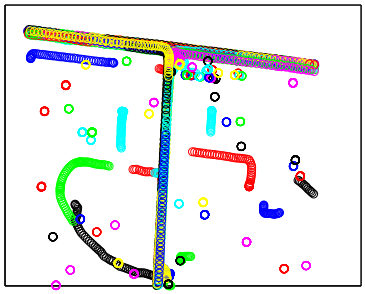

Figure 47: WPAFB ROI3 true tracks.

8.3 Comments

A summary of the data sets is provided in Table 15 below. The number of frames for the data sets was the same for all but one, the Socorro AOI while the number of detections had a wide variation. In the table, the number of mostly tracked and the number of mostly lost is derived from the percentage of a true track that had an associated track from the algorithm. If at least 80% was matched with associated tracks it was considered mostly tracked and if less then 20%, it was mostly lost. The ratio of the number of true tracks and the number mostly tracked and the number of mostly lost provides insight on how well the algorithm did on each data set. It had the most difficulty with WPAFB ROI2 and performed quite well on the Socorro AOI set. The small number of tracks in the Crest data limits increases the uncertainty of its results, but analysis of the results indicated that the algorithm might well have been impacted by the noise in that data. The extent that tracks are broken is indicated by the ratio of the number of found tracks with the

number true tracks. However, since the Socorro data had more frames and consequently longer tracks, those results need to be appropriately scaled. We divided the ratio by the number of frames and show the scaled ratio in the last row. While not rigorously correct since there is considerable variation in the track lengths, it does indicate that the WPAFB ROI1 and Socorro data likely had the fewest breaks per track length. Also note that the value for the Crest data is the largest and is possible evidence that the noise impacted the algorithm.

The performance of the proximity tracker depended highly on the type of traffic pattern in the data. It usually performed well on isolated tracks, stationary vehicles and roads where the vehicles were well spaced. On highways consisting of multiple lanes and many vehicles moving at a high rate of speed in parallel or at intersections with several vehicle in close proximity, the algorithm performed more poorly. However, that result is expected since the initial pairing is based solely on proximity and there is a higher probability that the distance between detections of different vehicles on separate lanes is smaller than the distance between detections of a single fast-moving vehicle. To compensate, the detections need to be provided at a higher frame rate thereby reducing the distance between detections for a single vehicle. Alternatively a constrained tracking feature could be added to the algorithm to prevent linking different vehicles across lanes. We would also suggest that the number of broken tracks could be reduced with a higher frame rate as well. On the Socorro data a larger *Track Termination* value would have helped as well because of the detection gaps. However, the merging algorithm is quite simple, so a more sophisticated track merging procedure would likely reduce the broken tracks as well.

The final row in the table shows the number of inconsistent tracks that were found and were separated into two tracks. A study of those tracks and the reason for the inconsistent links would probably be a fruitful enterprise. It could well lead to an improved process for forming tracklets and tracks. It is particularly interesting that the Crest data had such a high percentage of separations and had the real noise from a video.

This algorithm was implemented in batch mode for simplicity and to allow for detailed analysis of its performance. A real time version however could easily be implemented. We would recommend however that the active tracks be separated by position to limit the amount of effort required to search for links and tracklets in creating the tracks.

Table 15: Summary of proximity tracker results.

dataset	**Crest**	**AFRL-1**	**AFRL-2**	**AFRL-3**	**Socorro AOI**
Number Frames	71	71	71	71	506
Number Detections	881	1763	2617	6685	69969
Number True	11	68	85	125	241
Number Mostly Tracked	4	47	12	99	220
Number Mostly Lost	3	17	20	6	10
Number Found	25	72	148	201	1875
Broken Ratio	0.032	0.015	0.025	0.023	0.015
Breaks	5	3	14	22	35

9 Algorithm Comparison

A table comparing the results of the four different algorithms - RANSAC, MCMCDA, Factor Graph, and Proximity Tracker, is shown in Table 16. The metrics described in Section 3 are used to evaluate each algorithm. For the Socorro dataset, an additional tracking algorithm described in [30] was also evaluated. It employs a more traditional approach with a standard Kalman filter utilizing a Discretized Continuous White Noise Acceleration (DCWNA) motion model [11]. Track termination was identified by 5 consecutive missed detections, while track initialization utilized two-point differencing. The ID switch (IDS) and false positive (FP) metrics were not calculated for this approach.

Table 16: Comparison of Algorithm Results

Dataset	**Algorithm**	**MOTA**	**MT**	**ML**	**MST**	**MSL**	**FP**	**FN**	**IDS**
Socorro									
	RANSAC	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
	MCMCDA	0.011	0	0.999	0	0.999	0	1,472,674	173
	Factor Graph	0.8604	0.6697	0	0.0666	0.6773	0	202,742	5219
	Proximity	0.9012	0.8668	0.0218	0.2420	0.1575	0	144,982	2103
	Kalman Filter	0.9030	0.9138	0	0.0313	0.5392	N/A	92,721	N/A
Crest									
	RANSAC	0.4335	0.909	0.091	0.909	0.091	219	11	0
	MCMCDA	0.352	0.818	0.090	0.727	0.182	233	29	1
	Factor Graph	0.298	0.909	0.091	0.818	0.091	275	9	1
	Proximity	0.5813	0.3636	0.2727	0.2727	0.2727	81	87	2
AFRL ROI 1									
	RANSAC	0.761	0.721	0.147	0.632	0.177	0	344	78
	MCMCDA	0.564	0.750	0.044	0.118	0.397	0	195	576
	Factor Graph	0.883	0.706	0.206	0.632	0.206	0	188	18
	Proximity	0.9126	0.6912	0.2500	0.5588	0.2500	0	142	12
AFRL ROI 2									
	RANSAC	0.937	0.906	0.024	0.871	0.024	0	114	50
	MCMCDA	0.185	0.4	0.047	0	0.906	0	710	1,424
	Factor Graph	0.955	0.929	0.024	0.624	0.024	0	63	56
	Proximity	0.4528	0.1412	0.2353	0.1059	0.3294	0	1,299	244
AFRL ROI 3									
	RANSAC	0.947	0.920	0.032	0.840	0.032	0	258	100
	MCMCDA	0.538	0.896	0	0.136	0.264	0	432	2,662
	Factor Graph	0.979	0.936	0.040	0.784	0.040	0	107	31
	Proximity	0.9150	0.7920	0.0480	0.6240	0.0650	0	538	30

All of the algorithms had difficulty with broken tracks, which should be expected with urban tracking. This can be addressed by incorporating additional sensor information that aides in the association step. The addition of color for example in the Factor Inference Graph algorithm resulted in improved performance (see Table 9).

The next section presents phenomenology results from analysis of typical simulated vehicle targets.

10 Imaging Resolution Study Using DIRSIG

This section explores the phenomenology of typical vehicle targets. Signatures of a Peterbilt 359 semi truck and an Audi A3 car were studied to gain insight into the relationships between different sensing parameters.

10.1 Generating Synthetic Imagery with DIRSIG

Varying imaging platform resolutions can be evaluated by generating synthetic radiometric target imagery. DIRSIG is used to create this synthetic data. "The Digital Imaging and Remote Sensing Image Generation (DIRSIG) model is a first principles based synthetic image generation model developed by the Digital Imaging and Remote Sensing Laboratory at the Rochester Institute of Technology." [31]. The model is capable of generating hyper-spectral imagery from visible through the thermal infrared spectrum. This section describes radiometric image simulations using DIRSIG, as well as a study of the effect of platform resolution on observed target radiance.

10.2 Scene Simulation

10.2.1 Geometry Setup

DIRSIG simulations were set up under varying sun position, imaging platform position, and target azimuth rotation configurations. Sun positions are shown in Table 17. The imaging platform positions follow a simulated GPS track. The platform positions are shown in Table 18. Simulated targets are a Peterbilt 359 semi truck and an Audi A3 car. Both targets are rotated from 0° to 330° azimuth East of North in 30° increments. This yields 11 sun positions × 6 platform positions × 12 target rotations = 792 simulations/target.

Table 17: Simulated sun positions.

Elevation [°]	**Azimuth** [°]
15	90
30	90
45	90
60	90
75	90
90	N/A
105	270
120	270
135	270
150	270
165	270

All scenes were simulated at 0.25 m ground sampling distance (GSD). Scene size is 100 m × 100 m from the nadir imaging platform case. Targets were simulated on an asphalt background. Background only simulations without any target were also run for all platform and sun combinations. MODTRAN was used to simulate the atmospheric model used by DIRSIG. The DIRSIG mid-latitude summer weather profile was employed in all simulations. A U.S. standard tape5 profile was used for MODTRAN calculations. The imaging platform sensor spectral response ranges from 0.40 to 1.10 microns in 0.01 micron channels. Each spectral channel was simulated with a square

Table 18: Platform Positions of a Simulated GPS Track

Azimuth [°]	**Elevation** [°]	**Range** [km]
180	90	20200
325	75	20400
339	60	20900
341	45	21700
340	30	22800
335	15	24100

response. Spectral bands are integrated into the bands of interest shown in Table 19. Images generated by DIRSIG are in units of spectral radiance $[W/cm^2/sr/micron]$. All simulations were run with DIRSIG 4.5.3.

Table 19: Bands of interest to which DIRSIG output data is integrated.

Blue Edge [microns]	**Red Edge** [microns]
0.42	1.10
0.63	0.69
0.92	0.96
0.86	0.89
0.52	0.60
0.75	0.90
0.42	0.52

10.2.2 Peterbilt 359 Semi Truck

The first target simulated was a Peterbilt 359 semi truck. The truck was painted with white car paint, and has glass, rubber, and tar where appropriate (windows, tires, etc.). A sample simulation is shown in Fig 48. The relative dimensions of the semi-truck are shown in Fig 49.

10.2.3 Audi A3 Car

The other target simulated was an Audi A3 car. The car is painted with white car paint, and has glass, rubber, and tar where appropriate as in the case of the semi-truck. A sample simulation is shown in Fig 50. The relative dimensions of the car are shown in Fig 51. In contrast to the truck, the car has a much lower angle windshield. This allows direct sun reflections from the windshield to be seen by the imaging platform.

10.3 Resolution and Pixel Phasing

The primary metric of interest in evaluating the synthetic imagery generated is peak radiance above background. To compute this, a target empty background simulation from the matching sun and platform positions was subtracted from each simulation image on a per pixel basis. The peak

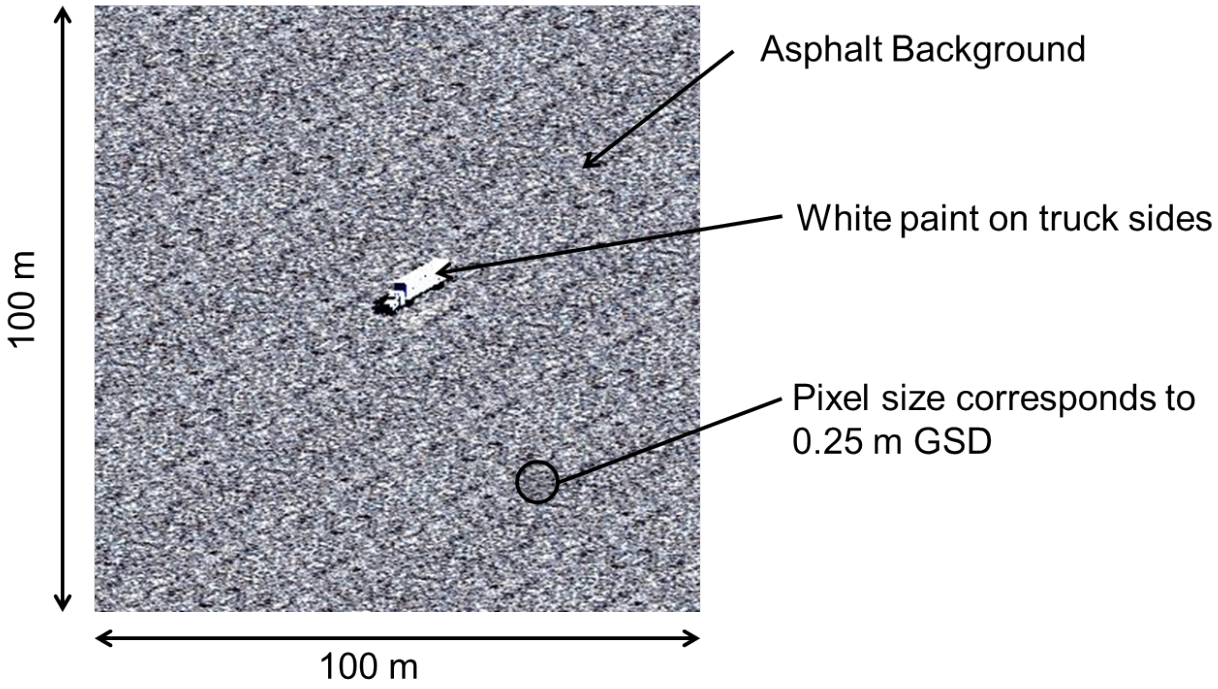

Figure 48: Simulation setup for the Peterbilt 359 semi-truck target.

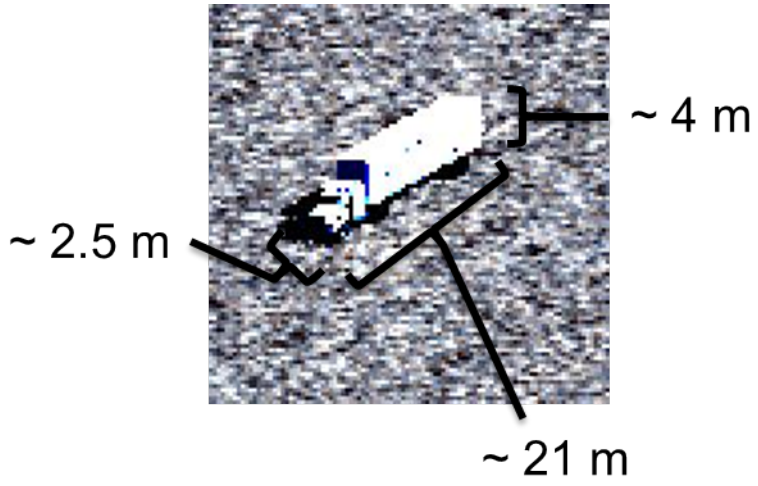

Figure 49: Relative dimensions for the Peterbilt 359 semi-truck target.

radiance above background is equal to the max pixel in the background subtracted image. The average spectral radiance levels of the background images are shown in Fig 52.

Generating synthetic imagery allows for evaluating the effects of imaging platform resolution on observed peak radiance above background. To this end, images are down sampled from the base 0.25 m GSD to lower resolutions. The resolutions of interest are shown in Table 20. Peak radiance above background was computed at each image resolution. Along with studying the effects of resolution are the effects of so-called pixel phasing. Pixel phasing refers to the alignment of sensor pixels on the physical target. The fraction of a pixel that is composed of the target vehicle has

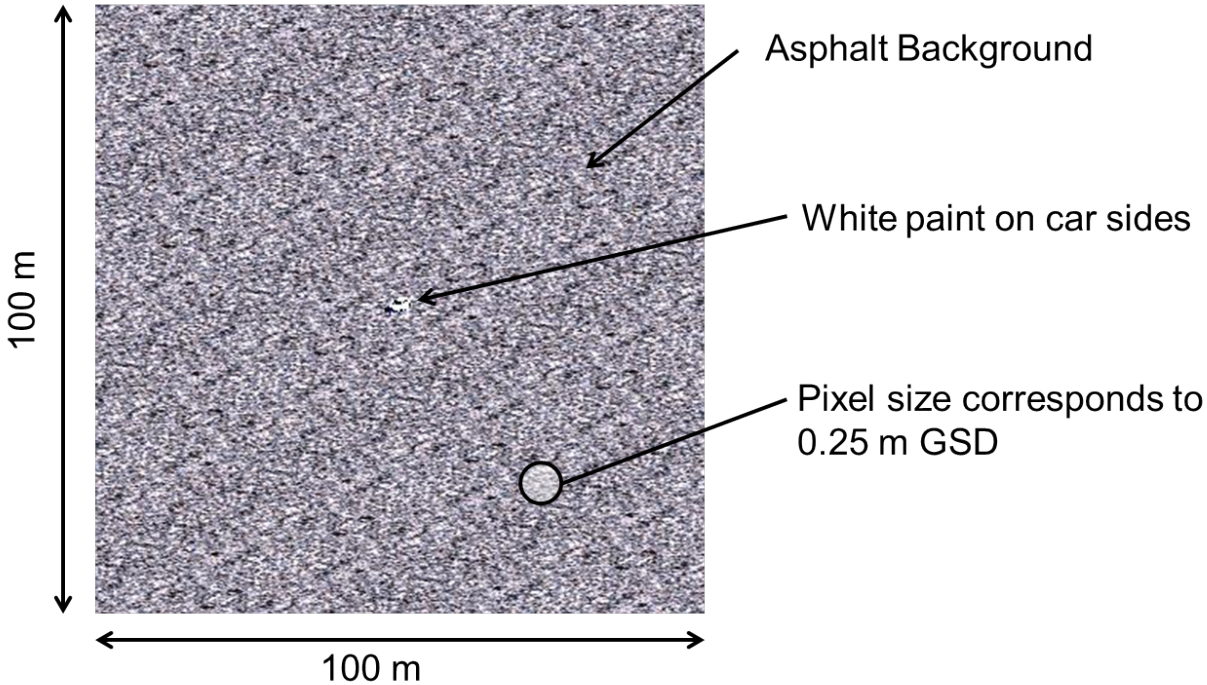

Figure 50: Simulation setup for the Audi A3 car target.

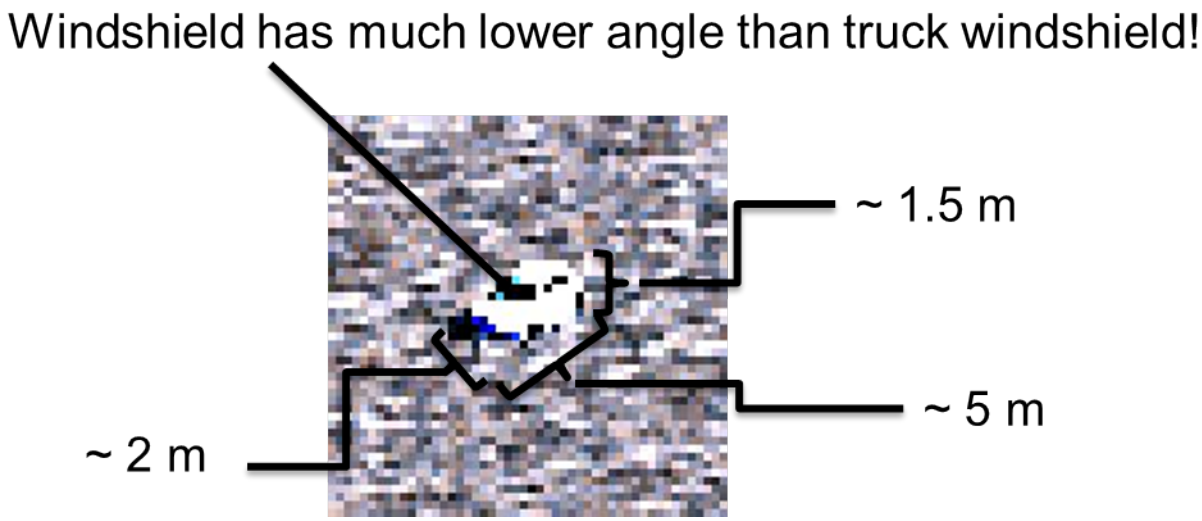

Figure 51: Relative dimensions for the Audi A3 car target.

a drastic impact on the pixel radiance observed by the sensor. For instance, if an entire pixel is composed of the target vehicle its radiance value will be very high. However, if the target is split in half between two pixels, those two pixels will have reduced radiance and the peak radiance will be much lower (only the highest radiance pixel is used). For an illustration of this effect see Fig 53.

Simulations were evaluated with quarter pixel phasing: pixels are shifted a quarter pixel length at a time, left to right and up and down, for a total of 16 positions per pixel. Half pixel phasing is used for 0.5 m GSD resolution and no pixel phasing is employed for 0.25 m GSD since simulation

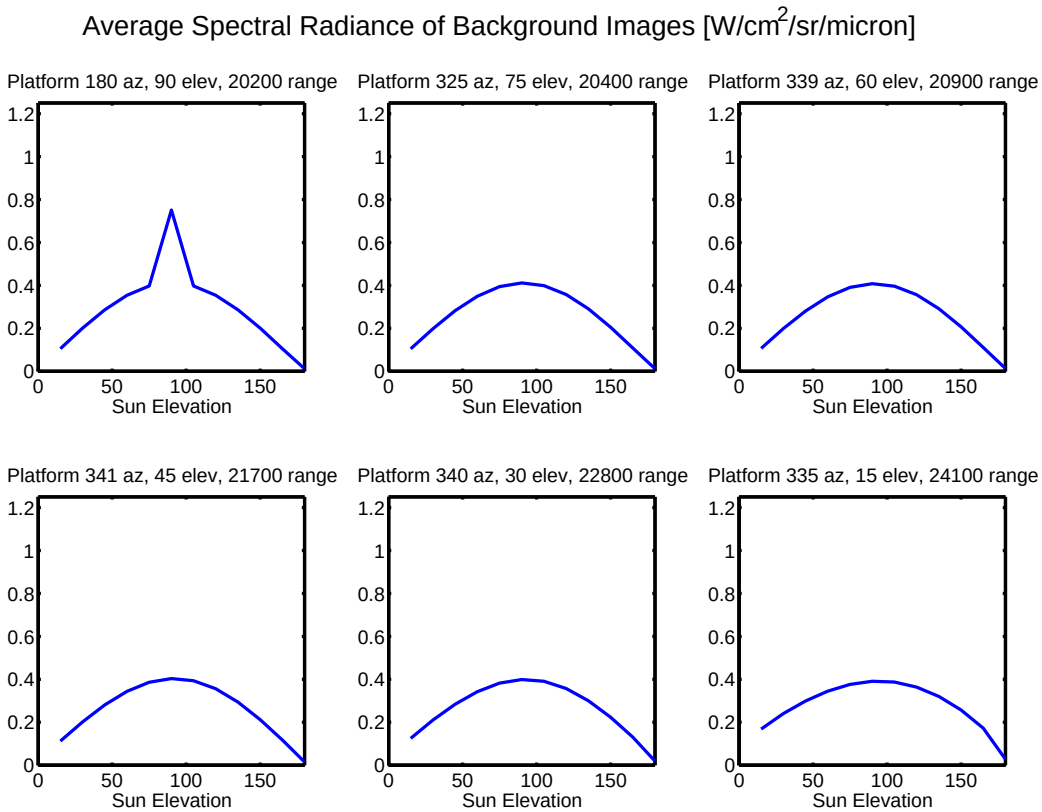

Figure 52: Average spectral radiance levels of background images. Each subplot represents a different platform viewing position. The y-axis is spectral radiance in units of $[W/cm^2/sr/micron]$. The x-axis is sun elevation angle in [°].

images are captured at a base 0.25 m GSD resolution.

Table 20: Ground sampling distances of interest to study the effects of resolution on peak radiance above background.

GSD [m]
0.25 (base simulation resolution)
0.50
1.00
2.00
4.00
8.00

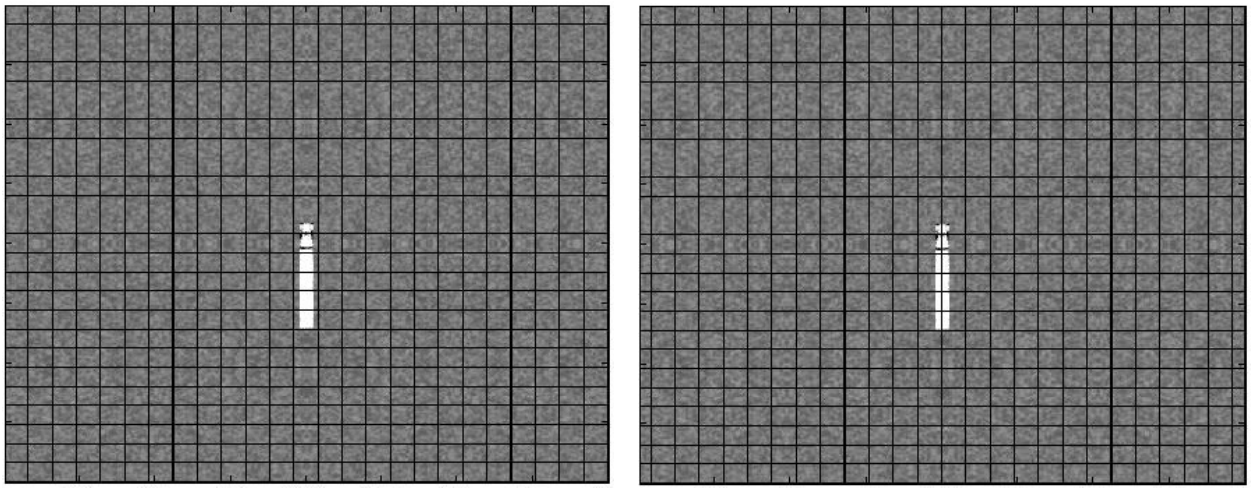

Figure 53: Pixel phasing refers to how the pixels align on the target and can have a drastic impact on observed peak radiance above background.

10.4 Results

Mean and standard deviation of peak radiance above background were computed across pixel phases for each platform position, sun elevation, target azimuth, band, and platform resolution. Standard deviation was normalized to percentage of mean. Normalized standard deviation serves as a measure of the effect of pixel phasing on peak radiance above background. The resulting data is six dimensional (peak radiance × platform position × sun elevation × target azimuth × spectral band × resolution) for each statistic.

The data dimensionality is reduced for visualization purposes by considering only the widest spectral band (0.42 - 1.10 microns) and fixing either the sun elevation or target azimuth in a given plot. Each plot contains six subplots, one for each platform view position. Within each subplot, radius represents resolution [GSD, m]. Angle represents either sun elevation or target azimuth, depending on the plot. The colored contours represent the value of the statistic (mean or normalized standard deviation) at a particular data point. Interpolation is applied to smooth the contours.

10.4.1 Peterbilt 359 Semi-Truck

Sun elevations were fixed to 15°, 90°, and 165°, and separate plots are made for each elevation. Within each subplot, rotation angle corresponds to target azimuth [°]. Since all platform locations except the nadir case view the target from the West, these sun elevations provide a variety of viewing scenarios. At 15° sun elevation, the platforms view the side of the target in shadow. At 90° sun elevation, shadows are not cast from the target and only the top of the target is illuminated. At 165° sun elevation, the illuminated side and top of the truck are visible to the viewing platform, and the shadow is on the opposite side of the target.

Mean peak radiance above background is plotted for 90°, 15°, and 165° in Figs 54, 61, and 63, respectively (90° is shown first as it is the most intuitive case). The general trend follows intuition: as pixel size increases, mean peak radiance decreases. As pixels increase in size they consist of smaller fractions of the target and larger fractions of background. For high platform elevation angles, mean peak radiance contours are relatively uniform. A view of the contours alongside simulation images under 90° sun illumination is provided in Fig 55. When the target faces the

imaging platform, the mean is higher as the bright roof of the target presents a large surface area to the imaging platform. When the target is side-on to the imaging platform, only a small sliver of bright roof is presented to the imaging platform and pixels quickly become primarily background as pixel size increases. This causes the mean to fall off rapidly.

Normalized standard deviation of peak radiance above background is plotted for 90°, 15°, and 165° in Figs 56, 62, and 64, respectively. The general trend is inverse of mean peak radiance: as pixel size increases, normalized standard deviation also increases. However, the contours are significantly more complex than for mean peak radiance, even in the 90° platform elevation case. Note that since the original simulations are at 0.25 m GSD and there is no pixel phasing for this resolution and the normalized standard deviation is zero. The specific geometry of contours is difficult to explain without explicitly viewing the simulation images from that geometry. Normalized standard deviation for the 90° platform elevation is shown alongside simulation images in Figs 57 and 58. When the target is oriented diagonally to the pixels, the roof is evenly divided across pixel phases. This leads to low standard deviation. When the target is oriented parallel to the pixels it alternates between being completed contained in a single pixel and evenly split between multiple pixels. This leads to high standard deviation. Normalized standard deviation for the 15° platform elevation is shown alongside simulation images in Figs 59 and 60. In the lower platform elevation, a large roof target allows for good pixel coverage and low standard deviation. A small roof target as in the side-on case is easily split among pixels and yields a high standard deviation.

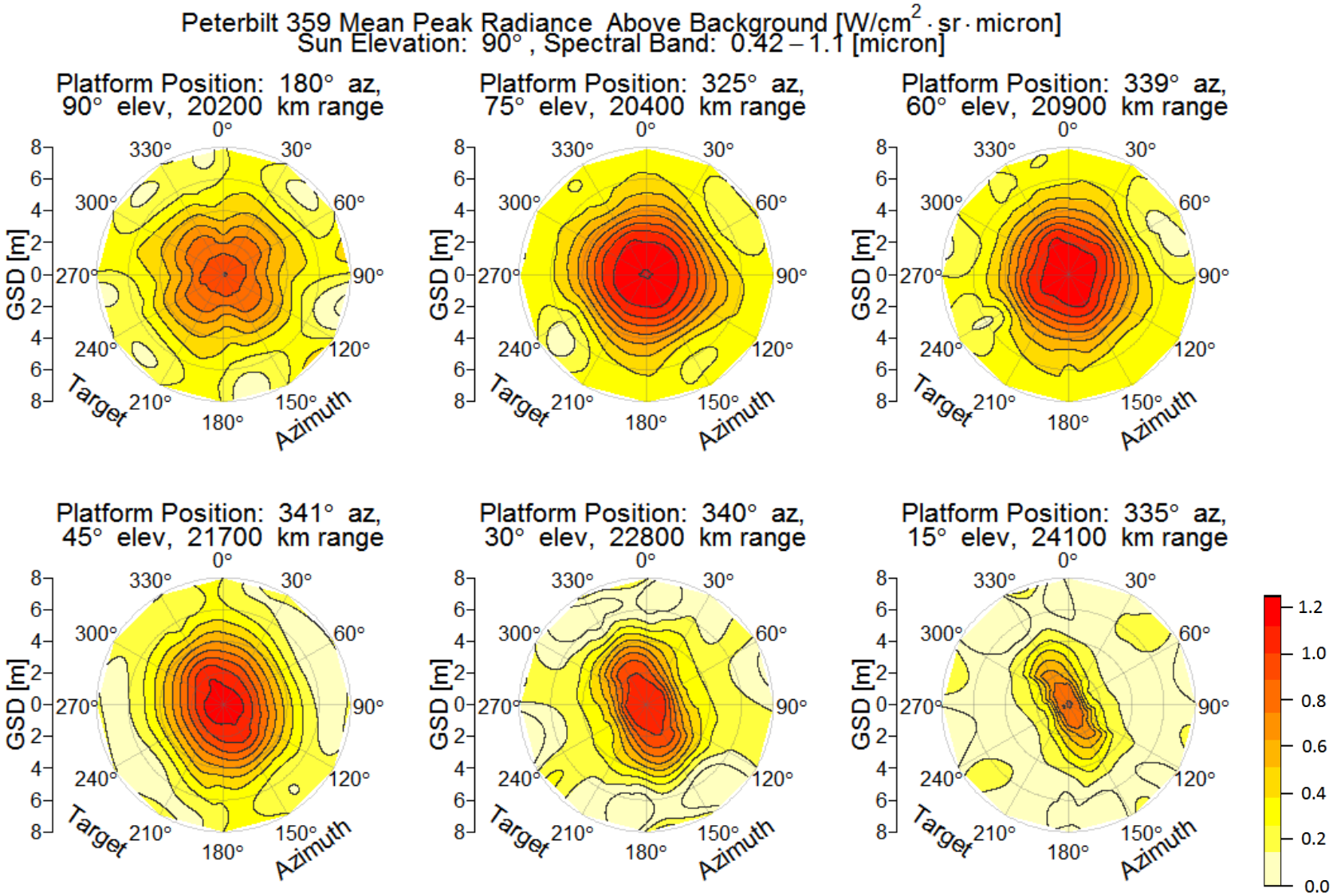

Figure 54: Mean peak radiance above background for the Peterbilt 359 semi-truck target with the sun fixed at 90° elevation.

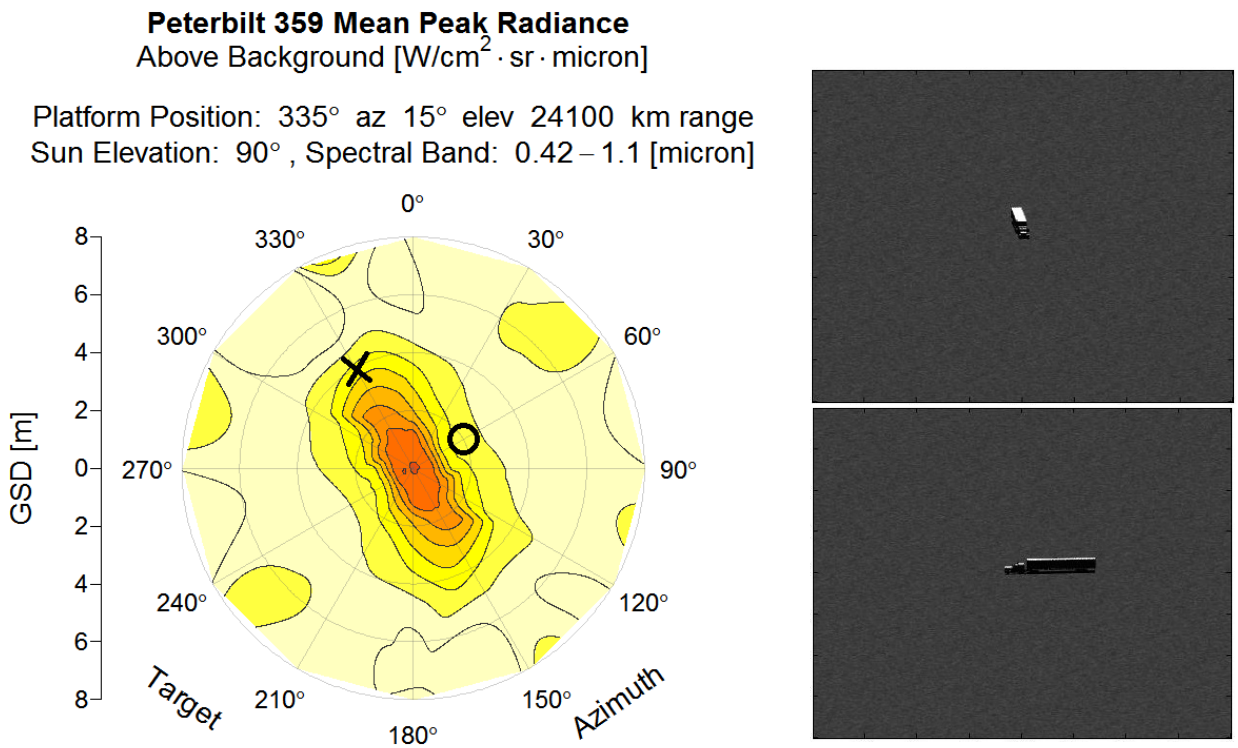

Figure 55: Mean peak radiance above background for the semi-truck target with the sun at 90° elevation and platform at 15° elevation. The top simulation image corresponds to **X**. At this target azimuth, there is a pixel that is mostly truck roof for most GSD's, causing mean to fall off slowly. The bottom simulation plot corresponds to **O**. At this target azimuth, roof pixels are mixed with the side of the truck and background for all but the lowest GSD's, causing the mean to fall off quickly as pixel size increases.

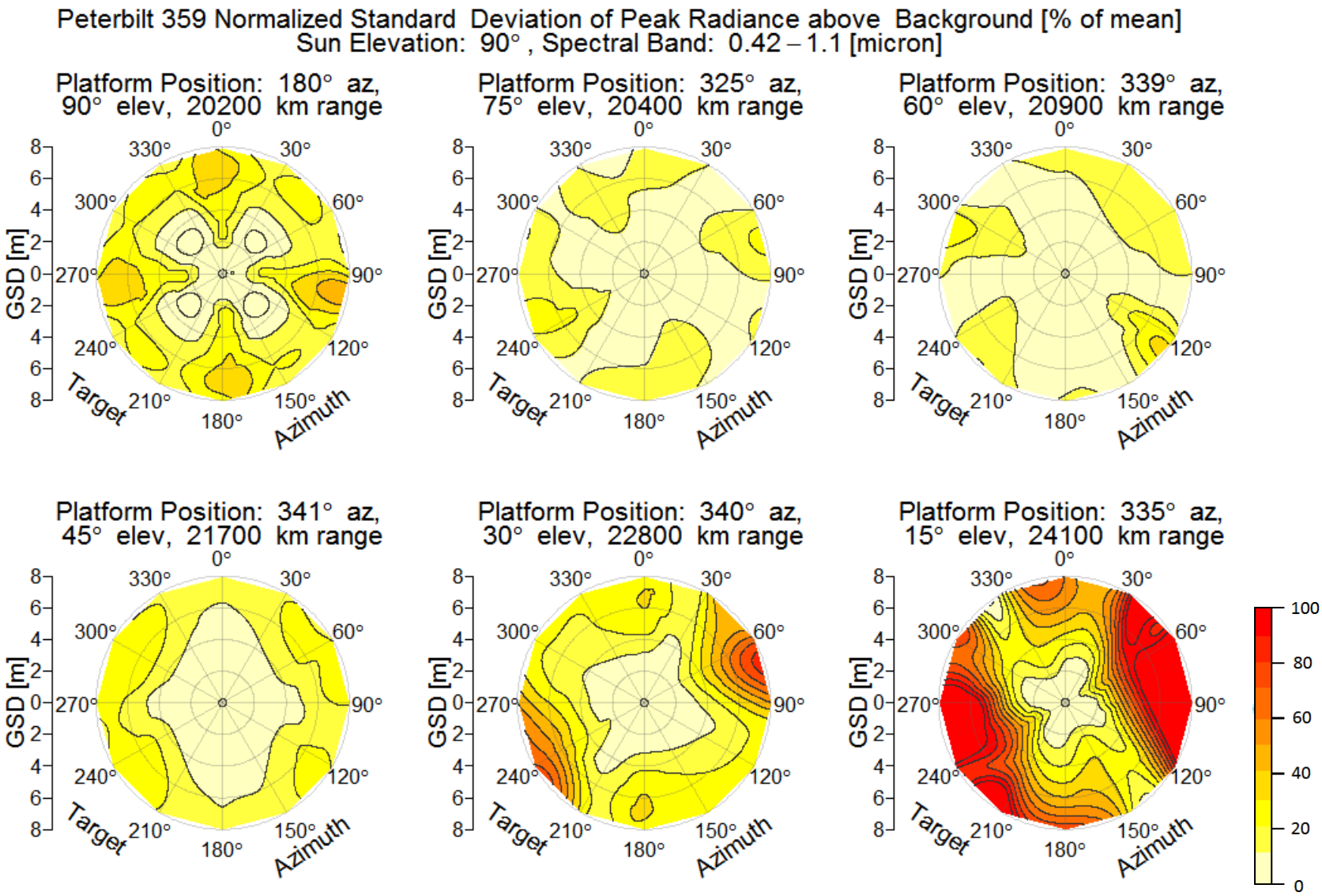

Figure 56: Normalized standard deviation of peak radiance above background for the Peterbilt 359 semi-truck target with the sun fixed at 90° elevation.

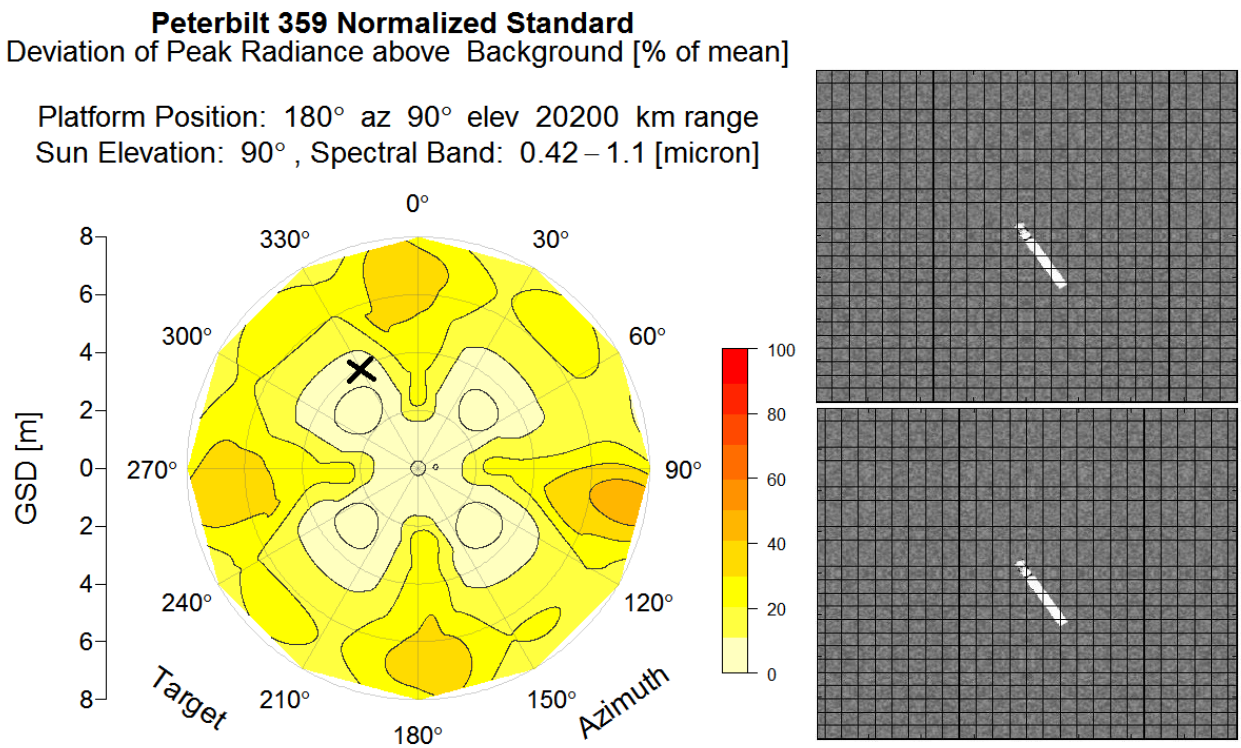

Figure 57: Normalized standard deviation at 90° platform elevation, 90° sun. **X**: Pixels are shifted to the right in the bottom simulation image. The roof is fairly evenly divided among pixels in both pixel phases. This yields a low standard deviation, as seen in the contour plot on the left.

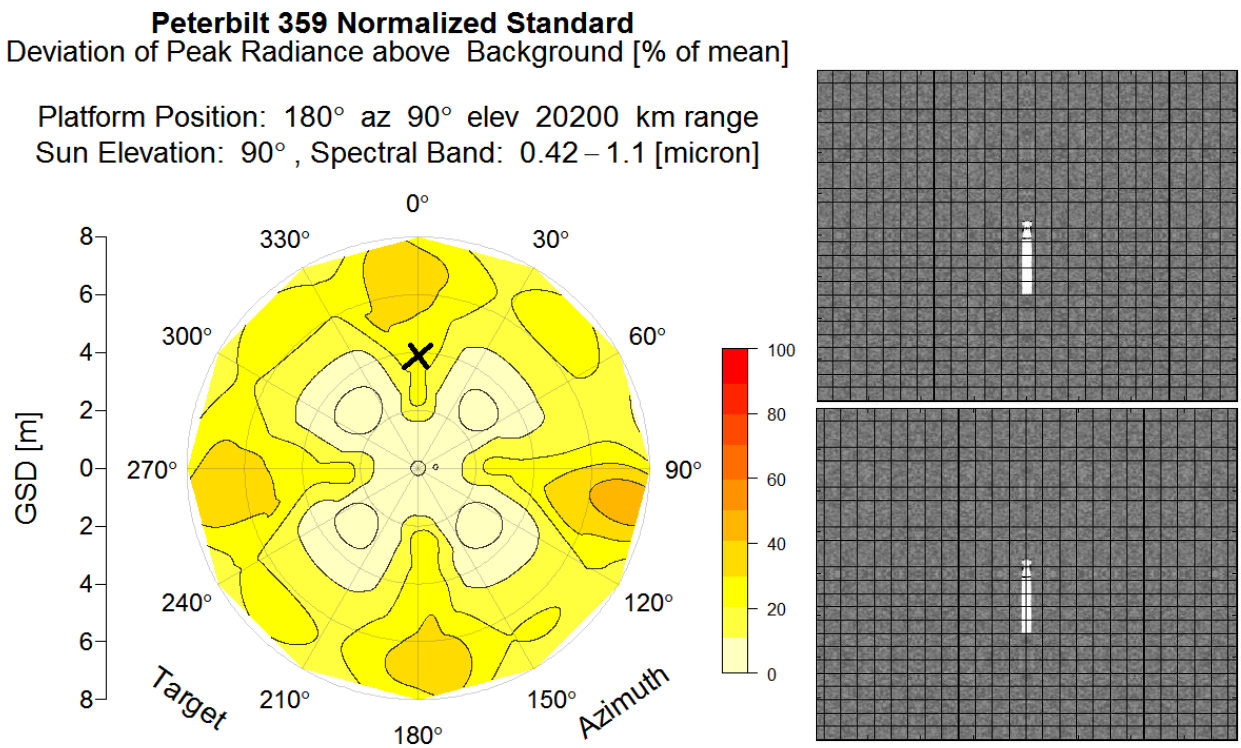

Figure 58: Normalized standard deviation at 90° platform elevation, 90° sun. **X**: Pixels are shifted to the right in the bottom simulation image. Pixel phasing has a drastic effect: the roof goes from being entirely contained in a pixel to being split between two pixels. This yields a large standard deviation, as seen in the contour plot on the left.

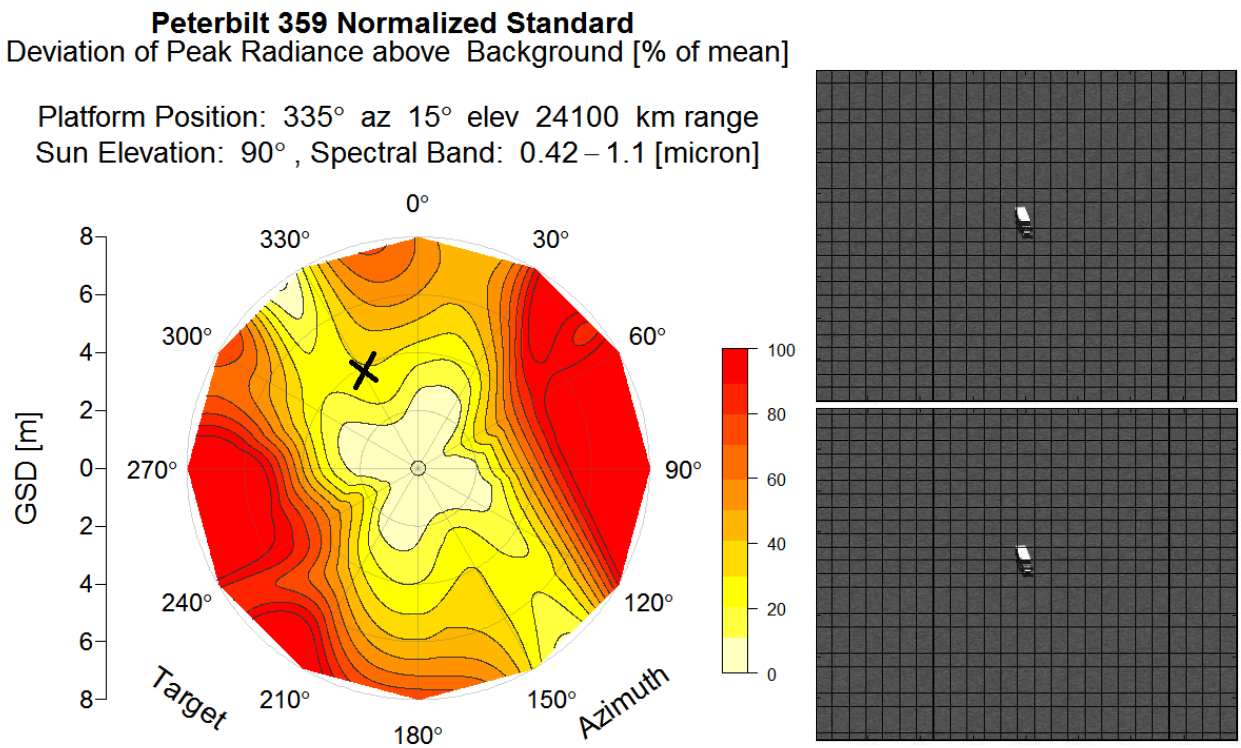

Figure 59: Normalized standard deviation at 15° platform elevation, 90° sun. **X**: Pixels are shifted down in the bottom simulation image. As can be seen from the images, the target is presented with a large roof area that covers most of the peak pixels. This yields lower standard deviation as the amount of roof in the pixels is consistent across phases.

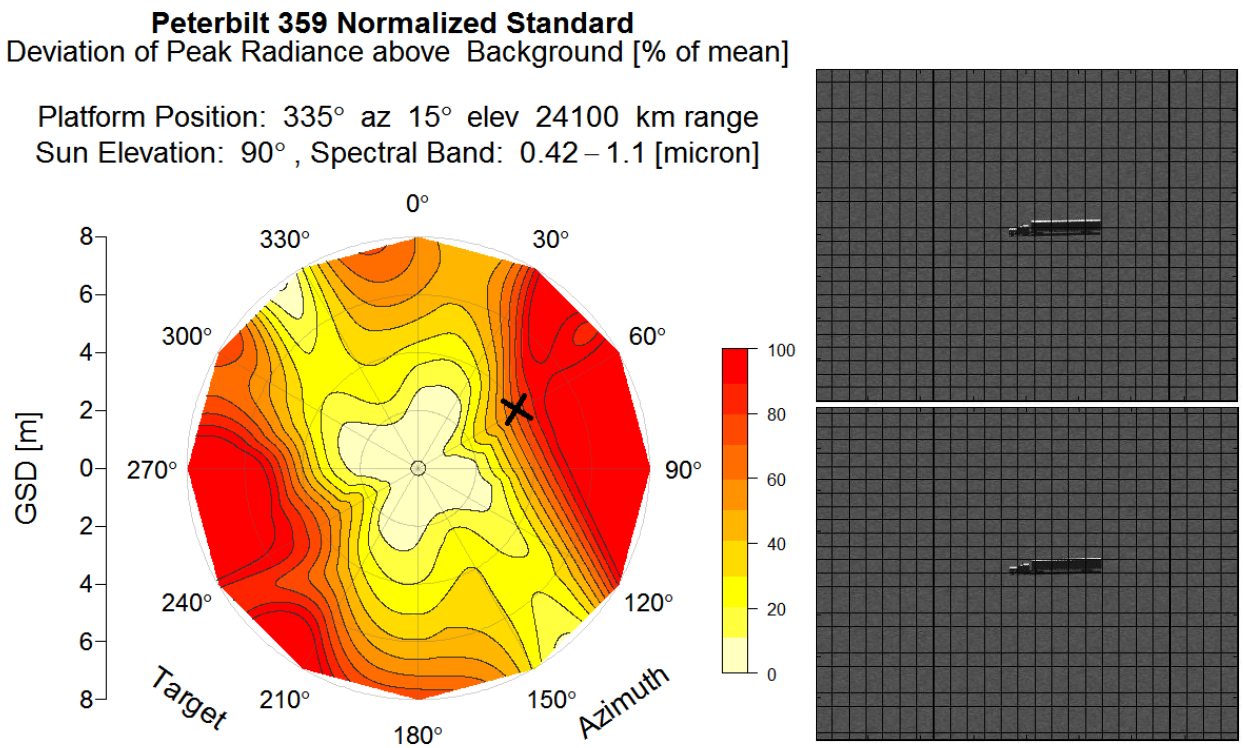

Figure 60: Normalized standard deviation at 15° platform elevation, 90° sun. **X**: Pixels are shifted down in the bottom simulation image. As can be seen from the images, only a small portion of the target roof is presented to the imaging platform. The amount of roof in a pixel varies drastically as pixels are phased and the small roof area yields a small mean peak. This leads to a high normalized standard deviation.

10.4.2 Audi A3 Car

The second target simulated was an Audi A3 automobile. Sun elevations were fixed to the same positions as for the semi-truck target. Mean peak radiance above background is plotted for 90°, 15°, and 165° in Figs 66, 68, and 70, respectively. General trends are the same as in the semi-truck simulations with two key exceptions. The first is that the car is significantly smaller than the semi-truck. This means that there is significantly less target per pixel as pixel size increases, causing the mean to fall off much faster for the car than for the semi-truck. The second key difference is that the windshield of the car is at a much lower angle relative to the ground plane. This results in the sun's reflection off of the windshield being directly observable by the imaging platform. This is illustrated in Fig 65. When the sun is directly observed, the peak radiance is roughly 4× that of the white paint on the car (and by extension, the semi-truck). However, the contour plots are scaled to the same levels as those of the semi-truck, so as to preserve detail in the plots. This means that for configurations in which the sun reflection is visible, the contours will be saturated to their max level.

Normalized standard deviation of peak radiance above background is plotted for 90°, 15°, and 165° in Figs 67, 69, and 67, respectively. As with the semi-truck, normalized standard deviation for the car target generally increases as pixel size increases. However, the normalized standard deviation is generally greater for the car than for the truck. This follows intuition: the car covers significantly less pixel fractions than the truck, causing the standard deviation to be much greater when normalized against mean. The contours are more complex for the car due to the smaller and less boxy geometry of the car.

10.5 General Trends

For both the car and the truck, mean peak radiance above background generally falls off as resolution decreases. Mean peak radiance of the car falls off significantly faster than the truck as it is a much smaller target. The lower windshield angle of the car allows for the possibility of sun glints that can be seen by the imaging platform. When sun glints are present, the car has significantly higher mean peak radiance. A side by side comparison of mean peak radiance above background for the two targets is shown in Fig 72.

The car and the truck both experience increased normalized standard deviation of peak radiance above background as pixel size increases. The car generally has greater standard deviation: it presents a smaller target and is easily split among multiple pixels such that there are no pixels composed primarily of the car. Both targets have similar contour shape based on how the target vehicles aligns with the pixels. A side by side comparison of the targets is shown in Fig 73.

In general, increasing GSD reduces peak radiance above background. Increasing GSD increases deviation from pixel phasing effects, as expected. Decreasing imaging platform elevation increases deviation and the impact of pixel phase effects. Decreasing sun elevation leads to non-uniformity in peak radiance contours and increases deviation from pixel phasing. For the truck target: a 2m GSD keeps deviation due to pixel phase effects to less than 10% in most cases. A 4m GSD keeps deviation due to pixel phases effects to less than 20% in most cases. For the car target: a 1m GSD keeps deviation due to pixel phase effects to less than 20% in most cases. A 2m GSD keeps deviation due to pixel phases effects to less than 30% in most cases.

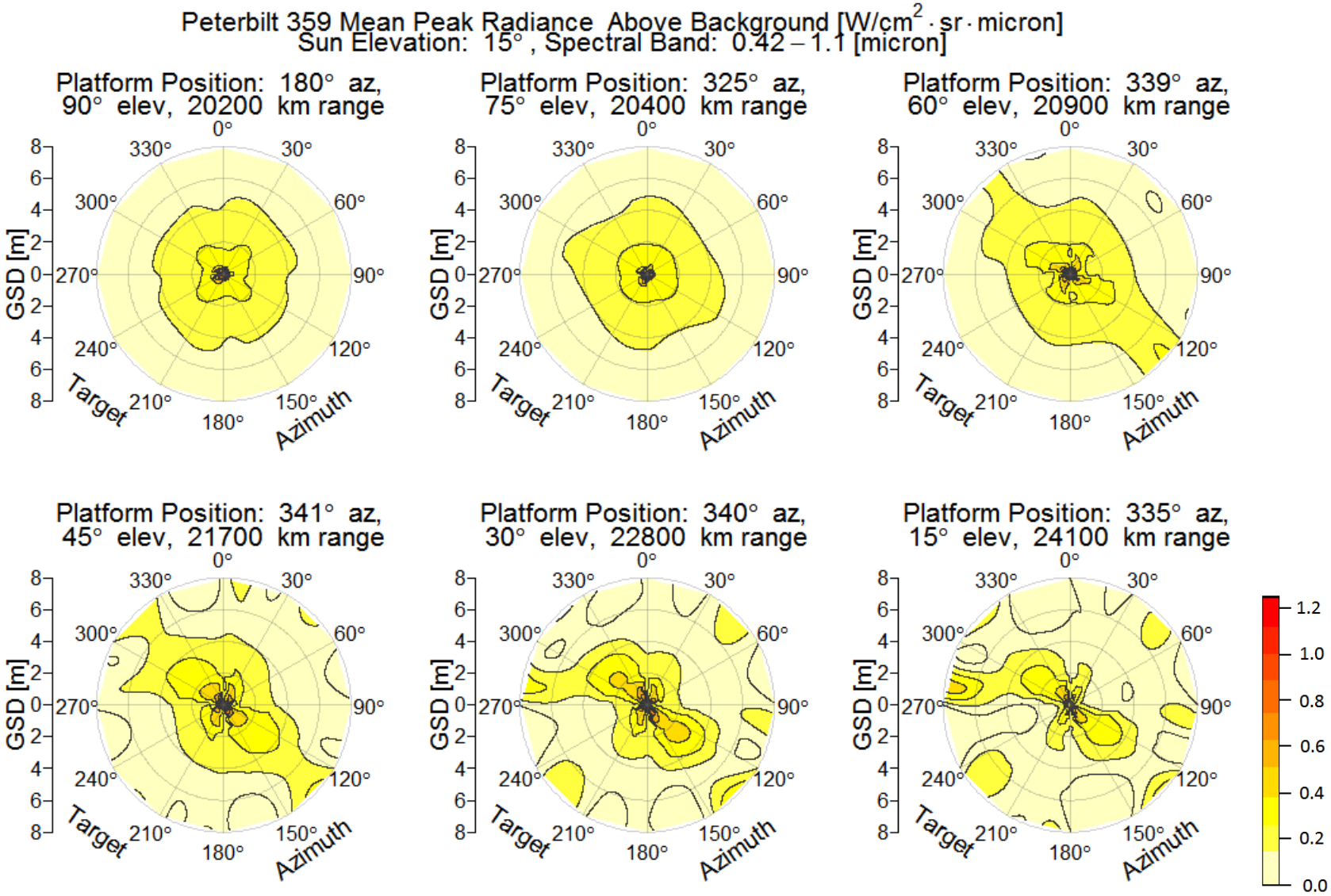

Figure 61: Mean peak radiance above background for the Peterbilt 359 semi-truck target with the sun fixed at 15° elevation.

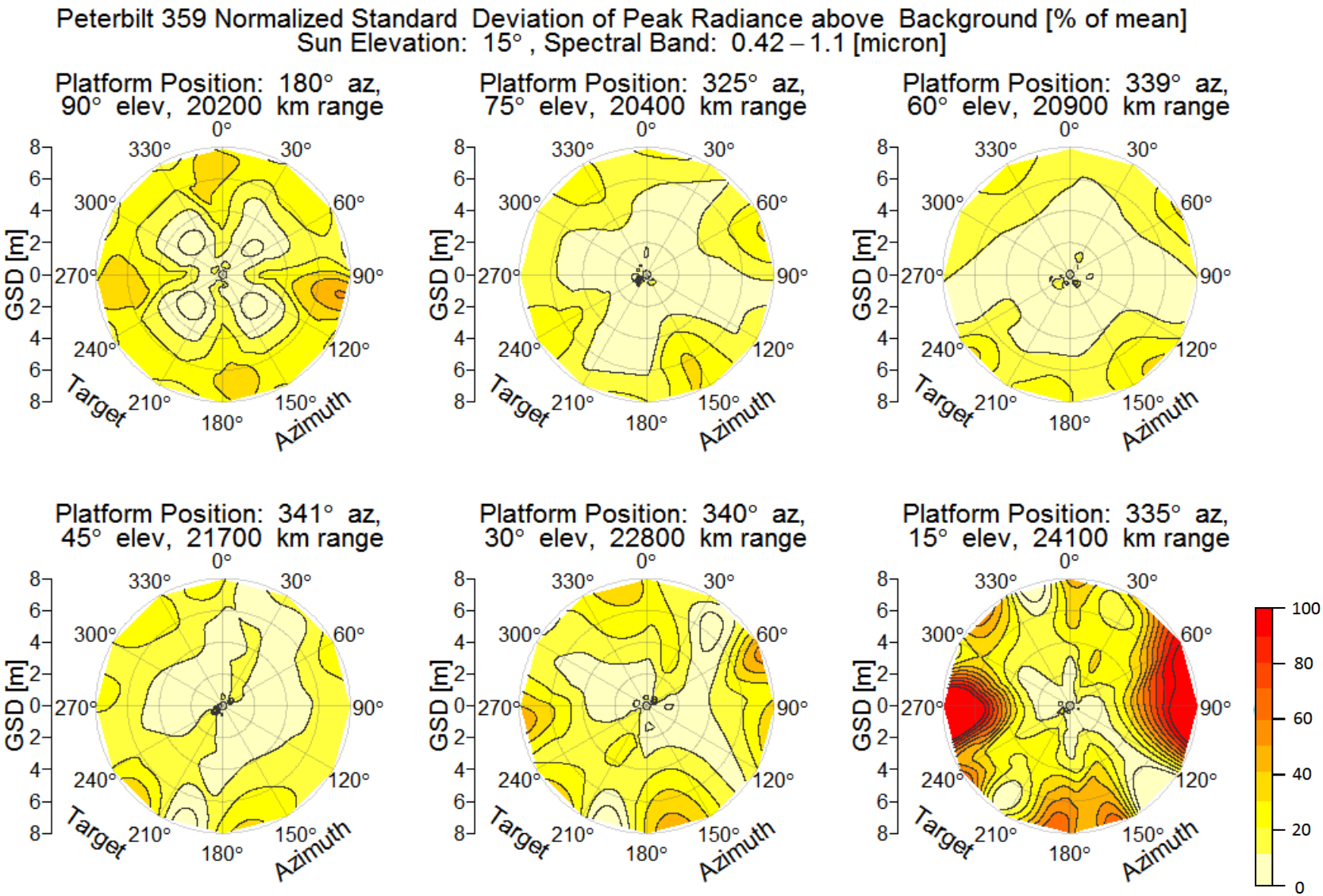

Figure 62: Normalized standard deviation of peak radiance above background for the Peterbilt 359 semi-truck target with the sun fixed at 15° elevation.

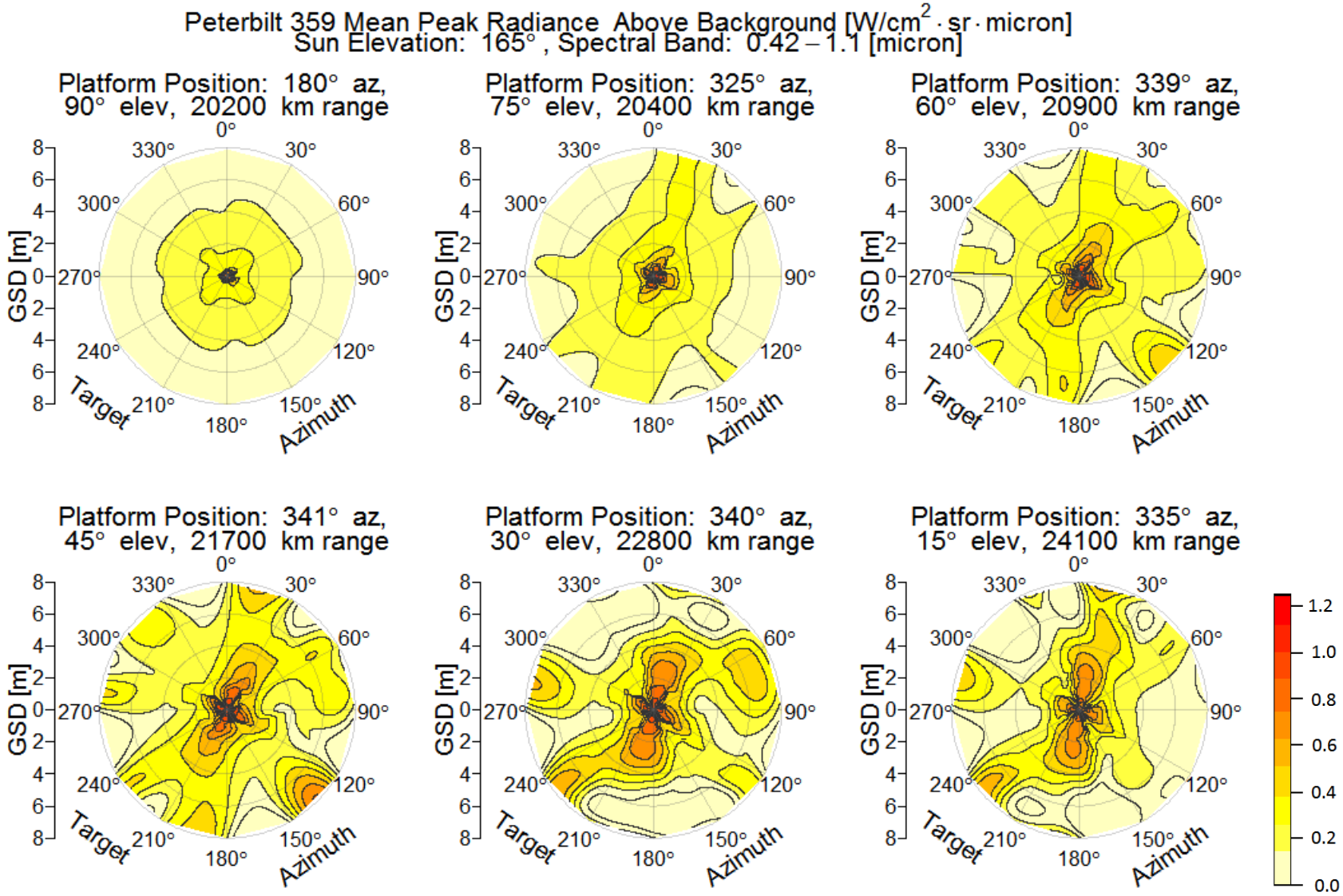

Figure 63: Mean peak radiance above background for the Peterbilt 359 semi-truck target with the sun fixed at 165° elevation.

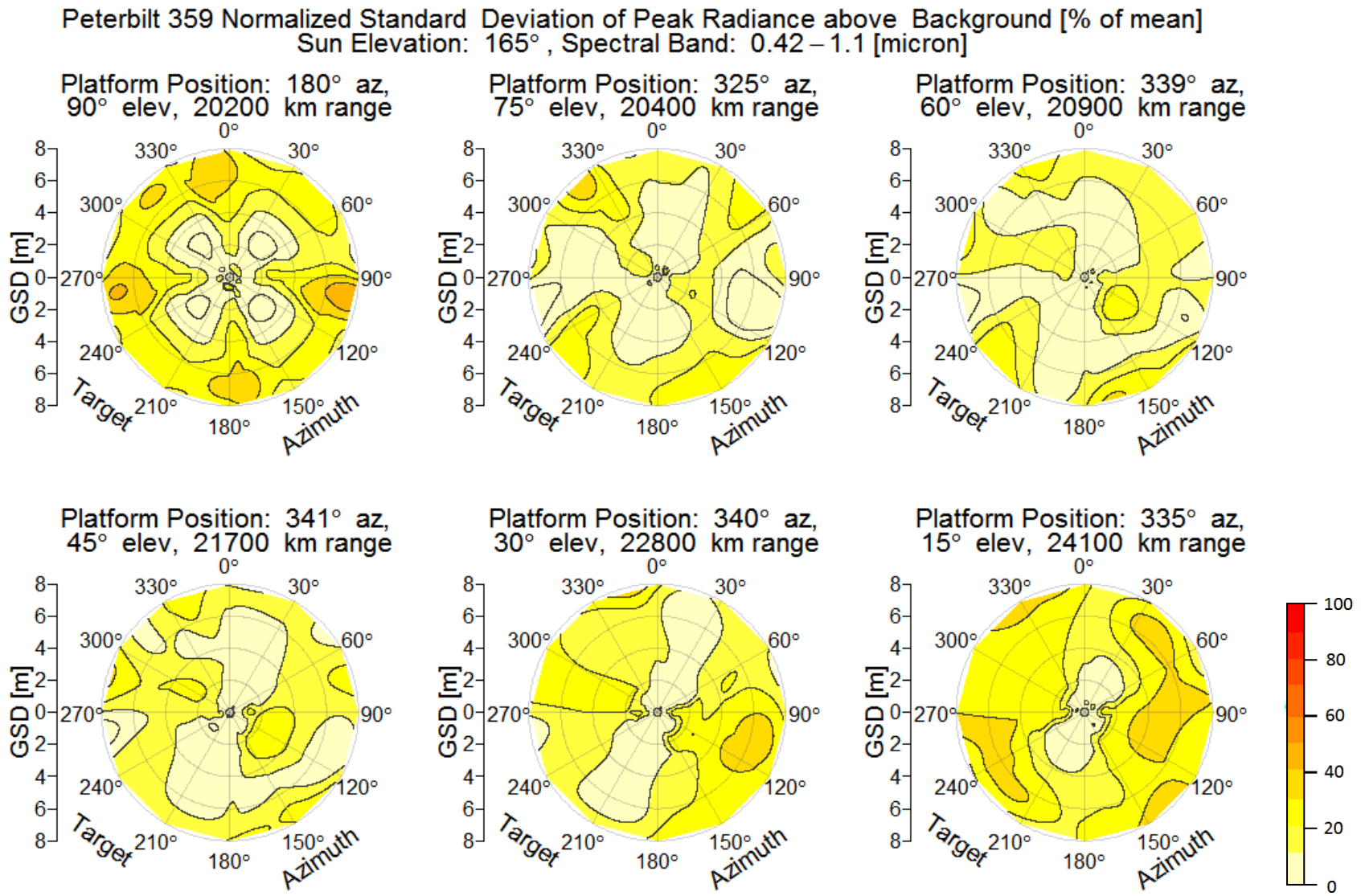

Figure 64: Normalized standard deviation of peak radiance above background for the Peterbilt 359 semi-truck target with the sun fixed at 165° elevation.

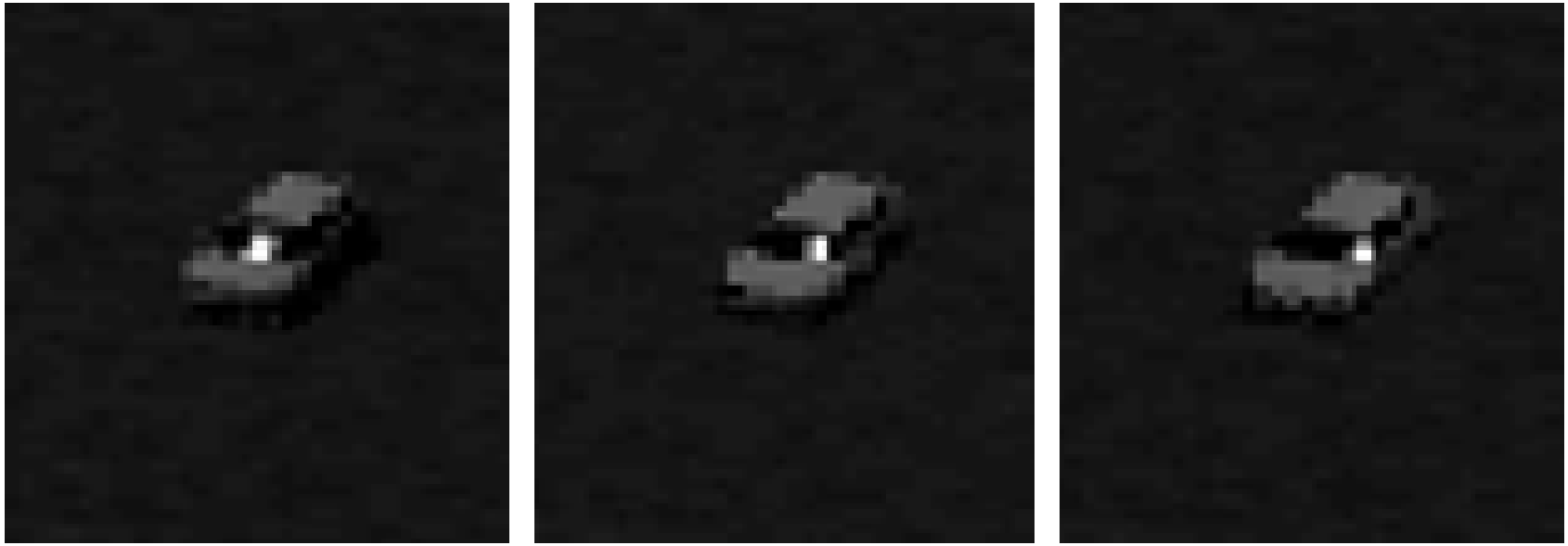

Figure 65: The lower windshield angle of the Audi car target allows for the sun's reflection off of the windshield to be directly observed by the imaging platform under certain view geometries.

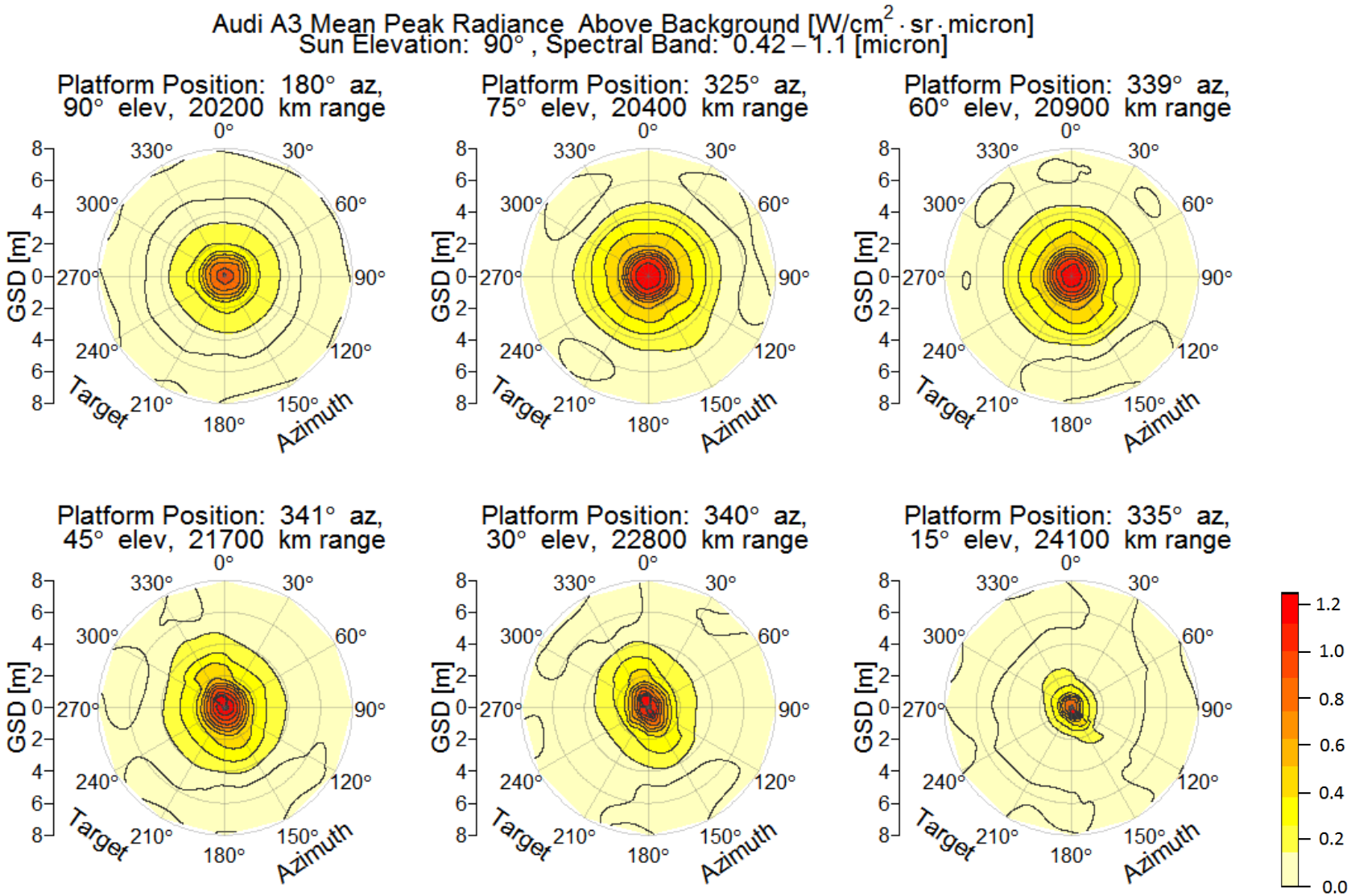

Figure 66: Mean peak radiance above background for the Audi A3 car target with the sun fixed at 90° elevation.

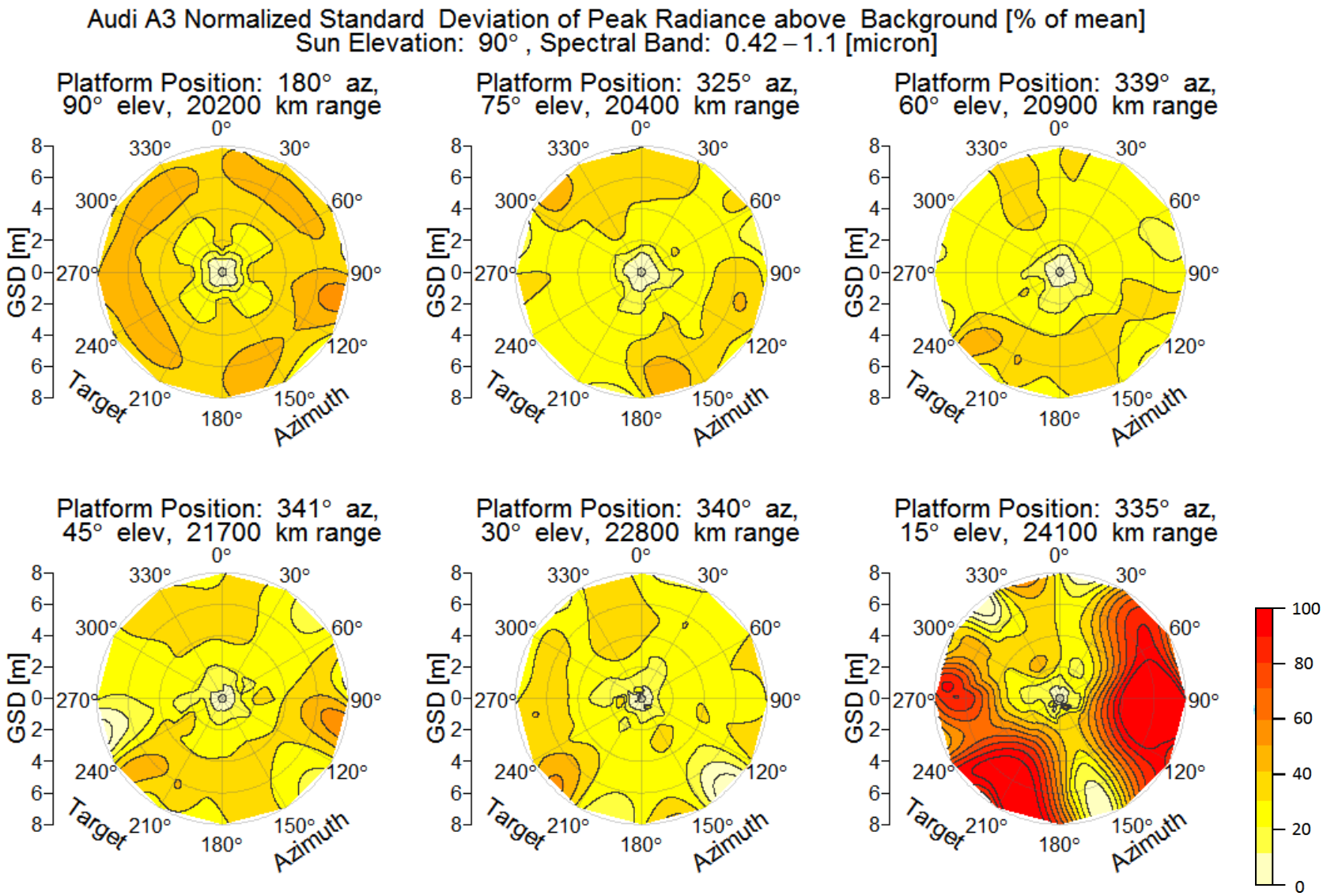

Figure 67: Normalized standard deviation of peak radiance above background for the Audi A3 car target with the sun fixed at 90° elevation.

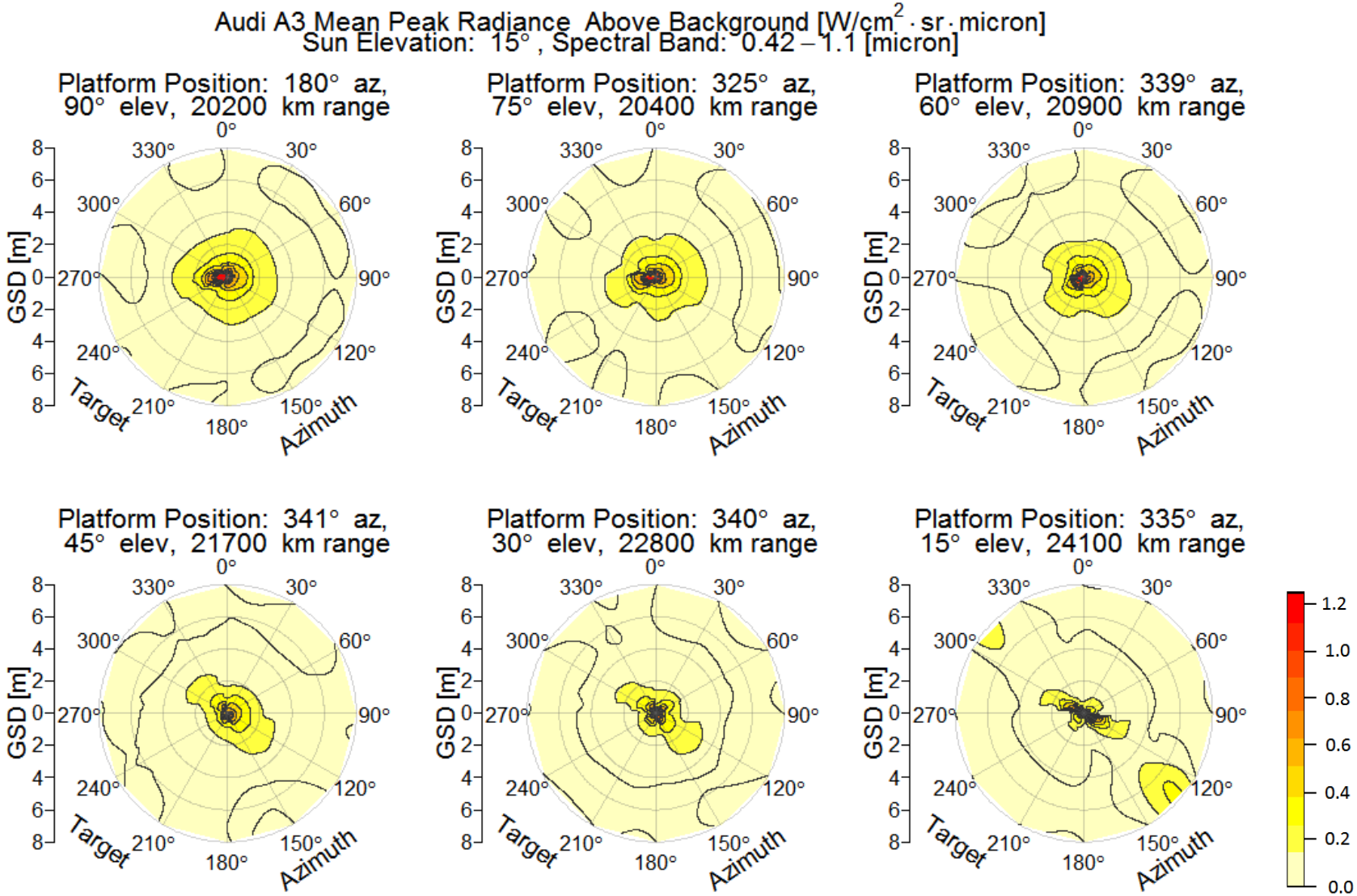

Figure 68: Mean peak radiance above background for the Audi A3 car target with the sun fixed at 15° elevation.

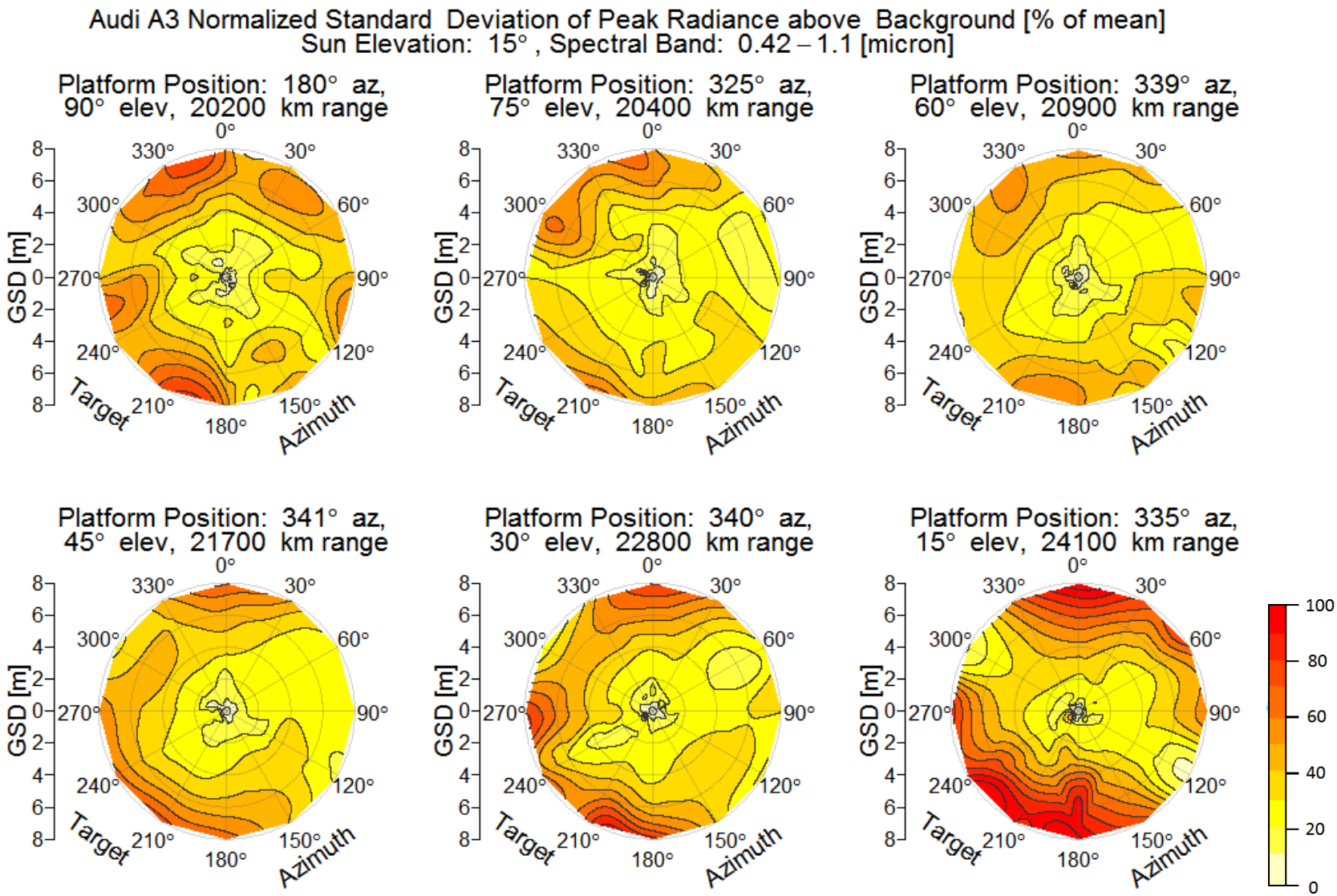

Figure 69: Normalized standard deviation of peak radiance above background for the Audi A3 car target with the sun fixed at 15° elevation.

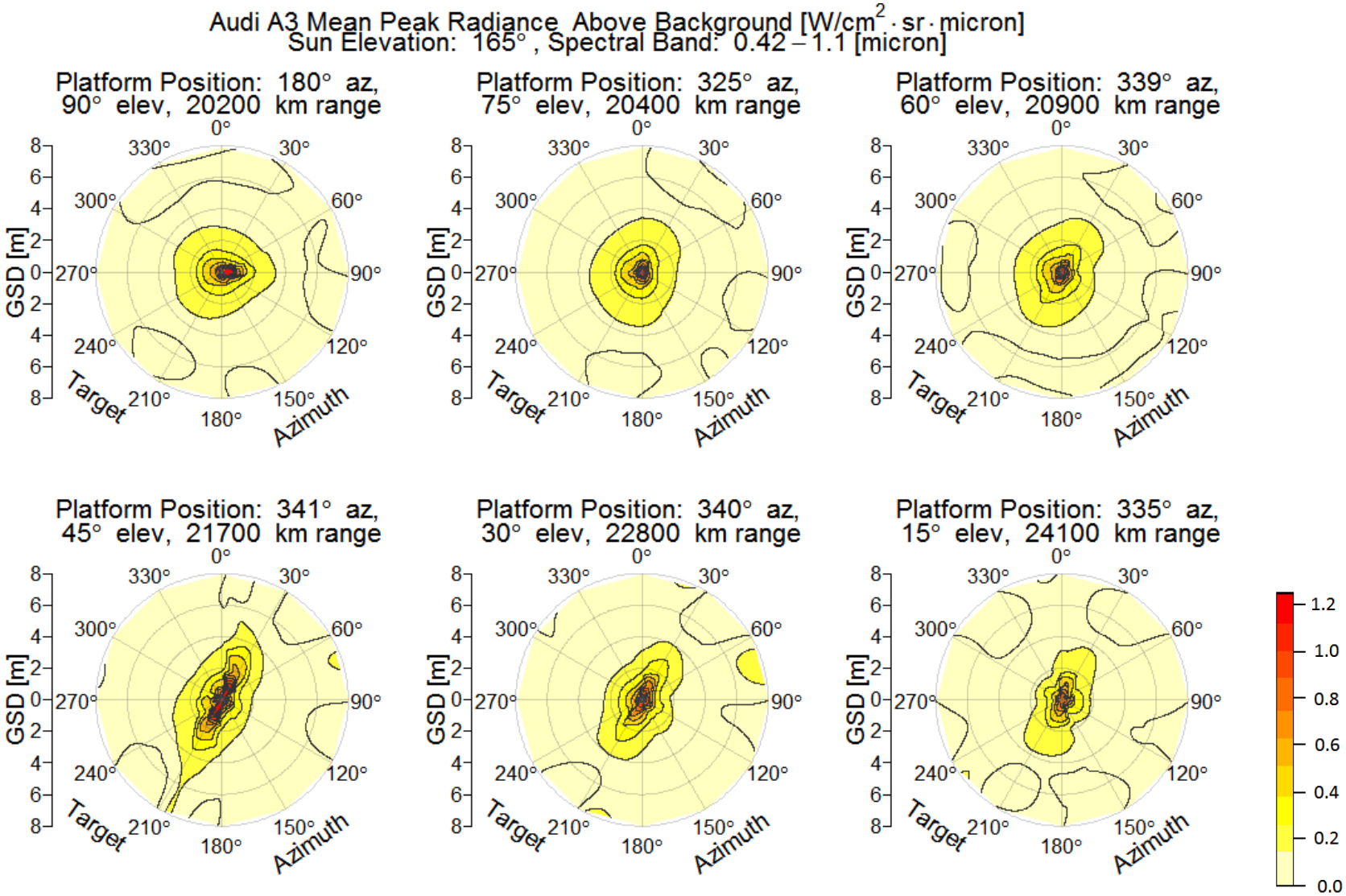

Figure 70: Mean peak radiance above background for the Audi A3 car target with the sun fixed at 165° elevation.

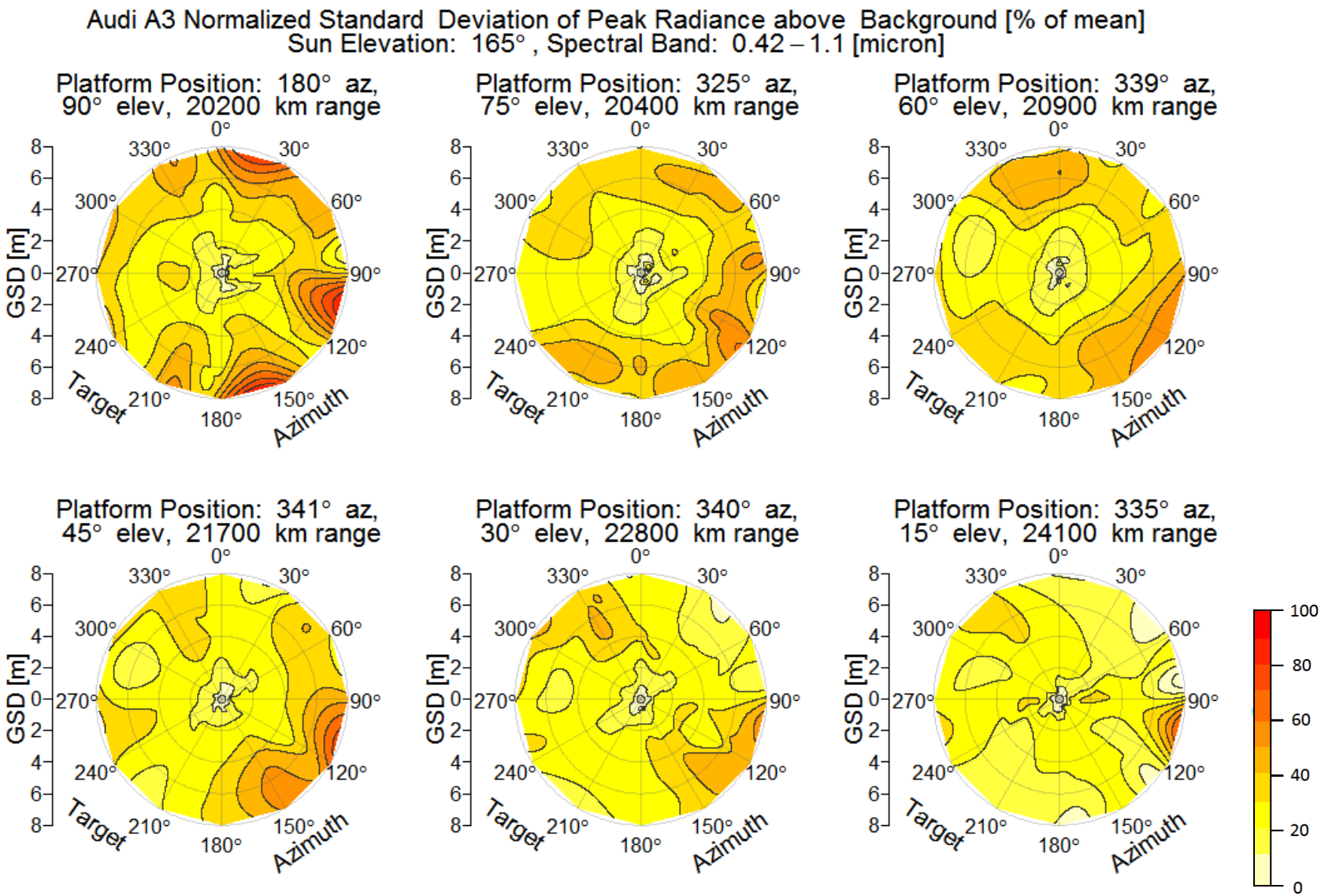

Figure 71: Normalized standard deviation of peak radiance above background for the Audi A3 car target with the sun fixed at 165° elevation.

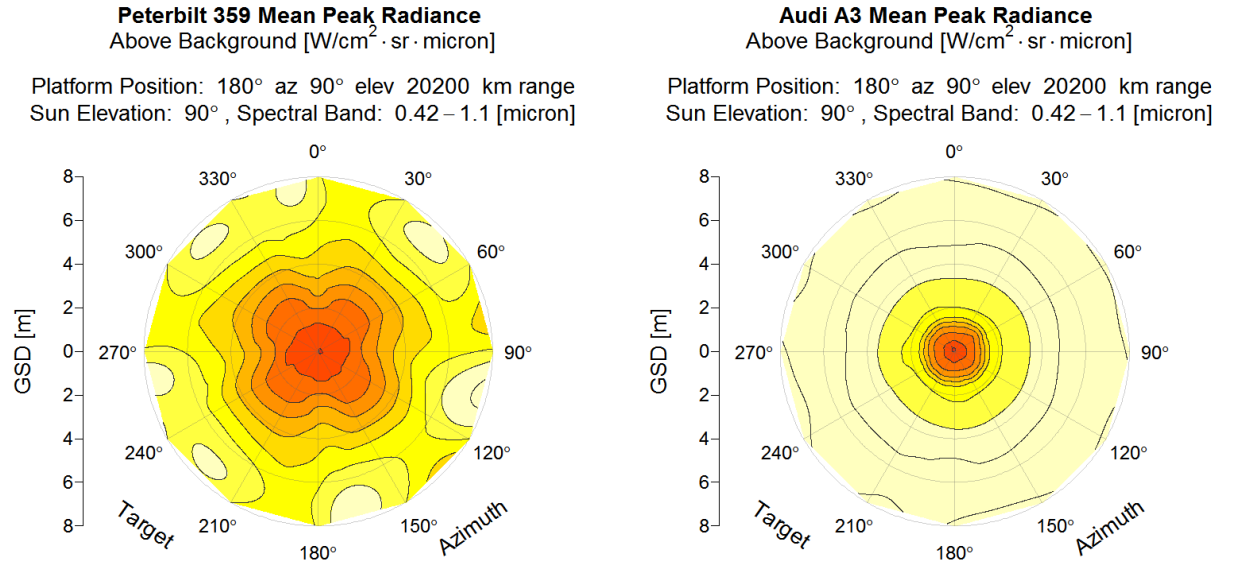

Figure 72: Pictured left: semi truck target. Pictured Right: car target. Mean peak radiance from 90° platform and sun elevation. The car falls off much faster as it is a smaller target than the semi truck and therefore takes up smaller fractions of pixels as resolution decreases.

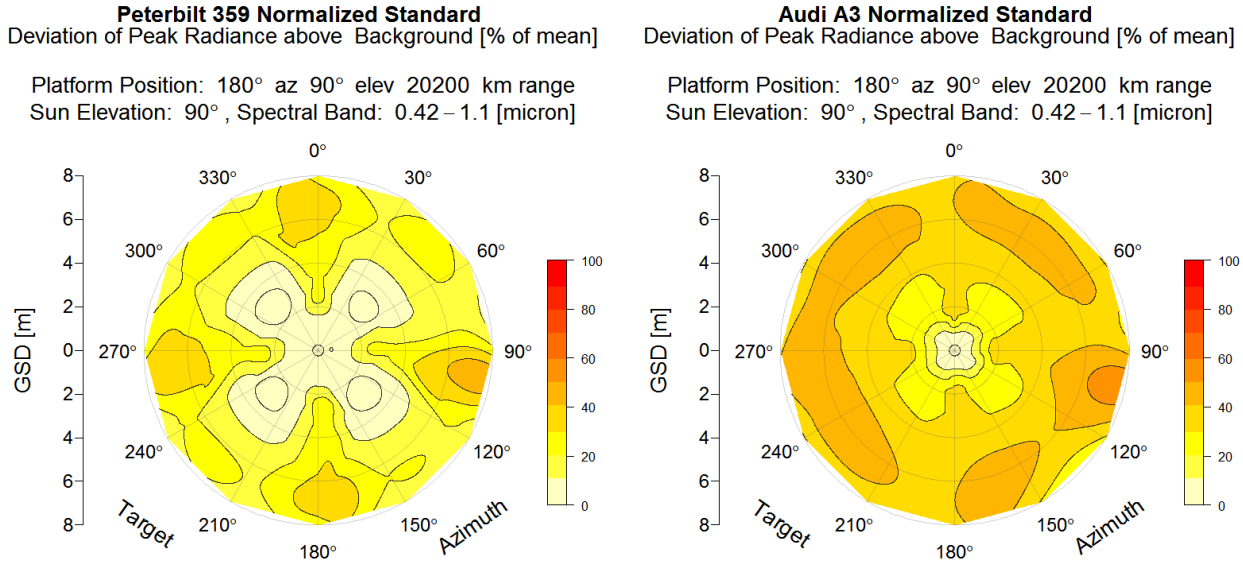

Figure 73: Pictured left: semi truck target. Pictured Right: car target. Normalized standard deviation of peak radiance from 90° platform and sun elevation. Both targets have similar contour shape based on how the target vehicles aligns with the pixels, and the car generally has greater standard deviation.

11 Summary and Conclusions

The goal of this effort was to evaluate several potential tracking algorithms that scale well to large-scale problems. Urban tracking of automobiles was used to focus the research since 100's to 1000's of vehicles may be present in a single frame. Another goal was to investigate the phenomenology of vehicle targets to gain insight into better detection schemes. The following algorithms were considered: the probabilistic multi-hypothesis tracker (PMHT); a RANSAC algorithm; a Markov chain Monte Carlo data association algorithm; tracklet inference from factor graphs; and a proximity tracker. The PMHT algorithm suffered from convergence problems, and was very sensitive to the initial conditions, so it was not considered for further analysis. The remaining algorithms were tested on a combination of simulated and real traffic detection data. A common set of metrics was used to compare the performance of the algorithms. While each approach had strengths and weaknesses, a common difficulty was handling a large number of broken tracks. Broken tracks often occur when a vehicle is occluded for some period of time, or when vehicles get close enough to be merged into a single detection. Using additional information to aide in the detection-to-track association, like vehicle color or shape, should help decrease the number of broken tracks. This was tested for the factor graph inference algorithm by incorporating vehicle color and yielded improved results.

The second part of this effort focused on studying the phenomenology of vehicle targets. Simulated signatures of a Peterbilt 359 semi truck and an Audi A3 car were studied to gain insight into the relationships between different sensing parameters. For both the car and the truck, mean peak radiance above background generally falls off as resolution decreases. Mean peak radiance of the car falls off significantly faster than the truck as it is a much smaller target. The lower windshield angle of the car allows for the possibility of sun glints that can be seen by the imaging platform. When sun glints are present, the car has significantly higher mean peak radiance. The car and the truck both experience increased normalized standard deviation of peak radiance above background as pixel size increases. The car generally has greater standard deviation: it presents a smaller target and is easily split among multiple pixels such that there are no pixels composed primarily of the car. Both targets have similar contour shape based on how the target vehicles aligns with the pixels. In general, increasing ground sample distance (GSD) reduces peak radiance above background. Increasing GSD increases deviation from pixel phasing effects, as expected. Decreasing imaging platform elevation increases deviation and the impact of pixel phase effects. Decreasing sun elevation leads to non-uniformity in peak radiance contours and increases deviation from pixel phasing. For the truck target: a 2m GSD keeps deviation due to pixel phase effects to less than 10% in most cases. A 4m GSD keeps deviation due to pixel phases effects to less than 20% in most cases. For the car target: a 1m GSD keeps deviation due to pixel phase effects to less than 20% in most cases. A 2m GSD keeps deviation due to pixel phases effects to less than 30% in most cases.

Future research will focus on incorporating additional information to aide in classification and detection-to-track association, as well as methods to parallelize the algorithms discussed in this paper.

References

[1] S. Blackman and R. Popoli, *Design and Analysis of Modern Tracking Systems*. Norwood, MA: Artech House, 1999.

[2] R. L. Streit and T. E. Luginbuhl, "Probabilistic multi-hypothesis tracking," Tech. Rep. NUWC-PT Technical Report 10,428, Naval Undersea Warfare Center Division, Newport, Rhode Island, September, 1995.

[3] S. Oh, S. Russell, and S. Sastry, "Markov chain monte carlo data association for multi-target tracking," *IEEE Transactions on Automatic Control*, vol. 54, no. 3, 2009.

[4] DuPont Corporation, "2012 automotive color popularity," 2012.

[5] J. Kerekes, D. Pogorzala, J. Parkes, A. Shaw, and D. Rahn, "Hyperspectral target detection using multiple platform cuing," in *2009 First Workshop on Hyperspectral Image and Signal Processing: Evolution in Remote Sensing, WHISPERS '09*, pp. 1–4, Aug 2009.

[6] G. Pulford, "Taxonomy of multiple target tracking methods," *IEE Proceedings - Radar, Sonar and Navigation,*, vol. 152, pp. 291–304, October 2005.

[7] Y. Cheng and T. Singh, "Efficient particle filtering for road-constrained target tracking," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 43, pp. 1454–1469, October 2007.

[8] Y. Cheng and T. Singh, "Efficient particle filtering for road-constrained target tracking," in *2005 8th International Conference on Information Fusion (FUSION)*, vol. 1, pp. 161–168, July 2005.

[9] I. Szottka and M. Butenuth, "Tracking multiple vehicles in airborne image sequences of complex urban environments," in *2011 Joint Urban Remote Sensing Event (JURSE)*, pp. 13–16, April 2011.

[10] M. Zhou, B. Chakraborty, and J. J. Zhang, "Urban terrain multiple target tracking using probability hypothesis density particle filtering," in *2011 Conference Record of the Forty Fifth Asilomar Conference on Signals, Systems and Computers (ASILOMAR)*, pp. 331–335, Nov 2011.

[11] Y. Bar-Shalom, P. K. Willett, and X. Tian, *Tracking and Data Fusion: A Handbook of Algorithms*. Storrs, CT: Yaakov Bar-Shalom, April, 2011.

[12] D. Krajzewicz, J. Erdmann, M. Behrisch, and L. Bieker, "Recent development and applications of SUMO - Simulation of Urban MObility," *International Journal On Advances in Systems and Measurements*, vol. 5, pp. 128–138, December 2012.

[13] http://www.openstreetmap.org, 2014.

[14] "WPAFB 2009 Dataset." `https://www.sdms.afrl.af.mil/index.php?collection=wpafb2009`.

[15] A. Milan, K. Schindler, and S. Roth, "Challenges of ground truth evaluation of multi-target tracking," in *2013 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pp. 735–742, June 2013.

[16] B. Wu and R. Nevatia, "Tracking of multiple, partially occluded humans based on static body part detection," in *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 1, pp. 951–958, June 2006.

[17] P. Willett, Y. Ruan, and R. Streit, "PMHT: problems and some solutions," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 38, pp. 738–754, Jul 2002.

[18] M. A. Fischler and R. C. Bolles, "Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography," *Commun. ACM*, vol. 24, pp. 381–395, June 1981.

[19] S. Choi, T. Kim, and W. Yu, "Performance evaluation of RANSAC family," *Journal of Computer Vision*, vol. 24, no. 3, pp. 271–300, 1997.

[20] M. Zuliani, C. S. Kenney, and B. Manjunath, "The multiransac algorithm and its application to detect planar homographies," in *2005 IEEE International Conference on Image Processing. ICIP 2005.*, vol. 3, pp. III–153, IEEE, 2005.

[21] S. Oh, S. Russell, and S. Sastry, "Markov chain monte carlo data association for multiple-target tracking," tech. rep., University of California, Berkeley, 2008.

[22] J. Prokaj, M. Duchaineau, and G. Medioni, "Inferring tracklets for multi-object tracking," in *Workshop of Aerial Video Processing joint with IEEE CVPR*, 2011.

[23] J. Prokaj, *Exploitation of Wide Area Motion Imagery.* PhD thesis, University of Southern California, August 2013.

[24] F. R. Kschischang, B. J. Frey, and H.-A. Loeliger, "Factor graphs and the sum-product algorithm," *IEEE Transactions on Information Theory*, vol. 47, no. 2, 2001.

[25] A. Globerson and T. Jaakkola, "Fixing max-product: Convergent message passing algorithms for map lp-relaxations.," in *NIPS* (J. C. Platt, D. Koller, Y. Singer, and S. T. Roweis, eds.), Curran Associates, Inc., 2007.

[26] F. Hutter, H. Hoos, and T. Stutzle, "Stochastic local search for the most probable explanation problem (sls4mpe)." `http://www.cs.ubc.ca/labs/beta/Projects/SLS4MPE/`.

[27] J. Kwisthout, "Most probable explanations in bayesian networks: Complexity and tractability," *International Journal of Approximate Reasoning*, vol. 52, no. 9, 2011.

[28] K. Murphy, "Bayes Net Toolbox for MATLAB." `https://code.google.com/p/bnt/`.

[29] J. Prokaj and G. Medioni, "Persistent tracking for wide area aerial surveillance," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2014.

[30] R. W. Osborne, Y. Bar-Shalom, and P. Willett, "Traffic tracking with optical sensors," tech. rep., University of Connecticut, Storrs, CT, 2014.

[31] Digital Imaging and Remote Sensing Laboratory, Rochester Institute of Technology, "DIRSIG." www.dirsig.org, 2014.

This page intentionally left blank.

DISTRIBUTION:

1	MS 0401	Ross L. Hansen, 5521
1	MS 0406	Joshua D. Zollweg, 5772
1	MS 0406	Dylan Z. Anderson, 5772
1	MS 0576	Raymond H. Byrne, 5521
1	MS 0576	John T. Feddema, 5521
1	MS 0576	Tian J. Ma, 5521
1	MS 0576	David K. Melgaard, 5521
1	MS 0822	David B. Karelitz, 9326
1	MS 0973	Gary L. Whitlow, 5771
1	MS 0980	Todd A. Pitts, 5773
1	MS 0980	Prabal Nandy, 5550
1	MS 0980	Daniel A. Bender, 5771
1	MS 1010	Joshua A. Love, 6533
1	MS 0899	Technical Library, 9536 (electronic copy)

Sandia National Laboratories