PROBABILISTIC REAL-TIME DOMAIN AWARENESS LEVERAGING

COMPUTER VISION AND COMPUTATIONAL INTELLIGENCE

A Thesis in
Aerospace Engineering
by
Mark P. Bolden

© 2018 Mark P. Bolden

Submitted in Partial Fulfillment
of the Requirements
for the Degree of

Master of Science

August 2018
The thesis of Mark P. Bolden was reviewed and approved* by the following:

David B. Spencer  
Professor of Aerospace Engineering  
Thesis Advisor

Lyle N. Long  
Distinguished Professor of Aerospace Engineering and Mathematics

Amy R. Pritchett  
Professor and Department Head of Aerospace Engineering

*Signatures are on file in the Graduate School.
Abstract

In tactical operations, decision making must exceed the speed of events [1]. In the space domain, events, such as a satellite break-up event, can occur unexpectedly creating large fields of debris objects. These debris objects are uncontrolled and pose an immediate threat to other satellites. Satellite operators must quickly decide whether or not to maneuver their satellites. If a maneuver is necessary, the operator must also decide where to maneuver the satellite to avoid a probable collision. Due to the complexity of this challenge, traditional statistical approaches struggle to fuse information to track debris clouds on a timeline that exceeds the speed of events. This thesis presents an alternative approach to statistical filters for real-time debris cloud tracking. The approach produces the same level of accuracy in the same amount of time across the entire domain independently from the number of objects in the domain. The approach relies on a new computer vision transform coupled with a new computational intelligence cluster detection algorithm. Combining these techniques enables real-time (millisecond) updates using information from any modality where the uncertainty can be approximated or bounded, ranging from sensing to anecdotal sources. It is capable of ingesting observations that do not meet the typical observability criterion, to include negative detections. It provides real-time state estimates in addition to a full domain population distribution assessment. This includes knowledge of where there is an object, where there is not, and where there is not enough evidence to make a determination. Ultimately, the approach enables the scalable real-time domain awareness required for a decision maker to understand the domain risks and the uncertainties for tactical decision-making. More background on space debris is discussed as the motivation for the technical challenges tackled by this research. Applying computational intelligence with computer vision is presented as a novel approach to solving these challenges in real-time. The concept is first explained on a simple example that leverages a standard computer vision technique known as the Hough transform. Next, the hardware and software design leveraged for testing is described along with performance results on the simple one-dimensional tracking example. How to apply this technique to the space domain is discussed in detail with a proof of concept for the computer vision transform. Potential applications of the computational intelligence algorithm to other domains are discussed. The technique appears to be a viable alternative to statistical filter approaches with significant theoretical advantages, however no direct comparison is presented in this thesis.
Contents

List of Figures viii
List of Tables x
List of Algorithms xi
List of Symbols xii
Acknowledgments xix

Chapter 1
Introduction 1
  1.1 Background . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 1
  1.2 Technical Approach . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 2
    1.2.1 Computational Intelligence for Real-Time Probability Distribution Sampling 2
    1.2.2 Computer Vision for Transforming the Domain . . . . . . . . . . . . . . . 2
    1.2.3 Computational Intelligence for Object Detection and Population Assessment 3
  1.3 Thesis Organization . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 3

Chapter 2
Background 4
  2.1 The Importance of the Space Domain for Humanity . . . . . . . . . . . . . . . . . 4
    2.1.1 Debris Creation . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 4
    2.1.2 Debris Mitigation and Prevention . . . . . . . . . . . . . . . . . . . . . . . 5
    2.1.3 Leverage Partial and Negative Detections from Multiple Modalities in
         Real-Time . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 7

Chapter 3
Establishing Technical Challenges 8
  3.1 Introduction . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 8
  3.2 Simple Case: Multiple Unknown Objects Traveling In One Dimension With Con-
      stant Velocity . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 9
  3.3 Multiple Unknown Objects Transiting or Orbiting the Domain Surrounding a
      Single Planetary Body . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 13
Chapter 4
Computer Vision and Computational Intelligence for Real-Time Finite Sample Generation

4.1 Computer Vision for Object Tracking ........................................ 16
  4.1.1 Detecting 2D Lines in an Image ........................................ 16
  4.1.2 Unmodified Hough Transform Advantages Summary ............... 19
  4.1.3 Applying the Unmodified 2D Hough Transform to 1D Object Tracking 21
  4.1.4 Unmodified Hough Transform Disadvantages Summary: .............. 22
4.2 Adapting the Hough Transform for Real-time Domain Awareness ...... 23
  4.2.1 Real-time Objective ..................................................... 23
  4.2.2 Non-Gridded Objective ................................................ 25
  4.2.3 Computational Intelligence for Real-Time Non-Gridded Sample Generation 25
    4.2.3.1 Sample Generation .............................................. 25
    4.2.3.2 The GODS Algorithm .......................................... 27
    4.2.3.3 The MUSE Algorithm ........................................... 30
    4.2.3.4 The LETHE Algorithm ......................................... 33
  4.2.4 Applying The GODS, The MUSE, and The LETHE ...................... 34
4.3 Summary and Next Steps ..................................................... 36

Chapter 5
BattaliaInfinitum: A real-time probabilistic computational intelligence cluster detection algorithm

5.1 Introduction ................................................................. 38
5.2 Generalized DBSCAN ....................................................... 39
5.3 BattaliaInfinitum Technical Approach ................................... 41
  5.3.1 Initializing BattaliaInfinitum ......................................... 42
  5.3.2 The Strategist-Recruiter-Assassin-Pacifist-Historian (SRAPH) Algorithms ..................................................... 43
    5.3.2.1 The Strategist ..................................................... 43
    5.3.2.2 The Recruiter ..................................................... 45
    5.3.2.3 The Assassin ...................................................... 47
    5.3.2.4 The Pacifist ...................................................... 48
    5.3.2.5 The Historian .................................................... 50
    5.3.2.6 SRAPH Algorithms Summary .................................. 50
  5.3.3 The Hippie-Anarchist-Rebel-Massacrist (HARM) Algorithms ....... 51
    5.3.3.1 The Hippie ....................................................... 51
    5.3.3.2 The Anarchist ................................................... 52
    5.3.3.3 The Rebel ....................................................... 53
    5.3.3.4 The Massacrist .................................................. 55
    5.3.3.5 The HARM Algorithms Summary .............................. 56
5.4 Summary ................................................................. 56

Chapter 6
Three-Dimensional Performance Assessment

6.1 Three-Dimensional Domain for the Performance Assessment .......... 57
  6.1.1 Technical Challenges .................................................. 57
  6.1.2 Technical Objectives ................................................. 59
  6.1.3 Technical Approach .................................................... 59
6.2 Software Architecture .......................................... 60
  6.2.1 Propagate Truth .............................................. 62
  6.2.2 Simulate Observations ....................................... 63
  6.2.3 The GODS .................................................. 63
    6.2.3.1 The MUSE ............................................ 63
    6.2.3.2 The LETHE ........................................... 63
  6.2.4 BattaliaInfinitum ........................................... 64
  6.2.5 Generate State Estimates from Clusters .................. 64
  6.2.6 Monitor Solutions and Compare to Truth ................ 64
  6.2.7 Simulated Observation Visualization .................... 64
  6.2.8 State Space Samples Visualization ....................... 64
  6.2.9 State Space Cluster Visualization ....................... 65
  6.2.10 Domain Knowledge and State Estimation Visualization .. 65
  6.2.11 Normalized Squared Euclidean Distance Performance Visualization .... 67
  6.2.12 Error Performance Visualization ........................ 67
  6.2.13 Target Lock Visualization ................................ 68
  6.2.14 Classification Analytics Visualization ................ 69
    6.2.14.1 False Positive Clusters ............................. 69
    6.2.14.2 Normalized Number of False Negative Points (Samples) .... 71
    6.2.14.3 Normalized Number of Points (Samples) Per Classification ... 72
  6.3 Simulated Test Cases Description .......................... 72
    6.3.1 Defining Simulated Observation Uncertainty ............ 72
    6.3.2 Defining Simulated Observation Probability .............. 72
    6.3.3 Defining Simulated Observation Minimum Data Delivery Rate ... 73
    6.3.4 Defining Simulated Observation Data Volume ............. 73
    6.3.5 Defining Example Test Cases ............................. 73
      6.3.5.1 Test Case 1: Slow Data Delivery Rate of Uniform Uncertainty with No Noise Out of Temporal Sequence ............ 74
      6.3.5.2 Test Case 2: Moderate Data Delivery Rate of Uniform Uncertainty with No Noise Out of Temporal Sequence ........... 80
      6.3.5.3 Test Case 3: High Data Velocity and Volume of Uniform Uncertainty with No Noise Out of Temporal Sequence .......... 85
      6.3.5.4 Test Case 4: High Data Velocity and Volume of Highly Variable Uncertainty with No Noise Out of Temporal Sequence ... 90
      6.3.5.5 Test Case 5: High Data Velocity and Volume of Highly Variable Uncertainty with High Noise Out of Temporal Sequence .... 95
  6.4 Results Summary ............................................. 100

Chapter 7
Modifications for Space Domain Application .......................... 101
  7.1 Processing Modifications for the Space Domain .............. 101
    7.1.1 Propagate Truth Modifications .......................... 104
    7.1.2 Simulate Observations Modifications .................... 104
    7.1.3 The MUSE Modifications ................................ 105
    7.1.4 BattaliaInfinitum Modifications ........................ 106
    7.1.5 Generate State Estimates from Clusters Modifications .... 107
    7.1.6 Visualization Modifications ............................. 107
  7.2 Space Domain Proof of Concept Example ....................... 107
Chapter 8

Conclusions

8.1 Top Level Objectives

8.1.1 Technical Approach

8.1.2 Scalability

8.1.3 Data Association

8.1.4 Fusion

8.1.5 Multiple Modalities and Measurement Types

8.1.6 Adaptability

8.1.7 Initial Assumptions

8.1.8 Performance Summary

8.2 Future Work

8.2.1 Next Steps For the Space Domain

8.3 Applicability to Other Domains

8.4 Conclusions Summary

Appendix

Videos of Live Results

References
List of Figures

3.1 Truth Locations In One Dimension Over Time for Three Ghosts .......................... 10
3.2 Truth Locations and Observations In One Dimension Over Time for Three Ghosts 10
3.3 Measurement Input Mapped to Desired Output ................................................ 11
3.4 Measurement and Cartesian States ................................................................. 14
4.1 Image With Three Lines .......................................................... 17
4.2 Image With Three Lines and Hough Lines ...................................................... 17
4.3 Hough Transform Processing Flow .............................................................. 18
4.4 Hough space Matrix, \( HT \) ................................................................. 19
4.5 SNR = 0.5 ........................................................................... 20
4.6 SNR = 0.2 ........................................................................... 20
4.7 SNR = 0.154 ........................................................................... 20
4.8 Truth Locations In one-dimension Over Time for Three Ghosts ......................... 21
4.9 Trade Space Comparison .............................................................................. 23
4.10 Screen Mesh Filter with SNR = 0.5 .............................................................. 24
4.11 Random Filter with SNR = 0.5 ...................................................................... 24
4.12 Simple Sampling Example ........................................................................ 26
4.13 GODS Processing Flow .............................................................................. 28
4.14 Visit Ratio Threshold versus Degree-of-Wisdom .......................................... 29
4.15 MUSE Processing Flow .............................................................................. 30
4.16 Cumulative Probability of Pulling New Samples < \( U_{\text{MaxNorm}} \) ................. 32
4.17 Probability Update versus De-weighting Parameter ........................................ 33
4.18 LETHE Processing Flow .............................................................................. 33
4.19 Simple Random Sampling Hough space Example ........................................... 35
4.20 Truth Locations and Observations In one-dimension Over Time for Three Ghosts 36
4.21 Observations and Corresponding Generated Samples ....................................... 37
4.22 Observations and Corresponding Generated Samples With Truth .................. 37
5.1 DBSCAN Inputs and Outputs .......................................................... 40
5.2 DBSCAN Process ........................................................................... 41
5.3 BattaliaInfinitum Top Level Processing Flow .................................................. 44
5.4 The Recruiter Example: First New Cluster ....................................................... 46
5.5 The Recruiter Example: Add Samples to Cluster ............................................. 47
5.6 The Recruiter Example: Second New Cluster .................................................. 47
5.7 The Assassin Example: First Negative Cluster .................................................. 48
5.8 The Pacifist Example: Unknown Cluster .......................................................... 49
5.9 The RAP Group Stable Example .......................................................... 49
List of Tables

3.1 Technical Challenges .................................................. 12
5.1 DBSCAN and BattaliaInfinitum Requirements Compliance Comparison . . . 42
6.1 Definitions for Example Test Cases ..................................... 74
List of Algorithms

4.1 Hough Transform Pseudo code .................................................. 18
4.2 The GODS Pseudo code ............................................................... 27
4.3 The MUSE Pseudo code ............................................................... 31
4.4 The LETHE Pseudo code .............................................................. 34
4.5 Probabilistic Random Sampling Hough Transform ...................... 34
5.1 Initializing BattaliaInfinitum ......................................................... 43
5.2 The Strategist ............................................................................ 45
5.3 The Recruiter ............................................................................ 46
5.4 The Assassin .............................................................................. 48
5.5 The Pacifist ............................................................................... 49
5.6 The Historian ............................................................................ 50
5.7 The Hippie ................................................................................. 51
5.8 The Anarchist ............................................................................ 52
5.9 The Rebel (DBSCAN) .................................................................. 54
5.10 The Massacrist .......................................................................... 55
7.1 The MUSE for Space ................................................................. 106
List of Symbols

$\vec{X}$ Cartesian state vector for simple case, p. 9
$x_0$ initial position for simple case, p. 9
$\dot{x}_0$ initial velocity for simple case, p. 9
$\vec{s}$ measurement sample vector for simple case, p. 9
$x$ position location for simple case, p. 9
$t$ time for simple case, p. 9
$p$ probability for simple case, p. 9
$\vec{U}_{Max}$ measurement sample uncertainty vector for simple case, p. 9
$u_x$ position uncertainty for simple case, p. 9
$u_t$ time uncertainty for simple case, p. 9
$u_p$ probability uncertainty for simple case, p. 9
$\vec{X}_{obj}$ Cartesian state vector in Planetary Centered Inertial (PCI) frame for object orbiting a planetary body, p. 13
$x_{obj}$ Cartesian x-position in PCI frame for object orbiting a planetary body, p. 13
$y_{obj}$ Cartesian y-position in PCI frame for object orbiting a planetary body, p. 13
$z_{obj}$ Cartesian z-position in PCI frame for object orbiting a planetary body, p. 13
$\dot{x}_{obj}$ Cartesian x-velocity in PCI frame for object orbiting a planetary body, p. 13
$\dot{y}_{obj}$ Cartesian y-velocity in PCI frame for object orbiting a planetary body, p. 13
$\dot{z}_{obj}$ Cartesian z-velocity in PCI frame for object orbiting a planetary body, p. 13
$\ddot{X}_{obj}$ Cartesian double derivative vector in PCI frame for object orbiting a planetary body, p. 13
$\mu$ gravitational constant for a planetary body, p. 13
$r_{\text{obs}}$ Cartesian position radius in PCI frame for object orbiting a planetary body, p. 13

$\dot{r}_{\text{obs}}$ Cartesian position radius rate in PCI frame for object orbiting a planetary body, p. 14

$\theta_{\text{obs}}$ Cartesian theta angle in PCI frame for object orbiting a planetary body, p. 14

$\dot{\theta}_{\text{obs}}$ Cartesian theta angle rate in PCI frame for object orbiting a planetary body, p. 14

$\phi_{\text{obs}}$ Cartesian phi angle in PCI frame for object orbiting a planetary body, p. 14

$\dot{\phi}_{\text{obs}}$ Cartesian phi angle rate in PCI frame for object orbiting a planetary body, p. 14

$t_{\text{obs}}$ time of observation in PCI frame for object orbiting a planetary body, p. 14

$\vec{S}_{\text{obs}}$ observation vector in PCI frame for object orbiting a planetary body, p. 14

$\vec{U}_{\text{Obs}}$ observation uncertainty vector in PCI frame for object orbiting a planetary body, p. 15

$u_{r_{\text{obs}}}$ observation radius uncertainty in PCI frame for object orbiting a planetary body, p. 15

$u_{\dot{r}_{\text{obs}}}$ observation radius rate uncertainty in PCI frame for object orbiting a planetary body, p. 15

$u_{\theta_{\text{obs}}}$ observation $\theta$ uncertainty in PCI frame for object orbiting a planetary body, p. 15

$u_{\dot{\theta}_{\text{obs}}}$ observation $\theta$ rate uncertainty in PCI frame for object orbiting a planetary body, p. 15

$u_{\phi_{\text{obs}}}$ observation $\phi$ uncertainty in PCI frame for object orbiting a planetary body, p. 15

$u_{\dot{\phi}_{\text{obs}}}$ observation $\phi$ rate uncertainty in PCI frame for object orbiting a planetary body, p. 15

$u_{t_{\text{obs}}}$ observation time uncertainty in PCI frame for object orbiting a planetary body, p. 15

$u_{p_{\text{obs}}}$ observation probability uncertainty in PCI frame for object orbiting a planetary body, p. 15

$\rho$ length of the perpendicular to a possible line from the origin in the image, p. 16

$\zeta$ angle of the perpendicular to a possible line from the origin in the image, p. 17

$X$ horizontal pixel dimension in the image, p. 17

$Y$ vertical pixel dimension in the image, p. 17

frame pointer to image frame for evaluation, p. 18

IMAGE matrix that contains the image for possible line detection, p. 18

$HT$ matrix that contains the Hough space for possible line detection, p. 18

$x_{\text{pix}}$ integer horizontal value of current pixel to be transformed into Hough space, p. 18
$N_x$  total number of x-dimension pixels to be transformed, p. 18
$y_{pix}$  integer vertical value of current pixel to be transformed into Hough space, p. 18
$N_y$  total number of y-dimension pixels to be transformed, p. 18
$D_w$  degree of wisdom - derived from the level of significance, p. 27
$\alpha$  level of significance - a user defined parameter for confidence in statistical tests, p. 27
$S_{add}$  variable for computing number of new samples to create, p. 27
$S_{del}$  variable for computing number of existing samples to delete or archive, p. 27
$T_{Samp}$  total number of existing samples, p. 27

$S_{range}$  number of most recent samples to evaluate to assess BattaliInfinitum performance, p. 27

$N_V$  number of samples visited by BattaliInfinitum, p. 27
$V_R$  visit ration - percent of samples visited by BattaliInfinitum, p. 27
$N_S$  number of samples to create or delete/archive, p. 27

$\overrightarrow{Res}_{max}$  user defined maximum relevant state estimation resolution, p. 28
$\overrightarrow{Res}_{min}$  user defined minimum relevant state estimation resolution, p. 28
$C_S$  number of observations to create $N_S$ samples from, p. 31
$\overline{U}_{Max}$  maximum uncertainty bound of a single observation, p. 31
$U_{Filt}$  maximum uncertainty bound to randomly query for observations, p. 31
$\bar{O}$  group of observations to randomly select from to create samples, p. 31
$S$  number of samples previously generated from a given observation, p. 31
$S_{min}$  minimum sample number which corresponds to the poorest sampled observations, p. 31

$U_{MaxNorm}$  maximum observation uncertainty normalized by the size of the domain, p. 31
$D_{range}$  size of the domain, p. 31
$P_{Obs}$  probability associated with the selected observation, p. 31
$U_{Max}$  maximum uncertainty associated with the selected observation, p. 31
$P_{Samp}$  probability to assign to samples in the Hough space, p. 31
$n$  de-weighting factor, p. 31
$P$  probability to de-weight, p. 31
$G_{\text{min}}$ sample number associated with the oldest sample, p. 34

$G_S$ global sample number to delete, p. 34

$x_{\text{samp}}$ value of $x$ for computing the sample values in Hough space, p. 34

$t_{\text{samp}}$ value of $t$ for computing the sample values in Hough space, p. 34

$p_{\text{samp}}$ value of $p$ for computing the sample values in Hough space, p. 34

$\zeta_{\text{samp}}$ value of $\zeta$ for computing the sample values in Hough space, p. 34

$\rho_{\text{samp}}$ computed value of $\rho$ for sample in Hough space, p. 34

$\text{searchDistance}$ dynamic variable that defines the radius for all BattaliaInfinitum operations, p. 43

$P_c$ local cumulative probability for all BattaliaInfinitum operations, p. 43

$\mathcal{N}_{\text{new}}$ dynamic variable for tracking the number of points in an iteration of BattaliaInfinitum, p. 43

$\text{minClusterNumber}$ dynamic variable for logging minimum cluster number assignment BattaliaInfinitum, p. 43

$\text{maxClusterNumber}$ dynamic variable for tracking the number of clusters found by BattaliaInfinitum, p. 43

$\bar{X}_{\text{cent}}$ centroid of random location in the state space to be evaluated, p. 45

$\bar{S}_{\text{range}}$ vector containing the range of the domain in all dimensions, p. 45

$\bar{S}_{R}$ vector containing the ids of any samples in local region, p. 45

$\bar{S}$ vector all samples in the state space, p. 45

$P_n$ probability for a single sample, p. 45

$\bar{P}_c$ vector of the cumulative probability from the most recent iterations of BattaliaInfinitum, p. 45

$\text{medianCumulativeProbability}$ median of the cumulative probability from the most recent iterations of BattaliaInfinitum, p. 45

$\text{meanCumulativeProbability}$ mean of the cumulative probability from the most recent iterations of BattaliaInfinitum, p. 45
\( \text{stdCumulativeProbability} \) standard deviation of the cumulative probability from the most recent iterations of \textit{BattaliaInfinitum}, p. 45

\( D_f \) degree of freedom to use for the non-central student t-test, p. 45

\( \delta \) non-centrality parameter to use for the non-central student t-test, p. 45

\( \text{nc}t\text{Thresh} \) threshold to use for the non-central student t-test, p. 45

\( \text{nc}t\text{PositiveThresh} \) threshold for a cluster to be positive, p. 46

\( \text{clusterNumbers} \) current cluster number labels for a local region, p. 46

\( \text{idxPos} \) current cluster number labels for a local region, p. 46

\( \text{modePositive} \) mode of current cluster number labels for a local region, p. 46

\( \text{nc}t\text{NegativeThresh} \) threshold for a cluster to be negative, p. 48

\( \overline{NP}_{old} \) dynamic variable for tracking the number of points in the last iteration of \textit{BattaliaInfinitum}, p. 50

\( NP_n \) number of samples in the last iteration of \textit{BattaliaInfinitum}, p. 50

\( \text{clusterNumber}_H \) cluster number of cluster for evaluation by the \textit{Hippie}, p. 51

\( \overline{S}_H \) vector containing the ids of any samples in cluster for evaluation by the \textit{Hippie}, p. 51

\( \text{unqClusterNums} \) vector containing the unique cluster numbers in the cluster for evaluation by the \textit{Hippie}, p. 51

\( \text{clusterNumber}_A \) cluster number of cluster for evaluation by the \textit{Anarchist}, p. 52

\( \overline{S}_A \) vector containing the ids of any samples in cluster for evaluation by the \textit{Anarchist}, p. 52

\( \text{newClusters} \) flag indicating whether or not a new clusters have been created by the \textit{Rebel}, p. 54

\( \text{clusterNumber}_R \) cluster number of cluster for evaluation by the \textit{Rebel}, p. 54

\( \overline{S}_R \) vector containing the ids of any samples in cluster for evaluation by the \textit{Rebel}, p. 54

\( \text{idx}S \) current index to evaluate in the \textit{Rebel}, p. 54

\( \overline{S}_{sub} \) vector containing the ids of any local region samples in cluster for evaluation by the \textit{Rebel}, p. 54

\( \text{rebelCount} \) number of new clusters to be created by the \textit{Rebel}, p. 54

\( \overline{S}_{more} \) vector containing the ids of any additional samples are found to belong to the current sub cluster, p. 54

xvi
$\text{clusterNumber}_R$  cluster number of cluster for evaluation by the **Massacrist**, p. 55

$\vec{S}_R$  vector containing the ids of any samples in cluster for evaluation by the **Massacrist**, p. 55

$\sigma$  standard deviation value, p. 67

$Weighted$  weighted number of false positives found, p. 70

$cluster\_probability$  cumulative probability of the cluster, p. 70

$falsePosCluster$  vector with samples from a single false positive cluster, p. 70

$truePosCluster$  vector with samples from the true positive clusters, p. 70

$Uncert_{max}$  maximum observation uncertainty, p. 72

$Uncert\_varFlag$  variable observation uncertainty flag, p. 72

$P_{Max}\Delta$  maximum range of probability, p. 72

$P_{varFlag}$  variable probability flag, p. 72

$P_{MaxNoise}$  maximum range of probability, p. 72

$P_{varFlag}$  random noise added to probabilities, p. 73

$P_{obsClean}$  probability without noise, p. 73

$P_{noise}$  probability noise to introduce, p. 73

$\Delta t_{max}$  maximum observation latency, p. 73

$N_{obs}$  number of observations to generate for the simulation, p. 73

$p_{SR}$  semilatus rectum, p. 101

$f$  modified equinoctial elements f-component, p. 101

$e$  eccentricity, p. 101

$\omega$  argument of periapsis, p. 101

$I$  retrograd factor, p. 101

$\Omega$  right ascension of ascending node, p. 101

$g$  modified equinoctial elements g-component, p. 101

$h$  second equinoctial element, p. 101

$i$  inclination, p. 101

$k$  third equinoctial element, p. 101

$L$  true longitude, p. 101

$t_0$  initial reference time, p. 101
\( \theta_{TL} \) true anomaly, p. 101

\( \bar{X}_{\text{samp}} \) Cartesian state vector in Planetary Centered Inertial (PCI) frame for a sample generated from an observation, p. 104

\( x_{\text{samp}} \) Cartesian x-position in PCI frame for a sample generated from an observation, p. 104

\( y_{\text{samp}} \) Cartesian y-position in PCI frame for a sample generated from an observation, p. 104

\( z_{\text{samp}} \) Cartesian z-position in PCI frame for a sample generated from an observation, p. 104

\( \dot{x}_{\text{samp}} \) Cartesian x-velocity in PCI frame for a sample generated from an observation, p. 104

\( \dot{y}_{\text{samp}} \) Cartesian y-velocity in PCI frame for a sample generated from an observation, p. 104

\( \dot{z}_{\text{samp}} \) Cartesian z-velocity in PCI frame for a sample generated from an observation, p. 104

\( \text{COE}_{\text{samp}} \) classical orbital elements for a sample generated from an observation, p. 104

\( \bar{n}_{\text{samp}} \) mean motion for a sample generated from an observation, p. 105

\( \bar{e}_{\text{samp}} \) eccentricity for a sample generated from an observation, p. 105

\( \bar{E}_{\text{samp}} \) eccentric anomaly for a sample generated from an observation, p. 105

\( \bar{\theta}_{\text{samp}} \) true anomaly for a sample generated from an observation, p. 105

\( \bar{M}_{\text{samp}} \) mean anomaly for a sample generated from an observation, p. 105

\( \bar{F}_{\text{samp}} \) hyperbolic eccentric anomaly for a sample generated from an observation, p. 105

\( a \) semi-major axis, p. 107
Acknowledgments

To my first career mentor: I would not have the career I have today without your patient guidance. I will never forget the three major lessons you shared: 1. Know what you know, know what you don’t know, but most of all, know who to ask when you don’t know. 2. Until you can demo on real data, be extremely skeptical. 3. Ninety-nine percent of success is due to the team, not the manager or the individual, so focus on building a strong team.

To an early supervisor: I’ll never forget your wisdom during one of my most opinionated moments: “Mark, it’s like I tell my kids, sometimes you just need to sit down, shut up, and color.”

To a former leader, thank you for highlighting the importance of shaping the narrative as well as the technology, and that there is a time and a place to be a “bull in a china shop.”

To a recent mentor, thank you for pointing out the importance of recognizing when you’ve become entrenched in localized confirmation bias, and how to grow past it.

To my fellow graduate students: Andrew Goodyer, Koundinya Kuppa, Molik Nayar, and Jason Reiter: I’ll always appreciate your interest in my research progress and willingness to bounce back ideas. I especially appreciate your peer review the day I rewrote the Computational Intelligence algorithm on the whiteboard in the grad student office.

To my research advisor, David Spencer: I am very thankful to have had you as a professor in all of my astrodynamics courses. Learning the fundamentals from a mathematical and engineering perspective rather than from a political view was long overdue. In addition, your guidance through the development of my thesis has been paramount to making it towards graduation. Special thanks for your understanding of the delays incurred due to my return to full-time work.
Dedication

I am dedicating this thesis to my parents, Patrick and Kathleen Bolden. If it were not for your unwavering support, I would never have made it to graduation. Thank you for your persistence on the importance of finally graduating. Every day I’m a better person due to your selfless efforts to ensure my success and happy life.
Chapter 1

Introduction

1.1 Background

For the last 60 years, humanity has been attempting to better understand the Earth orbit space domain population. Historically this has been accomplished by developing sophisticated networks of optical telescopes and radars to detect and track satellites and debris. However, the approaches typically leveraged are based on statistical filters and limited to positive detections that meet the observability criterion. As a result, the majority of observations in these networks are discarded, lowering the overall efficiency of the networks. Although the generalized traditional statistical filter approach can consistently produce very accurate state estimates, there are several disadvantages:

1. it scales non-linearly computationally with increasing number of objects and/or number of observations,

2. it does not leverage negative detections, and

3. it does not produce estimates in real-time (i.e. milliseconds).

Based on these disadvantages, the initial objective of this research was to develop a means to exploit partial and negative detections. During development it became clear that there were multiple other beneficial advantages possible with the chosen approach. These possibilities became the primary objectives during the development. At that point the objectives were well set and held consistent through the completion of this study.

The overarching objective became to generate real-time adaptive population distribution and state estimates for all regions of interest using any and all data sources (e.g. modalities, negative, positive, and partial detections). This is accomplished by applying computer vision techniques combined with computational intelligence to estimate the full domain occupancy by performing data association, initial orbit determination and state estimate updates.
Due to the population growth in the space domain, the developed approach scales linearly based on desired accuracy, database read/write performance, and computational resources independent of the number of observations and objects. The fusion occurs in parallel in the state space for all observations by mapping each to the state space independently. Each update to the fusion process occurs in milliseconds. Data association is not a binary process, rather probabilistic across all possibilities and performed in the state space. The developed approach is able to adapt to new observations without global re-processing of old information. It fuses observations from any modality (e.g. radar, optical, passive RF, anecdotal), of any detection type (e.g. positive, negative, or partial detections). The approach also requires no initial assumptions and can estimate uncertainty distributions of any kind (i.e. no Gaussian uncertainty assumption).

1.2 Technical Approach

Computer vision techniques, combined with computational intelligence, are applied to estimate domain population and to perform data association, initial orbit determination and state estimate updates. This approach is a fundamental shift from current approaches, which are typically formulated based on batch or sequential filters [2]. While there are promising techniques that are possible using more traditional approaches, there are multiple benefits of this type of an approach.

1.2.1 Computational Intelligence for Real-Time Probability Distribution Sampling

The real-time requirement is by achieved leveraging Probability Distribution Sampling. A fundamental question is the choice of how many samples to generate. There are multiple conventions for best practices, however these approaches are based on the concept of the final number of samples required for confidence and accuracy [3]. The unique approach of this research is to instead generate the maximum possible number of samples without impeding real-time requirements. The approach monitors the processing speed performance of the analysis and scales the number of samples appropriately. The benefit is that results are available in real-time, regardless of the sophistication of the computer architecture. Faster databases and more processors mean more accurate results with higher confidence.

1.2.2 Computer Vision for Transforming the Domain

The computer vision technique is the fundamental technology driving this method. It is based on a computer vision technique for detecting lines in multiple dimensions. Any object trajectory approximately following a system of exact governing equations can be completely described as a line in the relevant number of dimensions. This trajectory is known as a geodesic [4]. In this research, the proof-of-concept is demonstrated for objects moving through a single dimension
with constant velocity. The process adaptation required for the processing of the orbital space
domain is later described with a simple proof-of-concept example.

1.2.3 Computational Intelligence for Object Detection and Population
Assessment

The third pillar of the technical approach is a real-time cluster detection algorithm for assessing the transformed samples in the state space. The approach is unique in the sense that it produces live real-time estimates that are determined without any noise distribution assumptions. The end product is a full domain assessment based on the available computational resources and observations provided.

1.3 Thesis Organization

This thesis is broken down into multiple chapters and subsections to best communicate the need, motivation, technical approach, simple example results, future extensibility, and conclusions. Chapter 2 provides background information on the use case that motivated this research. It discusses history and challenges of space domain awareness as the earth orbiting space population continues to grow. Chapter 3 discusses a very simple non-space example that will be used to demonstrate the technique. In addition, it outlines the technical objectives that drive the need for a new approach. It highlights the reasons for why traditional approaches are not sufficient for these objectives. A significant challenge for this research is effectively communicating not only how this approach works, but also why it works. The reason for this difficulty is intrinsic in the design. The approach only functions properly when all of the individual algorithms are run in parallel. While the “how” can be communicated for each individual algorithm, the “why” can only be explained once the reader has a complete understanding of each individual component. Chapters 4 and 5 focus on the “how” by breaking down the algorithm of each independent function. Chapter 4 primarily discusses how the algorithm maps from Cartesian space to the geodesic state space and how the system decides when to generate new samples or deprecate old samples. Chapter 5 then discusses how these samples are analyzed. The chapter steps through each individual algorithm to explain “how” it works. Chapter 6 then demonstrates and discusses the results of the approach on the simulated simple example outlined in Chapter 3. Chapter 7 then discusses how this could be applied to the space domain. Before moving on to future work, Chapter 8 discusses the conclusions and why the approach works. It focuses on the relationships between the various algorithms and assumes the reader either fully understands how each algorithm functions, or is able to reference back to the previous section. The main objective of Chapter 8 discusses the conclusions that can be drawn based on the results from Chapters 6 and 7. In addition it discusses potential future work and possible other applications.
Background

2.1 The Importance of the Space Domain for Humanity

Shortly after NASA began exploring space, governments began to develop new strategies, such as the Navigation System with Timing and Ranging (NAVSTAR), to gain air, land, and sea superiority by leveraging space to safeguard their citizens from outside threats [5].

As time passed commercial companies also began to find cost effective technologies to improve communications and to better understand the characteristics of Earth in minutes [6][7]. As with any consistently available luxury, the human species is becoming more and more reliant on space across all facets of society. The most interesting thing about Earth orbital space is how abstracted our reliance on it has become to the majority of the world. Almost everyone in the world is now reliant on space in some way. Some of these benefits are obvious, such as GPS and communications satellites, while others are subtler, such as weather forecasting and resource monitoring [8][9].

With the understanding that all of Earth’s societies are reliant on the ability to leverage space, protection of this resource is critical. There are several major apparent risks to satellites in Earth’s orbit, however only one hazard is growing at an accelerated rate: the risk of collision between tracked objects in space [10].

2.1.1 Debris Creation

Though humanity has only been launching satellites for approximately 60 years, the space environment in near-Earth space is becoming increasingly utilized and congested [11]. As a result, two types of events are becoming more common: break-up events and collisions. A break-up event occurs when a satellite either falls apart or breaks apart in orbit without a known collision
with a second tracked object. Collisions occur when two known objects collide in space [12][11]. Break-up events create a single debris cloud that spreads out over time, while collisions create two separate but intersecting debris clouds. Over time, these debris clouds spread out and form rings around the Earth [13]. However, unlike the rings of Saturn, these rings are not stable flat equatorial rings. They contain large numbers of High Area-to-Mass Ratio (HAMR) objects, which are highly affected by non-conservative or non-gravitational forces. As a result they spread out over time creating a larger and larger debris cloud, threatening more and more satellites [14].

2.1.2 Debris Mitigation and Prevention

The good news is that in lower orbits, Earth’s atmosphere plays a large role in debris removal. The atmosphere acts as a source of drag, gradually slowing objects down over time. Eventually many debris objects will burn up in the atmosphere and no longer pose a collision hazard to other satellites. Depending on the orbit of the debris, this process can take years to centuries. However, there are some debris objects that will never decay. The reason for this is non-conservative forces. Everything from solar radiation pressure to the gravity of the moon can effectively “trap” debris in orbit forever. Mitigation strategies are needed to accelerate the process of debris removal. There are many concepts such as developing extremely large nets to capture and deorbit debris or building giant ground based lasers to ablate debris to effectively maneuver the debris into decay. Unfortunately the designs discussed so far require extremely large budgets and barely make a dent in the overall debris population [15].

Due to the impractical nature of these concepts, the space community has chosen to focus on prevention of new debris and management of existing debris, rather than the removal of existing debris. There are two primary ways to making progress on prevention and management:

1. Satellite design, and

2. High accuracy debris tracking for conjunction analysis to prevent future collisions.

Satellites are primarily designed to perform their mission as cost effectively as possible. Most developers have end-of-life plans to either decay the satellites or move them to higher orbits. Unfortunately, not all satellite missions go as planned. Anything from communications failure to collisions can thwart these good intentions. Every time that happens, more debris is created. In some communities, researchers are beginning to look at ways to design satellites to create as little debris as possible, should the worst happen. However, this is not a common practice [16].

The other approach is high accuracy tracking, which is the most common approach. The objective is to know the location of every single satellite and piece of debris and to alert satellite owner/operators (O/O) of potential collisions. The O/O’s then perform maneuvers to avoid the collisions. Even if this approach was achievable, it is still fundamentally flawed. Not all satellites are designed with maneuvering capability, and every maneuver exhausts propellant. Lastly, there
is no way to prevent two debris objects from colliding, even if you know the collision is imminent. While the Iridium/Cosmos collision in 2009 was not predicted ahead of time, if it had been, maneuvering would not have been possible [17].

Despite the limitations of the high accuracy tracking approach, it is the most effective approach available today. In fact, since the 1960’s, governments have been attempting to design systems and algorithms to keep pace with the growing space satellite and debris population. These include but are not limited to efforts such as Project Moonwatch [18], the Space Surveillance Network (SSN) [19], OrbitOutlook (O2) [20], Russian Space Surveillance System (RSSS) [21], International Scientific Optical Observation Network (ISON) [22], Chinese Space Surveillance System [23], and European Space Surveillance and Tracking [24]. An interesting development in recent years has been the emergence of commercial space surveillance networks such as ExoAnalytic Space Operations Center (ESpOC) [25] and AGI’s Commercial Space Operations Center (COMSPOC) [26].

Each of these systems produce measurements on possible satellites for downstream processing. Once a sensor has determined it has detected an object a minimum number of times, this information is relayed to fusion algorithms that attempt to correlate the measurements to known objects or create an orbit for a new object. At a very high level, this is commonly performed either using batch or Kalman statistical filters to estimate an orbit and a covariance for that orbit. A statistical filter is formulation which takes into account the mapping of the measurement space to a state space, and iteratively refines and initial estimate in the state space. There are multiple downsides to this approach:

1. Requires minimum number of positive observations (observability criterion)
2. Requires initial guess for estimate and uncertainty
3. Assumes Gaussian Uncertainty
4. Can converge to local minima (Sequential)
5. Requires global reprocessing (Batch Least Squares)
6. Can fail to converge
7. Scales non-linearly with the number objects and/or observations

Even despite all of these possible flaws there are no known methods proven to be superior for orbit determination that do not require either a Kalman or batch filter [2][27]. The challenge for this research is simple: Is there a fundamentally different way of performing orbit determination that does not suffer these challenges?
2.1.3 Leverage Partial and Negative Detections from Multiple Modalities in Real-Time

When this research effort began, figuring out a way to build orbits using the data that is typically unused, was the top priority. In fact, the initial publication for this research was titled Serendipitous Acquisition of Space Situational Awareness From Astronomical Surveys (SASSAFRAS) [28]. This paper went through a simple calculation to assess the potential available volume and value of this neglected data, which is virtually free. The analysis was based on a relevant-byte metric from a previous DARPA publication [29]. It was also based on open source claims about how much untapped data really existed. This simple analysis performed in SASSAFRAS argued that there was potential value in this neglected information if there was a method capable of leveraging it.

Ultimately, the motivation for domain awareness is to enable an owner/operator (O/O) to make a decision based on the information available. As a result, real-time fusion is extremely important. If the O/O needs to wait for a solution, the time to make the decision may have passed.

As development on the original SASSAFRAS concept progressed, it became increasingly obvious that the approach had significant other benefits if leveraged in the proper way. The original objectives of this research did not include real-time or full domain fusion, however it quickly became apparent that with modest computational intelligence algorithms, combined with a new computer vision technique, we could enable real-time multi-modal domain fusion. As a result, the emphasis became to develop and demonstrate the full process on a simple example before automating the process for space. Significant effort was put into figuring out which computer language and database combination was appropriate for the research and development. There were many factors, but ultimately query speed and open source availability drove the final decision. This powerful combination of technologies from multiple disciplines is necessary to solve the technical challenges posed by this research. The next chapter details these technical challenges and outlines specific applications for the technology.
Establishing Technical Challenges

3.1 Introduction

A process to probabilistically assess the population of a domain and to produce state estimates in real-time independent of the number of observations or objects in the domain is presented. The approach developed does not leverage statistical filters to estimate states and covariance matrices. Instead, it presents a way to assess the domain population, number of objects, states, and irregular uncertainty distributions using a combination of computer vision and computational intelligence techniques. While traditional statistical approaches will be referenced to contrast and draw corollaries between the distinct approaches, no assertions of true comparison will be made. The purpose of this research is to determine if it is possible to perform the objectives leveraging this non-traditional approach. Evaluating the performance of this technique versus any other approach is beyond the scope of this thesis.

Since this approach does not leverage any of the traditional conventions for statistical object tracking, a very simple example will be discussed primarily, to convey the approach with the objective of eventually applying the technique to the space domain. This chapter will not discuss how this process works, rather it will discuss the two specific use cases and the technical challenges associated with each. Details on how this process works will be covered in Chapters 4-5.

In order to most effectively explain the approach, two different use cases will be discussed throughout this thesis:

1. Multiple Unknown Objects Traveling In One Dimension With Constant Velocity
2. Multiple Unknown Objects Transiting or Orbiting the Domain Surrounding a Single Planetary Body
3.2 Simple Case: Multiple Unknown Objects Traveling In One Dimension With Constant Velocity

An extremely simple example will be the basis for initially explaining how this approach works. The challenge will be to track an unknown number of objects moving through a single dimension, $x$, over time, $t$, at constant velocities $\dot{x}$. For each object, the Cartesian state vector, $\vec{X}$, at any point in time is completely described by the following:

$$\vec{X} = \begin{bmatrix} x_0 \\ \dot{x}_0 \end{bmatrix}$$  \hspace{1cm} (3.1)

and is subject to the following equation.

$$x = x_0 + \dot{x}_0 t$$  \hspace{1cm} (3.2)

Each measurement sample, $\vec{S}$, will observe only position, $x$, time $t$, and probability of object existence, $p$, and assume knowledge of the maximum possible uncertainty, $\vec{U}_{Max}$, associated with that measurement in each parameter denoted, $u_x, u_t, u_p$.

$$\vec{S} = \begin{bmatrix} x \\ t \\ p \end{bmatrix}, \vec{U}_{Max} = \begin{bmatrix} u_x \\ u_t \\ u_p \end{bmatrix}$$  \hspace{1cm} (3.3)

It is important to emphasize that $\vec{U}_{Max}$ does not represent the actual error in the measurement. Rather, it represents the maximum possible error based on an independent assessment of the system providing the samples. This algorithm assumes other processes are in place that are capable of quantifying a maximum possible uncertainty of a given measurement. Ideally a probability distribution of the measurement uncertainty in the sample is preferred, however not assumed for this process. If it were, the algorithm would undoubtedly produce superior results in shorter time frames.

To preserve overall simplicity, the example assumes that objects do not interact as they occupy the same position, $x$. Due to this non-interaction each object is considered to be a ghost, in order to avoid confusion on this point. Lastly, for this example, the objective is to determine the number of ghosts traversing the domain, their path through the domain, and locations in time where there are not any ghosts. The algorithm will not have a priori knowledge of the number of ghosts in the domain. In Figure 3.1, both position and time have been plotted to illustrate that any objects motion is essentially just a line through the relevant dimensions. In this simple case, the relevant dimensions are position and time. Velocity is not directly measured and data association is not provided, therefore
Figure 3.1: Truth Locations In One Dimension Over Time for Three Ghosts

Each individual measurement does not satisfy the observability criterion. Figure 3.2 plots these random measurements. If a ghost is observed, then the observation location is plotted in yellow. If not, then the observation is plotted in blue. In real-time, as these observations of $\mathbf{S} = [x, t, p]^T$

Figure 3.2: Truth Locations and Observations In One Dimension Over Time for Three Ghosts

are randomly provided out of sequence, the algorithm must determine the following:

1. The probability of ghost existence for position, $x$, and as function of time, $t$ across the
1. Initial Trajectory Determination:
   - Each possible combination of positive detections, $\overrightarrow{S}$, can be considered to compute possible trajectory associations by using statistical filters to estimate trajectories.

2. Cluster Detection for Trajectory Association:
• Leveraging state estimates produced by the filters, a cluster detection algorithm can be used to determine how many objects have been detected, and observations estimated to correspond to those objects.

3. Trajectory Refinement:

• Now that the superset of observations corresponding to each object has been estimated, each superset can be run through the statistical filter again to refine the trajectory estimates.

While this generalized traditional approach can consistently produce very accurate state estimates there are several disadvantages:

1. It scales non-linearly computationally with increasing number of objects and/or number of observations.

2. It does not leverage negative detections.

3. It does not produce estimates in real-time (i.e. milliseconds).

In contrast, this research pursues a fundamental shift away from statistical filters to attempt to achieve the following technical objectives, as seen in Table 3.1.

Table 3.1: Technical Challenges

<table>
<thead>
<tr>
<th>Objective</th>
<th>Real-Time Domain Awareness</th>
</tr>
</thead>
<tbody>
<tr>
<td>Generate real-time adaptive population distribution and state estimates for all regions of interest.</td>
<td></td>
</tr>
<tr>
<td>Approach</td>
<td>Apply computer vision techniques combined with computational intelligence to estimate domain population and to perform data association, initial orbit determination and state estimate updates.</td>
</tr>
<tr>
<td>Scalability</td>
<td>Scales linearly based on desired accuracy, database read/write performance, and computational resources.</td>
</tr>
<tr>
<td>Data Association</td>
<td>Applied in the state space for all modalities simultaneously. Each observation classified to any or all possible objects.</td>
</tr>
<tr>
<td>Fusion</td>
<td>Each detection is mapped to state space independently and leveraged to update state estimates in parallel.</td>
</tr>
<tr>
<td>Modalities</td>
<td>Positive, negative, and partial detections from RADAR, optical, passive RF, thermal, and anecdotal.</td>
</tr>
<tr>
<td>Adaptability</td>
<td>New information incorporated real-time without reprocessing of old information.</td>
</tr>
<tr>
<td>Initial Assumptions</td>
<td>No initial assumptions required, i.e. no Gaussian assumption.</td>
</tr>
</tbody>
</table>

The overall objective is to assess the full domain for number objects, regions of uncertainty, and regions where it is known that there are no objects. In addition, estimates for object trajectories with unconstrained uncertainties, i.e. irregular distributions. The approach to achieve this objective is to implement and demonstrate a combination of computer vision and computational intelligence techniques. Additionally, the technique is required to scale in performance
independently from the number of objects in the domain. The goal is to preserve real-time performance in all cases, while scaling linearly in accuracy as more processing power is made available. Another objective is to never restrict an observation to belong to a single object. An observation must be able to belong to multiple objects if there is evidence that either is possible. The approach attempts to fuse all sensors from an entire network in a state space, such as modified equinoctial element space, rather than in the measurement space, i.e. right ascension, declination, range, range rate, or others. This enables multi-modal fusion from an entire network in aggregate, rather than combining state estimates from each sensor independently. To enable real-time processing, the approach must be able to learn and correct previous decisions that no longer agree with new information. This means no global reprocessing. The fusion process must automatically make these course corrections as new information is provided. Lastly, it is required to make no initial measurement quality assumptions. It must accept data from all fidelities of information. Measurements may range in quality from human reporting of a spotted satellite to data from the most pristine radar.

3.3 Multiple Unknown Objects Transiting or Orbiting the Domain Surrounding a Single Planetary Body

This use case, while still relatively simplistic conceptually, attempts the same technical objectives from Table 3.1 as applied to multiple objects in elliptical, circular, parabolic, and/or hyperbolic orbits traversing the domain surrounding a single planetary body. The challenge is to estimate states for an unknown number of objects traversing through the orbital domain, \([x, y, z]\), over time, \(t\), at non-constant velocities \([\dot{x}, \dot{y}, \dot{z}]\). For each object, the Cartesian state vector, \(\vec{X}\), at any point in time is completely described by the following:

\[
\vec{X}_\text{obj} = \begin{bmatrix} x_{\text{obj}} \\ y_{\text{obj}} \\ z_{\text{obj}} \\ \dot{x}_{\text{obj}} \\ \dot{y}_{\text{obj}} \\ \dot{z}_{\text{obj}} \end{bmatrix}
\] (3.4)

and is subject to the following equations.

\[
\ddot{\vec{X}}_\text{obj} = \frac{-\mu \vec{X}_\text{obj}}{r_{\text{obj}}^3}
\] (3.5)

\[
r_{\text{obj}} = \sqrt{x_{\text{obj}}^2 + y_{\text{obj}}^2 + z_{\text{obj}}^2}
\] (3.6)
Individual measurement samples will be supplied in all 128 possible permutations of: $r_{obs}, \dot{r}_{obs}, \theta_{obs}, \dot{\theta}_{obs}, \phi_{obs}, \dot{\phi}_{obs}, t_{obs}$, where $r_{obs}$ is the object range, $\dot{r}_{obs}$ is object range-rate, $\theta_{obs}$ is object longitude, $\dot{\theta}_{obs}$ is object change in longitude with respect to time, $\phi_{obs}$ is object latitude, $\dot{\phi}_{obs}$ is object change in latitude with respect to time, and $t_{obs}$ is time of the measurement. These variables comprise the vector associated with each sample as indicated in Equation 3.6.

All variables are described with respect to the body centered inertial reference frame as depicted in Figure 3.4.

$$\mathbf{S}_{obs} = \begin{bmatrix} r_{obs} \\ \dot{r}_{obs} \\ \theta_{obs} \\ \dot{\theta}_{obs} \\ \phi_{obs} \\ \dot{\phi}_{obs} \\ t_{obs} \\ p_{obs} \end{bmatrix} \quad (3.7)$$

With each measurement, the associated maximum uncertainty is also provided for each parameter.
From this summary, many of these observation types do not meet the observability criterion without accumulating multiple observations that have been associated as part of a single track. The observability criterion requires that equivalent number of independent dimensions be observed to estimate a state of $N$ dimensions. The presented approach meets the observability criterion by bounding the domain. This enables mapping a single observation that does not meet the observability criterion into the state space by sampling within the bounds of the domain.
4.1 Computer Vision for Object Tracking

To best explain the motivation for this approach, it is best to first provide some top-level background on computer vision and explain a simple example of a computer vision solution. Computer vision is a field that includes methods for acquiring, processing, analyzing, and understanding high-dimensional data from the real-world in order to produce information, in the forms of decisions [30][31][32][33].

4.1.1 Detecting 2D Lines in an Image

A very simple example of computer vision is the challenge of detecting an unknown number of lines in an image. A human can easily look at Figure 4.1 and determine that there are 3 lines, and could fairly easily trace out the path of each line. The challenge is to automate that process.

Fortunately, in 1962, Hough patented a simple line of code that enables a simple process for detecting a two-dimensional line in a two-dimensional image [34]. The way he solved this problem was by realizing that he could exploit the fact that he understood the dynamics of the system. He realized that any single line could be completely described by the distance and angle of the line that extended from the origin to the perpendicular intersection of the line. In Figure 4.2, three additional lines are overlaid on the original lines. These three lines completely describe each line uniquely in terms of the length and the angle of the colored lines. This length is commonly referred to as the radius, \( \rho \). In addition, each line is also described by the angle which each
colored line is rotated from the horizontal with the origin on the left side of the line. This angle is denoted as $\zeta$.

The goal is to determine the radius and the theta of each colored line. This is achieved using a very simple algorithm known as the **Hough Transform** [34]. The processing logic of the **Hough Transform** is detailed in Figure 4.3 and Algorithm 4.1. These detail how the **Hough Transform** iterates through an image and possible lines to transform the image into the Hough space where lines are represented in a discrete location.
Algorithm 4.1. Hough Transform Pseudo code

01. `IMAGE = load(frame);`
02. `HT = zeros;`
03. For `x_{pix} = 1 : N_x`
  04.   For `y_{pix} = 1 : N_y`
  05.     For `ζ = 0 : step : π`
  06.       `ρ = x_{pix} * cos(ζ) + y_{pix} * sin(ζ);`
  07.       `HT(ζ, ρ) = HT(ζ, ρ) + IMAGE(x_{pix}, y_{pix});`
  08.     End
  09.   End
10. End

This simple algorithm sequentially visits each individual pixel, and considers each possible value of `ζ` to compute the `ρ` at that pixel for each possible `ζ`. If the value of that pixel is 1, it adds a value of 1 to the Hough space, `HT` matrix, at the corresponding `r` and `ζ`. As a result, each individual pixel is represented as a line in the Hough space. Stepping through this process for the entire image, the result is a Hough space matrix, `HT` as depicted in Figure 4.4.

It becomes very clear that there are three “warped bow-tie” structures with distinct peaks in the center of each “warped bow-tie.” These peaks correspond to the `ρ` and `ζ` that completely describe each of the three lines. To understand how the rest of this research works, it is important to understand the concept of a Hough Transform. The transform has defined a state space, the Hough space, where each line can be described uniquely as a single point in that space `ρ` and `ζ`. This was possible because the algorithm knows it is looking for straight lines. If it were to fol-
following the same exact process, but instead looking for circles, it would not find them. The power of a Hough Transform shines when there is an exact equation that can roughly approximate the feature the algorithm must extract. In this case it is a line.

In addition, by transforming Cartesian space into the Hough space the signal to noise ratio drastically increases. This is because the transform co-adds each pixel from the line itself into a single point. Therefore by performing peak detection in the Hough space, it is possible to detect lines that would be completely undetected due to noise in the Cartesian space. In Figures 4.5-4.7, the noise is increased to demonstrate this improvement in signal to noise ratio.

In Figure 4.5(a), a human can still fairly simply trace out the three lines in the image. In Figure 4.5(b), it is still a trivial peak detection task. In Figure 4.6(a) a human might struggle to find the three, lines but it could be argued that this is still an achievable task. In Figure 4.6(b), there is more noise, however the peaks are still quite obvious. In Figure 4.7(a) the task of determining even the number of lines appears impossible, since the image appears as random noise, however when transformed in Figure 4.7(b), three distinct peaks are quite apparent.

4.1.2 Unmodified Hough Transform Advantages Summary

In essence the Hough Transform is an algorithm to find known features in a gridded domain such as lines in a 2D image. It not only automates a deceptively complicated task which may seem trivial to the human mind, but also surpasses the human mind’s ability to perform the task in high noise situations. The challenge next becomes, how best can these techniques be applied
to the challenge of object tracking. To simplify this explanation, it will be applied to the simple case from Chapter 3, before applying it to the space domain.
4.1.3 Applying the Unmodified 2D Hough Transform to 1D Object Tracking

Since the Hough Transform was patented in 1962, numerous applications have been developed for detecting a wide variety of structures in multi-dimensional data [35][36]. Later in Chapter 7, this thesis will discuss how a similar process can be followed to apply the same top level concept to the space domain. For now, this Chapter will focus on the simple example of tracking multiple objects traveling in one-dimension with constant velocity. Looking back at Figure 1.1, it is clear that an object’s path in position and time and constant velocity is simply a line. Since this is true, domain awareness is essentially just finding where there are and are not lines in a 2D image. In fact, in section 4.1.1 we just solved the problem using a batch process to completely characterize the domain. The possibilities have been stacked to fully assess the domain in the Hough state space with improved Signal to Noise Ratio, SNR, versus detecting objects in the measurement space. Recalling from Chapter 3, Figure 4.8 describes the domain truth.

![Figure 4.8: Truth Locations In one-dimension Over Time for Three Ghosts](image)

Also recalling from Chapter 3, the full state of each object, or ghost, can be completely described by the state vector:

\[
\bar{X} = \begin{bmatrix} x_0 \\ \dot{x}_0 \end{bmatrix}
\]  

(3.1)

and is subject to the following equation.
\[ x = x_0 + \dot{x}_0 t \] 

(3.2)

at any point in time. However, the probability of object existence is only sampled in position, \( x \),
and time, \( t \).

\[
\begin{bmatrix}
S \\
U_{Max}
\end{bmatrix}
= 
\begin{bmatrix}
x \\
t \\
p
\end{bmatrix},
\begin{bmatrix}
u_x \\
u_t \\
u_p
\end{bmatrix}
\] 

(3.3)

The **Hough Transform** can be utilized to simultaneously estimate the number of objects and velocity term of each object, without ever making a binary data-object association determination. Each observation can be transformed into the Hough space with no data association and thus not meeting the observability criterion. After completing the transform a simple peak detection algorithm can be run to find the three peaks and quantify the spread of the peaks to estimate uncertainty. With a simple threshold filter, the Hough states where it is believed that there is not a line, can be easily determined. This information can all be mapped back into the Cartesian space for additional investigation if necessary. This may be necessary in the case of domain awareness. The user may be interested in determining if two objects are going to “collide.” This requires mapping the state estimates and associated uncertainties back to the Cartesian domain via simple propagation. In summary, once the **Hough Transform** is understood, it is a straightforward application to the one-dimensional object tracking problem that simultaneously:

1. bypasses the requirement of data association,
2. estimates the number of objects,
3. estimates uncertainties in state estimates,
4. estimates where there is, is not, and where it is unknown if there is an object,
5. delays the observability criterion requirement, and
6. improves SNR.

### 4.1.4 Unmodified Hough Transform Disadvantages Summary:

The previous section explained how computer vision is applied to constant velocity object tracking in one-dimensional with no modifications to the generalized **Hough Transform**. However, the unmodified approach is far from ideal for three primary reasons:
1. It is a batch process, which means it is not real-time.

2. It is a gridded approach, which means that the measurements must be quantized uniformly, resulting in unnecessary additional noise.

3. It does not account for the known measurement uncertainties.

4.2 Adapting the Hough Transform for Real-time Domain Awareness

4.2.1 Real-time Objective

Real-time domain awareness to many decision makers is the ultimate objective for information fusion research. A real-time process, “controls an environment by receiving data, processing them, and returning the results sufficiently quickly to affect the environment at that time.” [37] In common environments “sufficiently quickly” is defined on the range of nanosecond, to millisecond updates [37]. Since real-time is a core requirement of this research, it was necessary to have an independent function that ensured the real-time requirement was met or exceeded independent of accuracy and cost of the computational architecture. This is the fundamental difference between the objectives for this research and other more traditional approaches. Figure 4.9 contrasts this approach compared to traditional approaches in the form of Venn Diagram

![Good, Fast, or Cheap? - Pick Two](Image)

Figure 4.9: Trade Space Comparison

This research is not the first effort to attempt to develop a real-time version of the Hough Transform [40]. The Hough Transform is notoriously computationally expensive. Since many applications are for autonomous vehicles that need to make decisions in real-time reducing the computational cost is of high importance. This minimizes the computational resources required of the vehicle. A common approach is to only sample a subset of the pixels in a two-dimensional
image, rather than compute every possibility. In Figures 4.10 and 4.11, two examples of this are shown, where approximately half of the pixels are not considered in the transform. Figure 4.10 is filtering as if looking through a screen mesh, and only the views in between are considered. Figure 4.11 is filtering by randomly selecting approximately half of the pixels. An important takeaway from these two figures is noticing the artifacts created by the approach in Figure 4.10. There are number peaks, that while in this example are below the three peaks, in other examples could create false peaks.

From these examples two primary things are observed:

1. It is possible to pull a finite number of samples and still recover truth.
2. It is necessary to pull as many samples as possible as uniformly as possible to prevent false detections.

Based on this, it is possible to provide estimates in real-time, but monitoring the solution as
more samples are generated. As time moves forward solutions improve and are only limited by computational resources. Accuracy and processing time is independent of the number of objects, ensuring scalability to any population size.

4.2.2 Non-Gridded Objective

Recalling from Figures 3.2 and 3.3, it is quite obvious that observations are not provided in a gridded fashion. Therefore, in order to leverage the Unmodified Hough Transform, it is necessary to bin the observations resulting in additional unwanted noise. Once the measurement domain is gridded, more noise is added by again, gridding the transform into the Hough space.

It is possible to completely bypass the gridding process altogether. Instead of binning the measurements, each measurement can be sampled and directly mapped to the Hough space with a finite number of samples. Choosing the “correct” number of samples is a subject of strong debate in the academic community. Typically the “correct” or “sufficient” number of samples is dictated by accuracy requirements. Remembering the requirements communicated in Figure 4.8, the computational resources currently available should determine the number of samples. It is important to emphasize the strong difference in approach on this issue. Rather than pull sufficient samples to preserve accuracy and sacrifice real-time, the algorithm will only generate the number of samples that ensure and enable real-time performance.

4.2.3 Computational Intelligence for Real-Time Non-Gridded Sample Generation

In order to achieve the objectives laid out in Sections 4.2.1 and 4.2.2, it is necessary to develop an algorithm capable of monitoring when the computational architecture is capable of ingesting new discretized samples while maintaining real-time performance. In essence this algorithm possesses the computational intelligence to protect the architecture from “choking” on the addition of new samples. As a result, this algorithm has been named, the Guardian Of the Discretization of Samples (GODS). The sole function of this algorithm is to determine how many samples the rest of the architecture can handle by monitoring system performance. If the criteria for new samples has been met, the GODS calls the Maker of Unique Samples for Estimation (MUSE) algorithm to create new samples. If the criteria has been met to deem the sample distribution temporally unbalanced, the GODS calls to the Liaison for the Extraction of The Historical Evidence (LETHE) to extract samples from the state space. As with all of the algorithms in this research, the algorithm for each function is intentionally conceptually simple by design. This is a deliberate choice to ensure real-time updates.

4.2.3.1 Sample Generation

In the context of this research, sample generation refers the process of discretizing an uncertainty region into a finite number of samples. Figures 4.12(a)-(d) shows a simple example of the probability distribution function sampling. In this example, there are two measurements each with an
uncertainty region. It is possible to randomly generate a finite number of samples in an attempt to describe the uncertainty region. In this simple example a uniform probability distribution function (PDF) can be assumed, and to leverage a simple uniform random generator, however in real world applications it is likely that non-uniform PDFs will be available. In this case, it is best to use another type of approach such as a Markov Chain Monte Carlo approach [39].

Figures 4.12(b)-(d) show the same two uncertainty regions with increasing number of samples. It is clear that more samples more accurately represents the uncertainty regions, however there is an obvious price to be paid: computational time. There are many theories on the best ways to determine the optimal number of samples, however these all approach the challenge with accuracy as the primary requirement. The GODS algorithm takes a very different approach. It assesses in real-time how many samples the system can handle without saturating the system. In essence, this is one of the two key pillars to this approach. Real time is enabled by dynamically scaling the sample size based on system performance.

Figure 4.12: Simple Sampling Example
4.2.3.2 The GODS Algorithm

The Guardian Of the Discretization of Samples (The GODS) algorithm (Figure 4.13) dynamically scales the number of requested and retained samples based on computational resources. It guards the system from oversampling uncertainties, where oversampling is driven by real-time requirements rather than accuracy requirements. The GODS algorithm ensures that the system does not overpopulate the database producing spurious and meaningless real-time results. The algorithm gradually accumulates samples based on the performance of other downstream algorithms. The GODS algorithm is presented in pseudo code in Algorithm 4.2.

Algorithm 4.2. The GODS Pseudo code

01. $D_w = \frac{1}{\alpha}$
02. $S_{add} = D_w$
03. $S_{del} = 1$
04. While $(1 > 0)$
05. $T_{Samp} =$ Total number of Samples
06. $Samp_{range} = \text{ceil}(D_w + \frac{T_{Samp}}{D_w})$
07. Query State Space for $Samp_{range}$ most recent samples
08. Set $N_V =$ number of Visited samples in $Samp_{range}$
09. $V_R = \frac{N_V}{Samp_{range}}$
10. If $V_R > 0.5 + \frac{0.5}{\text{Log}(D_w)}$
11. $N_S = S_{add}$
12. Authorize The MUSE to create $N_S$ new samples from the observations
13. $S_{add} = S_{add} + 1$
14. $S_{del} = 1$
15. ElseIf $V_R < 0.5 - \frac{0.5}{\text{Log}(D_w)}$ and $T_{Samp} > 1$
16. $N_S = S_{del}$
17. Authorize The LETHE to delete $N_S$ old samples
18. $S_{add} = D_w$
19. $S_{del} = S_{del} + 1$
20. End
21. End

There are only three user defined parameters in the entire architecture of this research: maximum relevant state estimate resolution, $\overline{Res}_{max}$, minimum relevant state estimate resolution, $\overline{Res}_{min}$, and the level of significance, $\alpha$. The GODS utilizes one of these three parameters: the level of significance, $\alpha$. The other two parameters will be discussed in detail in later sections. The user may use any value for the level of significance however $\alpha = 0.05$ or lower is strongly recommended. A value of $\alpha = 0.02$ will be used for all examples in this thesis. This value is
chosen based on the student’s t-distribution requirement of a minimum of 30 data points. An \( \alpha = 0.02 \) corresponds to a degree-of-wisdom, \( D_w = 50 \). This parameter, explained in more detail below, will drive the algorithms to maintain sample sizes of approximately 50 data points [54].

In line 01 of the **GODS** algorithm the first step is to compute a term called the degree-of-wisdom, \( D_w \). This value is leveraged throughout the architecture and will be consistent throughout the entire system. It is the primary parameter that dictates the balance between the speed of the algorithm updates and the confidence in the real-time estimates. The smaller the degree-of-wisdom, the faster the solution updates of lower confidence will be estimated. The larger the degree-of-wisdom, the slower the solution updates of higher confidence will be estimated. In the **GODS** algorithm this value ultimately initializes how many new samples are requested and how often. Since \( \alpha = 0.02 \), then the degree-of-wisdom, \( D_w = 50 \). In line 02 the \( D_w \) is leveraged to initialize the number of new samples to generate when authorized. In line 03 the number of samples to delete are initialized with the value of one. This means that the initialization of the number of added samples is always larger than the initialization of the number of deleted samples. This is done deliberately. It encourages the algorithm to grab more evidence in a controlled manner, but also gives it time to digest the new evidence before requiring the removal of evidence. A corollary would be to have time to study before taking a test, while at the same time, not allowing the test to cover every fact ever recorded.

Next, in line 04, the algorithm enters an infinite loop. The only reason to stop the loop is if no additional accuracy is required and no additional data is to be provided. In line 05 the total number of existing samples is queried from the database into \( T_{Samp} \). Then in line 06, the **GODS** computes the number of the most recent samples to leverage, \( Samp_{range} \), for evaluating **BattliaInfinitum**’s performance (see Chapter 5). In lines 07-09, the database is queried to find the \( Samp_{range} \) most recent samples and compute the visit ratio, \( V_R \). In later sections it will be discussed in detail, what it means for a sample to be “visited.” At the top level, it means that
the other algorithms, to be detailed in the next chapter, have analyzed the sample at least once.

In line 10, the algorithm enters a simple if statement. If the visit ratio, $V_R$, is large enough, the creation of more samples is authorized. The visit ratio is “large enough” if $V_R > 0.5 + \frac{0.5}{\log(D_w)}$. Otherwise, if the visit ratio, $V_R$, is small enough, the removal of samples is authorized. The visit ratio is “small enough” if $V_R < 0.5 - \frac{0.5}{\log(D_w)}$. These if statements can be visualized as a function of the degree-of-wisdom and the threshold visit ratio is plotted in Figure 4.14. The threshold criteria is specifically designed to generate and delete samples more often with higher degree-of-wisdom. Thus allowing the algorithm to learn more frequently.

![Visit Ratio Threshold versus Degree-of-Wisdom](image)

Figure 4.14: Visit Ratio Threshold versus Degree-of-Wisdom

Assuming the criteria has been met to authorize new samples, the **GODS** authorize the **MUSE** to make $N_S$ new samples. Note that the **MUSE** will be discussed in detail in a following section. The number of new samples to be created is determined by the simple relationship of $N_S = \frac{N_V \log(D_w)}{\log(D_w)}$. This relationship can be seen in Figure 4.12. Once the criteria has been met, the number of new samples, $N_S$ is tempered by both the current visit ratio as well as the degree-of-wisdom. So, the lower the visit ratio and the higher the degree-of-wisdom, a lesser number of samples will be authorized for creation. In contrast when the visit ratio is high and the degree-of-wisdom is low, a higher number of samples will be created. This approach incorporates the computational intelligence algorithm to be “self-aware” of the confidence derived from the degree-of-wisdom. The greater the value of the degree-of-wisdom, the more “willing” the **GODS** is to manipulate the samples by authorizing the **MUSE** or the **LETHE**. The less “wise” the sampling is, the less “willing” the **GODS** are to authorize changes.

It is important to point out that this is contrary to the normal approach. It is typically highly discouraged to ever remove samples from consideration. By definition it will likely result in a loss of accuracy, however this is not necessarily true when multiple observations are available.
Since the system is primarily designed for real-time performance, there are a finite number of samples that can be retained over a given time window. As a result, if some older samples are not “forgotten,” then new samples cannot be created without compromising solution stability or real-time performance. The word “forgotten” is used in quotes to emphasize that it is not necessary to delete previous samples, however it is required to archive them in another system. The reasons for this will be explained in later sections when discussing the computational architecture.

While this section has explained how and why the GODS algorithm is structured, there are two additional sub functions which require explanation: the MUSE, and the LETHE. These algorithms are responsible for making or forgetting new samples once authorized by the GODS algorithm, as explained previously.

4.2.3.3 The MUSE Algorithm

As explained in the previous section, once certain criteria have been met, the GODS algorithm calls the the MUSE algorithm to make $N_S$ new samples. As previously mentioned, the MUSE acronym, aptly named, stands for the Maker of Unique Samples for Estimation. This section will discuss how this process is performed first, agnostically to the domain, then specific to the one-dimensional constant velocity problem discussed in the last chapter. The MUSE algorithms purpose is to intelligently query the observation space, and to create a finite number of new samples into the state space. The processing logic flow is shown in Figure 4.15.

Figure 4.15: MUSE Processing Flow
Algorithm 4.3. The MUSE Pseudo code

01. \( C_S = 0 \)
02. Query observation space for \( \max(\mathbf{U}_{\text{Max}}) \)
03. Query observation space for \( \min(\mathbf{U}_{\text{Max}}) \)
04. While \( N_S > C_S \)
05. \( U_{\text{Filt}} = \min(\text{randn}(1)) \times (\max(\mathbf{U}_{\text{Max}}) - \min(\mathbf{U}_{\text{Max}})) + \min(\mathbf{U}_{\text{Max}}) \)
06. Query for observations, \( \mathbf{O} \), with Sample Number \( S = S_{\text{min}} \) and \( \mathbf{U}_{\text{Max}} < U_{\text{Filt}} \)
07. Query State Space for Minimum Sample Number, \( S_{\text{min}} \)
08. If \( \text{length}(\mathbf{O}) > 0 \)
09. Choose \( \text{idx} = \text{ceil}(\text{rand}(1) \times \text{length}(\mathbf{O})) \)
10. Load \( \mathbf{X} \) from \( \mathbf{O}(\text{idx}) \) in the state space to include \( P_{\text{Obs}}, U_{\text{Max}} \)
11. \( U_{\text{MaxNorm}} = \frac{U_{\text{Max}}}{D_{\text{range}}} \)
12. \( P_{\text{Samp}} = \text{probDeWeight}(P_{\text{Obs}}, 1 + |U_{\text{MaxNorm}}| \times D_W) \)
13. Make \( N_S \) random samples in state space with \( P_{\text{Samp}} \)
14. \( C_S = C_S + 1 \)
15. End
16. End
17. Function \( P = \text{probDeWeight}(P,w) \)
18. \( n = \frac{1}{w} \)
19. \( P = \frac{P^n}{P^n + (1-P)^n} \)
20. End

The MUSE first initializes \( C_S = 0 \) to indicate that zero samples have been created for the latest request from the GODS. Next in lines 02-03, the bounds of the uncertainty vectors are queried. Until the MUSE has generated the requested number of samples \( N_S \), it will iteratively continue to execute lines 05-14. In line 05, a vector, \( \mathbf{U}_{\text{Filt}} \) is initialized for randomly generating an uncertainty vector within the bounds of the extreme. For uniform uncertainty data, this is not a required parameter, however it is necessary to prioritize pulling samples from observations of varying uncertainty ranges. The prioritization of which parameters to pull samples from is done using a simple C++ Armadillo library randomized Gaussian function \( \text{randn}(1) \).

This relationship can be seen in the Figure 4.16. We see that the smaller the uncertainty in a given observation, the higher the probability that it will be randomly selected for additional samples. Inversely the opposite is true. The observations with the highest uncertainty will almost never be selected for generating new samples. This relationship could be changed to anything the user identifies to best meet their needs, (i.e. uniform sampling, linear sampling, etc.).

Line 06 queries the observation space to find all observations that are insufficiently sampled and also are more accurate than the randomized filter determined in line 05. If any observations are found meeting this criterion, then lines 09-14 create a single new sample using one
randomly selected observation from the observations returned in line 06. Before this step, in line 07, the state space is queried to find the minimum sample number, $S_{min}$, to be used in later steps.

The first step is to randomly select one of the observations in line 09. Line 10 simply loads the relevant variables from the selected observation. In line 11 the uncertainty vector is then normalized by the full range of the domain, $D_{Range}$, in each dimension to compute $U_{MaxNorm}$. Line 12, is responsible for de-weighting probabilities based on the relationship of the norm of the normalized uncertainty vector. Lines 17-20 detail this simple function called probDeWeight. Using this function in Line 12 de-weights samples based on how large the uncertainty is. If the uncertainty is zero, then the probability $P_{Samp} = P_{Obs}$, however as the uncertainty grows, the probability of the sample is de-weighted closer to probability of 0.5. This relationship is shown in Figure 4.17.

The next step generates a random sample in the state space based on the uncertainties of the chosen observation. This is achieved by randomly choosing parameters with the uncertainty of all measured parameters and within the defined domain for all unobserved remaining dimensions. Line 12 is where the computer vision technique, Hough Transform, is leveraged. This will be discussed more in a later section. Lastly, line 14 simply adds an increment of 1 to the count of total number of pulled samples. Thus concludes the general domain agnostic MUSE algorithm. By following this procedure, the MUSE has created additional samples by prioritizing which observations to pull from based on simple logical relationships leveraging easily queried variables.
4.2.3.4 The LETHE Algorithm

The Liason for the Extraction of The Historical Evidence (the LETHE) algorithm, serves two primary purposes:

1. ensuring that new observations are rapidly sampled, and
2. ensuring that observations are evenly sampled.

This is achieved combinatorially by the logic built into the GODS algorithm and the LETHE algorithm. The GODS algorithm monitors the sample numbers to determine when to authorize the LETHE, and the LETHE identifies the oldest samples and removes them from consideration. This can be done either by deleting or archiving the removed samples. If possible, archival is always preferred to deletion. This is a very simple algorithm. The purpose of this function is to remove samples that are the oldest in the system. The purpose is to free up system services for new samples from new observations, and preserve overall performance and sample distributions. The processing flow logic is shown in Figure 4.18.
Algorithm 4.4. The LETHE Pseudo code
01. For $1 \rightarrow N_S$
02. Query State Space for Minimum Global Sample Number, $G_{S_{min}}$
03. Remove sample with global sample number $GS = G_{S_{min}}$
04. End

4.2.4 Applying The GODS, The MUSE, and The LETHE

This section will discuss how to apply these three algorithms to a one-dimensional domain with objects moving at constant velocity. The goal is to take observations with probabilities and uncertainties and to create finite numbers of samples in real-time without saturating the performance of downstream algorithms. To apply these three algorithms to any domain only requires additional math in one line of the MUSE algorithm. In Algorithm 4.2, line 13, is where this logic is added. In the case of a one-dimensional domain with objects moving at constant velocity, this is simply a Hough Transform described in Algorithm 4.1 with a very slight change. The Hough Transform is a gridded approach. Rather than gridding the problem, the objective is to create a finite number of samples. In this case, given:

$$\begin{bmatrix} x \\ t \\ p \end{bmatrix}, \begin{bmatrix} u_x \\ u_t \\ u_p \end{bmatrix}$$

(3.2)

the objective is to create $N_S$ samples. Thus Algorithm 4.1 becomes:

Algorithm 4.5. Probabilistic Random Sampling Hough Transform
01. $x_{samp} = x_{obs} + \text{rand}(1) * (0.5 * u_x)$
02. $t_{samp} = t_{obs} + \text{rand}(1) * (0.5 * u_t)$
03. $p_{samp} = p_{obs} + \text{rand}(1) * (0.5 * u_p)$
04. $\zeta_{samp} = \pi * \text{rand}(1)$
05. $\rho_{samp} = x_{samp} * cos(\zeta_{samp}) + t_{samp} * sin(\zeta_{samp})$;

Lines 01-04 generate a random sample for $x$, $t$, $p$, and $\zeta$, to compute the corresponding $\rho$ value if all of those randomized parameters were true.

Figure 4.19 is a simple example of this type of sampling from observations. There are three observations that form a line, shown as a dashed line in the observation space. Each observation is sampled $N$ number of times in the Hough space. In this simple example three cases are shown, when $N = 15, 50$, or $150$. The dashed circle represents the point or cluster of highest density, which corresponds to an exact solution for the line in the observation space. This is a simple density based implementation.
Density is the only consideration if only positive detections are provided, however, in the probabilistic Random Sampling Hough Transform, all observations are provided rather than just positive detections. As a result the criteria shifts to regions of high cumulative probability. Chapter 3 described a simple use case for this example.

Three ghosts are traversing a one-dimensional domain, while being measured. This results in observations of both high and low probability that a ghost was observed. This is shown in Figure 4.20. These observations can then be sampled by the GODS, MUSE, and LETHE to populate the Hough space. Figure 4.21 shows an example where both positive and negative observations are transformed into the Hough space by the approach described in this chapter.

There are distinct differences between this approach and the standard gridded Hough Transform shown in Figure 4.4. Rather than gridded pixels there are a finite number of samples in the observation space, shown in Figure 4.21 on the left. Each observation has a corresponding finite number of samples in the Hough space, shown in Figure 4.21 on the right. Upon close inspection it can be visually determined there are three dense regions of high probability in the Hough space. These dense regions are circled in Figure 4.22. The objective becomes to automatically assess how many of these dense regions exist, where they are located in the Hough space, and...
where it is known that there are not any dense regions. This enables a user to understand where objects have been and will be, where it is unknown if there are objects, and where it is unknown.

4.3 Summary and Next Steps

In summary, this section detailed a process by which any domain can be sampled in real-time on any computational architecture by leveraging computer vision and computational intelligence. At this point, it has been explained how the domain can be sampled, but not how to analyze it in real time. On multiple occasions, samples were referred to as “visited,” and “downstream algorithms” were mentioned. The next section will discuss this collection of parallelized algorithms, what it means for a sample to be “visited,” and how all the algorithms run in parallel together to make decisions in real-time. The solution to this technical challenge of detecting dense regions of high probability in real-time with live streaming observations is discussed in Chapter 5. The challenge in Chapter 5 is to produce the estimates shown in Figure 4.22, leveraging the observations shown in Figure 4.20.
Figure 4.21: Observations and Corresponding Generated Samples

Figure 4.22: Observations and Corresponding Generated Samples With Truth
Chapter 5

BattaliaInfinitum: A real-time probabilistic computational intelligence cluster detection algorithm

5.1 Introduction

Chapter 4 presented the motivation for the technical challenge of detecting dense regions of high probability in a distribution of probability samples in real-time. This objective can be described real-time probabilistic cluster detection. Real-time probabilistic cluster detection on dynamic, live streaming, sample distributions poses a significant challenge for decision makers. An ideal algorithm always provides a human decision maker real-time awareness of the best hypotheses given all available information and computational resources. These hypotheses should update without re-processing the entire distribution in batch. It should not become trapped in local minima and should adapt to dynamic, or live-streaming, distributions. To achieve these objectives a real-time approach which learns from its previous hypotheses, but also forgets information that is no longer part of the sample is required.

This chapter discusses a suite of parallel algorithms developed based on these requirements titled BattaliaInfinitum. The algorithm suite is named BattaliaInfinitum due to the structure of the approach. The technique relies on the concept of competing logic that infinitely refines solutions. The net effect is an “infinite battle” between competing algorithms. The ecosystem of algorithms independently operate in parallel on a “virtual battlefield.” The word battalia is a 16th century term referencing an arrangement of troops on a battlefield [55]. Since the approach
will continuously refine the solution, a user would only turn it off if the solutions are no longer worth allocating compute resources to the domain. This means that the approach is effectively infinite, as in it can be run forever on a domain without ever needing to be re-initialized. Infinitum is a Latin term which means to infinity; endlessly; without limit [56]. These two words combined effectively describe the approach design, thus the technique is called BattaliaInfinitum.

Each algorithm has a very simple decision it attempts to make. These objectives effectively compete, or battle, in parallel to determine hypotheses in real time about the presence of clusters, no clusters, or insufficient information to make a decision. This process does not have a discrete point of convergence since it is developed for live-streaming information with changing states, and to be run in parallel with the algorithms from Chapter 4: the GODS, the MUSE, and the LETHE. As a result, the algorithm is designed to infinitely improve the solution the longer it continues to run. For the space application live-streaming information would include information such as optical, RADAR, or passive RF measurements, as well as any soft data (i.e. anecdotal) and any available orbital state information.

The approach is designed to produce results to the user after only a single iteration, providing real-time updates that refine the estimates. Each iteration benefits from all relevant previous iterations with no impact to the speed of the current iteration. It is easily parallelized with a linear increase in speed with respect to number of workers and adapts to dynamic, or real-time, samples without saturating in local minima. BattaliaInfinitum runs on a single laptop, but can handle any distribution size which the computer disk/memory can support. The supported data volume improves drastically if accessing the distribution from a database rather than loading it into memory.

BattaliaInfinitum was originally conceived to enable real-time object tracking, as described in Chapter 4. In this domain, measurements and soft information may be dynamically added and removed in real-time ($\ll 1$ ms per addition or removal).

### 5.2 Generalized DBSCAN

A trade study performed did not identify any high-speed real-time cluster detection algorithms that were easily parallelized and extensible to cumulative probabilities and N-dimensional problems. However, the technical philosophy behind density-based space clustering of applications with noise (DBSCAN), did have several desired qualities [41]. For this reason the DBSCAN model was used as the inspiration for BattaliaInfinitum. Before discussing the technical approach details of BattaliaInfinitum, it is important to be comfortable with the specifics of how DBSCAN performs cluster detection of irregular clusters. At a high-level, DBSCAN works by
looking for clusters of points in a distribution based on the density in a given region. There are only three inputs to the generalized DBSCAN algorithm to be further explained later in this section:

1. distribution of point locations,
2. ring distance, and
3. order the points should be visited.

In Figure 5.1, the first two inputs are shown for a small distribution of two-dimensional points. The objective is to find:

1. the unique clusters,
2. the points associated with each unique cluster, and
3. the points not associated with any of the unique clusters.

These outputs are also plotted in Figure 5.1 on the right plot. Colored points represent points in clusters, and gray points are points not belonging to clusters.

Figure 5.1: DBSCAN Inputs and Outputs

The DBSCAN algorithm is a simple algorithm. It sequentially visits each point in the distribution, and checks for nearby points. If there are any points within the specified radius of the visited point, they are considered to belong to a cluster. Each point in the new cluster is visited to determine if there are any points adjacent to those new points. If so, those points are also added to the cluster. This process is summarized on the simple example in Figure 5.2, specifically in boxes 1, 2, and 3. The solid line represents the first point visited, and the dashed ring represents each added new point. Box 4 demonstrates that there are outliers that cannot be associated to any other points. These are marked in gray in box 5. The algorithm calls a point “visited” once it has been checked for nearby points. The algorithm does not stop until all points have been flagged as visited. This point is represented in Figure 5.2 box 5.
There are many advantages to the **DBSCAN** algorithm, but it requires significant modifications to analyze in real-time the streaming probabilistic sample distribution described in Chapter 4.

### 5.3 BattaliaInfinitum Technical Approach

As discussed in the previous section, **BattaliaInfinitum** was inspired by **DBSCAN**, however there are significant modifications and additions required to leverage the approach for the objectives of this research. Both approaches are:

1. capable of cluster detection/classification,
2. are simple logic to enable high-speed performance,
3. can find arbitrary cluster shapes, and
4. are extensible to N-dimensional distributions.

However **DBSCAN** as formulated:

1. does not classify based on cumulative probabilities,
2. is a batch process,
3. cannot adapt to changes in dynamic distributions,
4. is dependent purely on the sample distribution location density, and
5. requires sophisticated managers to parallelize which scale non-linearly.
Table 5.1 outlines the key differences between BattaliaInfinitum and DBSCAN.

Table 5.1: DBSCAN and BattaliaInfinitum Requirements Compliance Comparison

<table>
<thead>
<tr>
<th>Requirements</th>
<th>DBSCAN</th>
<th>BattaliaInfinitum</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cluster Detection</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Cluster Classification</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Simple Algorithms for High Speed Performance</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Arbitrary Cluster Shape Detection</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Extensible to N-Dimensional Distributions</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Cumulative Probability Criterion</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Adaptive Learning for Dynamic Distributions</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Agnostic to Sample Location Distribution</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Linear Parallelized Performance Scaling</td>
<td>No</td>
<td>Yes</td>
</tr>
</tbody>
</table>

BattaliaInfinitum is designed to achieve these objectives by competing algorithms running in parallel. There are nine algorithms running in parallel. Each algorithm can run independently, however the products produced by a single algorithm are not of value without being balanced by the other algorithms. The remainder of this chapter will sequentially step through the logic of each algorithm. The nine algorithms are named consistent with the theme of battle, and appropriately to their function. The motivations for the naming conventions of each algorithm will be discussed in each section corresponding to that algorithm. The nine algorithms are the: Recruiter, Assassin, Pacifist, Anarchist, Hippie, Rebel, Massacrist, and Historian.

Each operator working alone biases or corrupts the solution, but when operating in parallel each algorithm improves the current hypothesis by complementing the logic of the other algorithms. The top-level flow is pictorialized in Figure 5.3. There are five parallel processes, comprised of nine total algorithms.

5.3.1 Initializing BattaliaInfinitum

The user of BattaliaInfinitum must initialize the algorithm by defining three parameters:

1. the level of significance $\alpha$ (range: $0 < \alpha < 1$),
   - $\alpha$ will be leveraged in the non-central student T tests described in Section 5.3.2.

2. the minimum relevant resolution $\text{Res}_{\text{Min}}$ (units domain specific $\text{Res}_{\text{Min}} > 0$), and
   - Defines the minimum state uncertainty objective.

3. the maximum relevant resolution $\text{Res}_{\text{Max}}$ (units domain specific $\text{Res}_{\text{Max}} \geq \text{Res}_{\text{Min}} > 0$).
   - Defines the largest uncertainty accepted for individual object states.

Once these parameters are set, the BattaliaInfinitum can be initialized by Algorithm 5.1.
Algorithm 5.1. Initializing BattaliaInfinitum

01. $searchDistance = Res_{Max}$
02. $D_w = \frac{1}{\alpha}$
03. $P_c = 0.5$
04. $NP_{new} = D_w$
05. Save $searchDistance$, $D_w$, $P_c$, and $NP_{new}$ to memory or database.
06. $minClusterNumber = -1$
07. $maxClusterNumber = 1$

Line 01 initializes the $searchDistance$ as the $Res_{Max}$. The $searchDistance$ is the radius of the ring in each dimension and defines the size of the region to assess in the algorithms. It will be updated on each iteration to attempt to maintain a sample size of $D_w$ samples. Line 02, calculates the degree of wisdom $D_w$ to be used by multiple algorithms, including those outlined in Chapter 4. The value of $D_w$ will be the number of samples which BattaliaInfinitum will target for the number of required samples for any hypothesis tests. The variable $P_c$ represents the cumulative probability computed from the visited samples in a given iteration. Line 03 initializes the log of previous iterations to $P_c = 0.5$. The initial value of 0.5 is chosen since there are no previous iterations on the first iteration and at least one value is required for the SRAPH group, to be described in Section 5.3.2. Line 04 $NP_{new}$ is initialized to $D_w$. This variable represents the number of sample points found within an iteration. It is only set to the $D_w$ on the original initialization of the algorithm. This is the chosen value since this will be the target value that the algorithm will attempt to maintain as it learns from previous iterations. Line 05, saves these parameters to memory or a database for the other algorithms to access as needed. Lines 06-07 initialize the values of the minimum and maximum cluster number to -1 and 1.

5.3.2 The Strategist-Recruiter-Assassin-Pacifist-Historian (SRAPH) Algorithms

SRAPH is an acronym for the Strategist, Recruiter, Anarchist, Pacifist, and Historian algorithms. These five algorithms work as a suite of algorithms in a logical sequence. The purpose of this algorithm suite is to visit a randomly chosen region in the sample state space to determine if the samples within the region should belong to a positive or a negative cluster. If there is insufficient statistical evidence to conclude either, the region marked as unknown. Once the decision has been made, several parameters are updated and logged for future iterations.

5.3.2.1 The Strategist

The Strategist algorithm primary task is to learn from history and strategize the parameters for the iteration. Thus it is aptly named the Strategist. Before each algorithm in SRAPH is executed the Strategist must initialize multiple parameter and choose a region in the state space and computes the cumulative probability of that region. Algorithm 5.2 details this process.
Figure 5.3: BattaliaInfinitum Top Level Processing Flow
Algorithm 5.2. The Strategist

01. Load $\text{searchDistance, maxClusterNumber}$
02. $\overline{\mathbf{X}}_{\text{cent}} = \overline{SS}_{\text{range}} \ast \text{rand}(\text{size}(SS_{\text{range}}))$
03. $\overline{S}_R = \text{find}(|\overline{S} - \overline{X}_{\text{cent}}| \leq |\text{searchDistance}|)$
04. $P_c = 0.5$
05. For $n = 1 \rightarrow \text{length}([S_R])$
06. $P_c = \frac{P_c \cdot P_n + (1 - P_c) \cdot (1 - P_n)}{1 - P_n}$
07. End
08. $\overline{P}_c = \text{thelastD}_w$ most recent values of $P_c$
09. medianCumulativeProbability = $\text{median}(\overline{P}_c)$
10. meanCumulativeProbability = $\text{mean}(\overline{P}_c)$
11. stdvCumulativeProbability = $\text{stdv}(\overline{P}_c)$
12. $D_f = \text{length}(S_R)$
13. $\delta = \frac{\text{medianCumulativeProbability} - \text{meanCumulativeProbability}}{\text{stdvCumulativeProbability}}$
14. $\text{netThresh} = \frac{P_c - 0.5}{\text{stdvCumulativeProbability}}$

Line 01 first loads the current $\text{searchDistance}$ from Algorithm 5.1 or previous iterations of the Historian. Line 02 chooses a random location, $\overline{X}_{\text{cent}}$, in the state space, $\overline{SS}_{\text{range}}$. Line 03 finds all samples, $S_R$ within the $\text{searchDistance}$ of the that random location $\overline{X}_{\text{cent}}$. Lines 04-07 iteratively compute the cumulative probability, $P_c$ of all samples, $P_{1 \rightarrow n}$ in the region. Lines 08-10 compute the median, the mean, and the standard deviation of the $D_w$ most recent $P_c$ values. Line 12 defines the degrees of freedom for the statistical test as the number of samples found within the $\text{searchDistance}$. Line 13 computes the non-centrality parameter $\delta$ to be used in the non-centralized student t-test. This is a statistical test that accounts for skewed distributions. Using this type of approach also accounts for distributions when probabilistic samples are not Gaussian-distributed around probability 0.5. This is necessary to account for domains that are either sparsely populated, such as the examples in this research, or over populated. This approach also accounts for any biases in how the domain is being sampled as well.

5.3.2.2 The Recruiter

The Recruiter’s task is to find samples that should belong to a positive cluster and “recruit” them into that cluster or form a new cluster. For this reason, it is named the Recruiter. This requires a simple hypothesis test to check if we have enough statistical evidence to conclude that $P_c \geq \text{medianCumulativeProbability}$. The non-central student t-distribution hypothesis test [42] is leveraged in this approach.
Algorithm 5.3. The Recruiter

01. $nctPositiveThresh = nctRightTailed(D_f, \delta)$

02. If $nctThresh > nctPositiveThresh$ and $\text{length}(\bar{s}_R) > D_w$

03. Load $\text{clusterNumbers}$

04. $\text{idxPos} = \text{Find}(\text{clusterNumbers} > 1)$

05. If $\text{length}(\text{idxPos}) > 0$

06. $\text{modePositive} = \text{mode}(\text{clusterNumbers})$

07. Set $\text{clusterNumber}$ of all samples in region to $\text{modePositive}$

08. Else

09. Set $\text{clusterNumber}$ of all samples in region to $\text{maxClusterNumber} + 1$

10. End

11. End

Line 01 computes the right tailed non-central student t-test given the degrees of freedom $D_f$ and the non-centrality parameter $\delta$. In line 02 If $\text{length}(\bar{s}_R) \geq D_w$ and the cluster passes the test, the Recruiter checks to determine if any of the $n$ points already are assigned to another cluster with number $> 1$ in lines 03-05. If so, in lines 06-07 the $n$ points are assigned to the dominant cluster by computing the mode of the existing cluster assignments. If not, in line 09 the Recruiter creates a new cluster and assigns the $n$ points to cluster $\text{maxClusterNum} + 1$.

Figure 5.4 shows a simple distribution of probability samples of either probability equal to 0.1 or 0.9. Algorithm 5.2 has chosen a random location and computed the cumulative probability. This is seen on the left of Figure 5.4. This cumulative probability is then run through a hypothesis test to determine if there is significant statistical evidence to conclude that those samples should belong to a positive cluster. If the Recruiter’s test is passed, then those samples are assigned to a positive cluster. A positive cluster is defined as a region where the cumulative probability crosses the threshold for a high probability region. In the tracking example the positive cluster corresponds to detection of a region in the state space where an object trajectory has been detected. The representation in the samples can be seen on the right in Figure 5.4.

![Figure 5.4: The Recruiter Example: First New Cluster](image)
The example shown in Figure 5.4 captures an example of the creation of a new cluster. However another example is the second iteration when the test is again passed. The **Recruiter** also includes logic to recruit new positives into existing clusters. This is shown in Figure 5.5.

![Figure 5.5: The Recruiter Example: Add Samples to Cluster](image)

In Figure 5.5, the test for positive samples has again been passed and the **Recruiter** associates the new samples with the existing cluster rather than creating a new cluster. If the **Recruiter** condition is satisfied in the case of a new cluster, then it assigns the samples to a new cluster. The new cluster is shown in a different color in Figure 5.6.

![Figure 5.6: The Recruiter Example: Second New Cluster](image)

However, if the **Recruiter**'s test is not satisfied, then the **Assassin** is the next algorithm to analyze the sample.

### 5.3.2.3 The Assassin

The **Assassin**’s task is to find samples that should belong to the negative cluster and “assassinate” them into negative status. The cumulative probability, $P_C$, is run through a hypothesis test to determine if there is significant statistical evidence to conclude that those samples should belong to the negative cluster. This algorithm is detailed in Algorithm 5.4.
Algorithm 5.4. The Assassin

01. `nctNegativeThresh = nctLeftTailed(D_f, δ)`
02. If `nctThresh < nctNegativeThresh and length(S_R) > D_w`
03. Set `clusterNumber` of all samples in region to $-1$
04. End

Again, the approach will leverage the non-central student t-test [42]. Line 01 computes the left tailed non-central student T test threshold given the degrees of freedom $D_f$ and the non-centrality parameter $δ$. In line 02 If $length(S_R) \geq D_w$ and the cluster passes the test, the Assassin assigns the $n$ points to cluster $-1$.

Figure 5.7 shows the same simple distribution of probability samples of either probability equal to 0.1 or 0.9. Algorithm 5.2 has once again chosen a random location and computed the cumulative probability. This is seen on the left of Figure 5.7. This cumulative probability is then run through a hypothesis test to determine if there is significant statistical evidence to conclude that those samples should belong to the negative cluster. If the Assassin’s test is passed, then those samples are assigned to the negative cluster.

It is apparent that by working together the Recruiter and Assassin are classifying finite numbers of samples as belonging to positive clusters or the negative cluster. Next, the Pacifist algorithm is introduced.

5.3.2.4 The Pacifist

If neither the Recruiter’s or Assassin’s statistical tests have been passed, the Pacifist un-assigns the samples to indicate that there is insufficient statistical information to conclude if the samples belong to a positive or negative cluster.
Algorithm 5.5. The Pacifist

01. If (The Recruiter and Assassin Tests both fail)
02. Set clusterNumber of all samples in region to 0
03. End

The Pacifist can be seen in Figure 5.8 acting on the same samples shown in previous examples.

![Figure 5.8: The Pacifist Example: Unknown Cluster](image)

Assuming this process has been running for many iterations in parallel it will stabilize to a solution similar to Figure 5.9.

![Figure 5.9: The RAP Group Stable Example](image)

In this example most of the samples have been assigned to one of two positive clusters, while the rest have been assigned to the negative cluster. A few of the samples are marked unknown, since they reside on the borders between the high probability and low probability samples. These are typical and desired results when running the Recruiter, Assassin, and Pacifist (RAP) group. There is a fourth algorithm in addition to the RAP group algorithms known as the Historian.
5.3.2.5 The Historian

The Historian records the cumulative probability of the iteration and analyzes the iteration to learn from the sample to update the searchDistance for future iterations.

Algorithm 5.6. The Historian

01. \( NP_{old} = NP_{new} \)
02. \( NP_n = \text{length}(S_R) \)
03. \( NP_{new} = \frac{NP_{old} \times (D_w - 1) + NP_n}{D_w} \)
04. \( \text{searchDistance} = \frac{(D_w - 1) + \left( \frac{NP_{new}}{D_w} \right)}{D_w} \times \text{searchDistance} \)
05. If \( \text{searchDistance} < \text{Res}_{\text{min}} \)
06. \( \text{searchDistance} = \text{Res}_{\text{min}} \)
07. Else If \( \text{searchDistance} > \text{Res}_{\text{max}} \)
08. \( \text{searchDistance} = \text{Res}_{\text{max}} \)
09. End
10. Save searchDistance to database or memory

Lines 01-03 updates the mean number of samples found in the region over the last \( D_w \) iterations. In Line 04 this is used to update the recommended searchDistance based on this mean value. In lines 05-09 the Historian checks to determine if either of the predefined parameters for minimum and maximum resolution have been violated by the update, and makes the appropriate corrections if necessary. In line 10, the value is saved to the database or memory for use by the Strategist in future iterations.

5.3.2.6 SRAPH Algorithms Summary

For a static distribution the Strategist, Recruiter, Assassin, Pacifist, and Historian perform quite well on their own, however this is less the case when the distribution is dynamic. Multiple things can happen:

1. the Recruiter can be slow to recognize two clusters should be one cluster due to new samples,
2. the Recruiter may create clusters so large, that they violate the \( \text{Res}_{\text{max}} \) criteria,
3. the Recruiter can end up assigning multiple clusters to a single cluster as more samples are provided, and/or
4. the Assassin and Pacifist may reduce the number of samples making the cluster statistically insignificant creating false positive clusters due to new samples.

As a result, four additional algorithms were developed to enable real-time processing of dynamic samples: the Hippie, Anarchist, Rebel, and Massacrist. Unlike the SRAPH algorithms,
these algorithms run in parallel to each other to mitigate the shortcomings of the SRAPH algorithms.

5.3.3 The Hippie-Anarchist-Rebel-Massacrist (HARM) Algorithms

The HARM algorithms are essential to enabling BattaliaInfinitum to adapt to dynamic distributions over time, and to preventing the assertion of meaningless results. The HARM algorithms operate in parallel to each other and to the SRAPH algorithms. Unlike the SRAPH algorithms, the HARM algorithms make decisions based on the structure of the positive clusters rather than on the cumulative probability of local regions. The HARM algorithms are responsible for the real-time capability of the BattaliaInfinitum approach.

5.3.3.1 The Hippie

Algorithm 5.7 addresses the issue mentioned above: the Recruiter can be slow to recognize two clusters should be one cluster due to new samples. The objective of the first algorithm is to find separate clusters that should actually belong to the same cluster based on the distance of separation. If the clusters are close enough to “hug” they are joined into a single cluster. Since hugging is commonly associated with the term hippie, this algorithm has been named the Hippie.

Algorithm 5.7. The Hippie

01. Load $\text{searchDistance}$
02. $\text{clusterNumber}_H = \text{FindOneRandom}(\text{clusterNumber} > 1)$
03. $\vec{S}_H = \text{find}(\|\vec{S} - \vec{X}_\text{cent}\| <= |\text{searchDistance}| \text{ and } \text{clusterNumber} > 1)$
04. $\text{unqClusterNums} = \text{unique}(\text{clusterNumbers of } \vec{S}_H)$
05. If(length($\text{unqClusterNums}$) > 1)
06. Set $\text{clusterNumber}$ of all samples $\vec{S}_H$ of in region to $\vec{X}_\text{cent} \text{ clusterNumber}$
07. End

Line 01 loads the current $\text{searchDistance}$ for the algorithm. Line 02 chooses a random sample from the clusters with a cluster number greater than one. Then in line 03, the Hippie finds all samples with cluster number greater than 1 within the current search distance. In lines 04-05, the Hippie determines if there are multiple clusters within the region. If so, then line 06 updates all of the samples from any clusters with samples within the search distance to the cluster number of the originally chosen sample. This effectively joins the cluster. This can be seen pictorially in Figure 5.10.

In Figure 5.10, on the left side there two clusters, green and magenta, which both lie within the region of the Hippie search distance. On the right, it can be seen that due to this, the magenta cluster has been joined to the green cluster.
5.3.3.2 The Anarchist

Algorithm 5.8 addresses the next issue mentioned above: the Recruiter may create clusters so large, that they violate the $\text{Res}_{\text{max}}$ criteria. When this criteria is violated, the cluster is deemed an oversized cluster. The Anarchist randomly visits positive clusters to assess whether or not the cluster has grown too large to be relevant. It effectively marks positive samples but marks them as irrelevant. This algorithm is named the Anarchist to indicate that the cluster is akin to a group with no overarching affiliation. To achieve this the Anarchist assesses the weighted standard deviation of uncertainty in each dimension. If any have grown larger than the $\text{Res}_{\text{Max}}$, the cluster number is set equal to $\text{minClusterNumber} - 1$. This tells the algorithm, that there is statistical evidence that a cluster or clusters exist in this region, however the uncertainty is too large to assert uniqueness. This becomes important for reasons discussed in later chapters specifically related to applications of BattaliaInifinitum to object tracking.

**Algorithm 5.8. The Anarchist**

01. Load $\text{Res}_{\text{Max}}$, and $\text{minClusterNumber}$
02. $\text{clusterNumber}_A = \text{FindOneRandom}(\text{clusterNumber} > 1)$
03. $\vec{S}_A = \text{Find}(\text{clusterNumber} = \text{clusterNumber}_A)$
04. $\vec{X}_{\text{stdv}} = \text{stdvWeighted}(\vec{S}_A)$
05. If($\vec{X}_{\text{stdv}} > \text{Res}_{\text{Max}}$ for any element pair)
06. Set $\text{clusterNumber}$ of all samples $\vec{S}_A$ to 1
07. End

Line 01 loads the user defined maximum relevant resolution of the domain in each dimension and the current minimum cluster number. Next in lines 02-03, the Anarchist chooses a random positive cluster of samples to assess. Line 04 finds the weighted standard deviation of the distribution in each dimension. Line 05 compares each of these values to the maximum relevant resolution to determine if any of those values have grown too large. If any of the one sigma values are greater than the corresponding maximum relevant resolution, Line 06 sets all of the
samples in the cluster equal to 1. This can be seen pictorially in Figure 5.11. In Figure 5.11, the

![Diagram of clusters with Anarchist and Rebel algorithms]

Figure 5.11: The Anarchist Example

left side demonstrates that the magenta cluster is significantly larger than the maximum relevant resolution. On the right side, the Anarchist has marked the cluster as a negative cluster value to indicate that it has violated the relevant resolution criteria.

5.3.3.3 The Rebel

Algorithm 5.9 addresses the third issue mentioned above: the Recruiter can end up assigning multiple clusters to a single cluster as more samples are provided. This is achieved by simply running an instantiation of DBSCAN on one randomly chosen positive cluster. The Rebel (DBSCAN) operates on a randomly chosen positive cluster to determine if the cluster should be broken up into multiple clusters. Note: There is a potential that the Rebel could cause a non-linear scaling relationship if clusters become too large. While this potential for non-linear scaling exists, it is largely mitigated by the Anarchist. While this potential for non-linear scaling exists, since the Rebel is run in parallel to all other algorithms, it does not affect the linear scaling of any other algorithm in the suite if system resources are properly allocated to each algorithm.
Algorithm 5.9. The Rebel (DBSCAN)

01. Load $\text{maxClusterNumber}$, and $\text{searchDistance}$
02. Set $\text{newClusters} = 0$
03. $\text{clusterNumber}_R = \text{FindOneRandom}(\text{clusterNumber} > 1)$
04. $\bar{S}_R = \text{Find}(\text{clusterNumber} = \text{clusterNumber}_R)$
05. For $idxS = 1 \to \text{length}(\bar{S}_R)$
06. If $(\bar{S}_R(idxS))$ is not visited.
07. Mark $\bar{S}_R(idxS)$ as visited.
08. $\bar{S}_{sub} = \text{find}(|\bar{S}_R - \bar{S}_R(idxS)| \leq |\text{searchDistance}| \text{and visitNumber} = 0)$
09. rebelCount = 0
10. If (length($\bar{S}_{sub}$) > 0)
11. If (newClusters = 0)
12. Set $\text{clusterNumber}$ of $(\bar{S}_R(idxS)) = \text{clusterNumber}_R$
13. Else
14. Set $\text{clusterNumber}$ of $(\bar{S}_R(idxS)) = \text{maxClusterNumber} + 1$
15. End
16. While (rebelCount < length($\bar{S}_{sub}$))
17. If ($\bar{S}_{sub}(\text{rebelCount})$ is not visited
18. Mark $\bar{S}_{sub}(\text{rebelCount})$ as visited
19. $\bar{S}_{more} = \text{find}(|\bar{S}_R - \bar{S}_{sub}(\text{rebelCount})| \leq |\text{searchDistance}| \text{and visitNumber} = 0)$
20. Add $\bar{S}_{more}$ to $\bar{S}_{sub}$
21. End
22. rebelCount = rebelCount + 1
23. End
24. newClusters = 1
25. End
26. End
27. End

Line 01 loads the maximum cluster number and current search distance for BattaliaInfini-
tum. Line 02 initializes the flag for whether or not new clusters have been detected as false. Line 03 chooses a random cluster and line 04 finds all of the samples in that cluster. At this point the Generalized DBSCAN algorithm is leveraged in lines 05-27. Lines 05-07 iterate through all non-visited samples and mark each sample as visited, when visited. In line 8, each unvisited sample is checked for neighboring samples. If any samples are found, lines 11-15 assign the sample to the appropriate cluster. Then each of the neighboring samples found are iterated through in lines 16-23 and expanding the clusters found as appropriate. Line 24 indicates that the first cluster is complete by setting the flag to true. As the Rebel continues to iterate from this point, any clusters found, will be broken off into new clusters. The net result of this algorithm
is displayed pictorially in Figure 5.12. This figure shows on the left, a single magenta cluster
which the Rebel will attempt to break apart, given the search distance. Also shown on the left
is the search distance overlaid around a full cluster found to meet the criteria to secede from the
cluster. On the right the end result is four separate clusters.

5.3.3.4 The Massacrist

Algorithm 5.10 addresses the fourth issue mentioned above: the Assassin and Pacifist may
reduce the number of samples making the cluster statistically insignificant creating false positive
clusters due to new samples. The objective of the Massacrist is to find any clusters which have
been stranded, and flag these clusters as statistically insignificant. Since this algorithm is effec-
tive wiping out the clusters from consideration, the algorithm is aptly named the Massacrist.
The algorithm randomly visits clusters and checks to determine the size of a cluster is less than
the square root of the degree of wisdom. If so the cluster number is set to zero.

Algorithm 5.10. The Massacrist

01. Load $D_w$, and minClusterNumber
02. $\text{clusterNumber}_M = \text{FindOneRandom}(\text{clusterNumber} > 1)$
03. $\vec{S}_M = \text{Find}(\text{clusterNumber} = \text{clusterNumber}_M)$
04. If($\text{length}(\vec{S}_M) < \sqrt{D_w}$)
05. Set $\text{clusterNumber}$ of all samples $\vec{S}_M$ to 0
06. End

Line 01 loads the degree of wisdom a minimum cluster number. Line 02-03 finds the list of
samples belonging to a randomly chosen positive cluster. In lines 04-06, if the number of samples
is less than the degree of wisdom, then the cluster samples are all set to zero. The net result of
this algorithm is displayed pictorially in Figure 5.13.
5.3.3.5 The HARM Algorithms Summary

The HARM algorithms are the key to enabling the real-time performance of BattaliaInfinitum. The algorithms ensure that as the sample distribution and decisions of the SRAPH algorithms evolve, old decisions can be re-evaluated. The probabilistic integrity is maintained by the SRAPH algorithms, but the structural distribution integrity is maintained by the HARM algorithms.

5.4 Summary

Chapter 5 presented the technical approach of and rational of the BattaliaInfinitum algorithm. Intelligence is achieved through the balance of competing objectives. High-speed performance is a result of algorithmic simplicity. Independent algorithms were specifically designed for easily spawning multiple workers for parallelization. The SRAPH algorithms operate in sequence together to classify samples as belonging to positive or negative regions, while the HARM algorithms monitor the relationships between positive clusters in parallel to classify contiguous positive detection regions. The next chapter will discuss the computer architecture implementation and results for a performance assessment on the three-dimensional example.
Chapter 6

Three-Dimensional Performance Assessment

As described in the previous chapters, before applying this process to the space domain, it is necessary to evaluate the algorithm performance in the simplest implementation possible. This chapter will discuss the scalable architecture developed to assess whether or not this is possible in the simplest of cases. It will also present the results of these simulations.

6.1 Three-Dimensional Domain for the Performance Assessment

The simple case described in Section 3.2 is the challenge to be described in this chapter. For the initial development testing, the decision was made to begin with the simplest example: produce state estimates for multiple unknown objects moving through a one-dimensional domain with constant velocities.

The objective is take randomly supplied observations, \( \vec{S} = [x, t, p]^T \), where velocity is not measured. This is shown on the left of Figure 6.1. The goal is to estimate the number of objects, the associated trajectories, and the uncertainty in those estimates in real time as shown in Figure 6.2.

6.1.1 Technical Challenges

These objectives appear simple at first glance, however, there are multiple factors adding complexity to this challenge:

1. The algorithm does not know how many objects are in the domain, and thus has no a priori assumptions.
2. No velocity measurement is provided, so each individual observation does not meet the observability criterion.

3. The algorithm is also provided negative detections and must assert where there are no objects.

4. At times the objects cross paths making observations ambiguous as to which object was actually sampled.
5. All of the observations have uncertainties associated with both time and position.

6. The observations are provided via a live stream and provided randomly out of time sequence.

6.1.2 Technical Objectives

Without any information other than the observations themselves, the algorithm must determine the following:

1. Number of objects in the domain,
2. State Estimates for each of the objects,
3. Non-Gaussian probability distributions of uncertainty for each of the state estimates,
4. Non-Gaussian probability distributions for regions where there are no objects,
5. Non-Gaussian probability distributions for regions where it is unknown if there are objects, and
6. Real-time (i.e. millisecond) updates to products 1-5.

6.1.3 Technical Approach

It becomes apparent very quickly, that this is not quite so simple of a task using traditional statistical approaches. Each individual challenge and objective is not necessarily a major challenge, however when all 12 requirements are coupled together across an entire domain, traditional statistical approaches will definitely struggle. Fortunately, there is a different way to look at the problem. Instead of visualizing time as an unique dimension, time of measurement is treated as a dimension no different than position, velocity, or acceleration. An object’s path over time can be thought of as string, line, or curve, through space-time and all of it’s relevant derivatives to the domain. By exercising careful consideration, all of the relevant dimensions and time derivatives it becomes obvious that this is a simple line detection problem. For the two-dimensional problem, a very specific and well-vetted technique, the Hough Transform, can be leveraged to detect static lines in two dimensions (position and time). With this technique, velocity can be estimated based on slope of each detected two-dimensional line. Coupling this approach with computational intelligence and randomized sampling methods can satisfy all 12 requirements. The details to this approach are explained in the previous chapters. The computational architecture is directly related to the accuracy performance of this approach. Deviations from the architecture will result in deviations in the resolution of the resultant products. Probability Density Function (PDF) resolution is where the deviations will manifest. The following section will describe the genesis of the software architecture employed for testing the algorithms.
6.2 Software Architecture

In the early stages of this development, each component of the full architecture was initially developed in MATLAB. However MATLAB did not possess the speed to retrieve, process, store, and visualize results in real-time during development on a single laptop. So to demonstrate the ability to run the approach in real-time, a trade study was performed to identify a more efficient architecture. The RAM limitations of the laptop and the large numbers of samples being generated were a primary concern. This was initially partially mitigated by migrating to a database solution. Choosing the right database was a function of the following requirements:

1. open-source due to budget restrictions
2. high speed read/write operations
3. scalability to cloud design for large datasets, and
4. compatibility with a single MacBook Pro (the laptop used in this study).

After performing a literature search, MongoDB was clearly the best choice [43, 44, 45]. MongoDB is open-source, high speed, and allows a developer the flexibility to modify the data model on the fly without a complete re-design of the database or the code base. While, MongoDB did improve performance in MATLAB, it was still sub-optimal. It required passing every query through a Java driver and then parsing the returned string iteratively.

Another, and perhaps the largest concern, was the time to update the plots within MATLAB for real-time visualization. MATLAB was not able to refresh the scatter plot of 100s of thousands of points in tenths of a second or less. For these reasons it became clear that another language must be utilized. After evaluating reviews and performance comparisons of major open-source languages, C++ was chosen for several reasons [46, 47]. While C seems to be the clear winner for speed, C++ is a close second for many applications, and is the native language of MongoDB [43]. It is also compatible with an open source visualization capability known as OpenGL designed to take advantage of GPU hardware [48]. To simplify the syntax of the OpenGL framework, two open source libraries are also leveraged: GLFW and GLEW [50, 51]. Two additional libraries were needed to complete the architecture. First, an open-source linear algebra C++ library known as Armadillo [49] enables basic linear algebra functions with similar syntax to MATLAB. The second, known as RapidJSON, is also required to quickly parse database returns when the MongoDB _id field is required [52]. Lastly, in order to run the architecture in parallel on a MacBook Pro, OpenMPI, an open source message passing interface is required [53]. All C++ code is edited in X-code. Figure 6.3 depicts the full functional flow.

Eleven algorithms are required in total to produce the results to be presented. Ten of which must run in parallel on a single laptop. In addition one of the ten algorithms, BattaliaInfini-tum, was run with four separate instances to demonstrate the ability to run in parallel.
Figure 6.3: Performance Assessment Software Architecture
In Figure 6.3 the legend references multiple database collections. In a NoSQL database, a database collection is a “bucket” within the database where a collection of key-value pairs is stored. It is a best practice to store key-value pairs to be queried in aggregate in a single collection. Keeping these collections as small as possible improves overall query speeds. There are five separate collections in the single database for this architecture.

### 6.2.1 Propagate Truth

To initialize the simulation, a total of three objects were chosen for the testing.

\[
\begin{align*}
\overrightarrow{X}_0 &= \begin{bmatrix} x_0 \\ \dot{x}_0 \end{bmatrix}, \overrightarrow{X}_1 &= \begin{bmatrix} -1.0 \\ -1.5 \end{bmatrix}, \overrightarrow{X}_2 &= \begin{bmatrix} 0.75 \\ 0.2 \end{bmatrix}, \overrightarrow{X}_3 &= \begin{bmatrix} -0.5 \\ 2 \end{bmatrix}
\end{align*}
\]

(6.1)

and are subject to Equation 3.2.

\[x = x_0 + \dot{x}_0 t\]

These initial state vectors were then propagated through the full domain prior to any other algorithms running. This is depicted in Figure 6.3 as the Propagate Truth Algorithm. This information was then stored to the Domain Truth database collection. Next, the following algorithms were run in parallel:

1. Simulate Observations
2. The **GODS** (The **MUSE**, The **LETHE**)
3. **BattaliaInfinitum** (Run in parallel on 4 processors)
4. Generate State Estimates from Clusters
5. Monitor Solutions and Compare to Truth
6. Simulated Observation Visualization
7. State Space Samples Visualization
8. State Space Clusters Visualization
9. Population and State Estimation Visualization
10. Euclidean Distance Performance Visualization
11. Error Performance Visualization
12. Target Lock Visualization
13. Classification Analytics Visualization
6.2.2 Simulate Observations

Observations are simulated by choosing variable rules or static values for five parameters:

1. measurement uncertainty,
2. measurement probability,
3. measurement location in space-time,
4. total number of observations, and
5. observation frequency.

Rules must be defined for choosing a location in space-time for the next sample. This choice can be defined by any rule set bounded by the defined domain. New observations are generated at the prescribed rate, which can be constant or irregular. For any given measurement, the uncertainty in that measurement is also reported and thus must be chosen by some set of rules. It can be randomized within bounds or set to a static value. The probability associated with a positive or negative detection must also be established by a certain set of rules. It can be set as a static value or by randomizing within specific bounds. Each measurement is then recorded to the measurement space database collection. This process is then repeated at the defined frequency until the total number of observations has been generated.

6.2.3 The GODS

These algorithms are describe in detail in previous chapters. There are only three user-defined parameters for these three algorithms:

1. maximum relevant state estimate resolution, $\overline{Res}_{max}$,
2. minimum relevant state estimate resolution, $\overline{Res}_{min}$, and
3. the level of significance, $\alpha$.

The GODS algorithm monitors to the Cartesian and state spaces database collection and calls to the MUSE or the LETHE algorithms to operate on the system when the criteria described in Chapter 4 have been met.

6.2.3.1 The MUSE

When called by the GODS, the MUSE algorithm queries the measurement space database collection and writes to the Cartesian and state space database collections.

6.2.3.2 The LETHE

When called by the GODS, the LETHE subscribes to the Cartesian and state space collections and writes to the Cartesian and state spaces collection.
6.2.4 BattaliaInfinitum

Four processors running BattaliaInfinitum operate in parallel by subscribing to the Cartesian and state space collections and the history collection. On every iteration, each worker writes to the Cartesian and state spaces collection and the history collection.

6.2.5 Generate State Estimates from Clusters

This algorithm is always running by subscribing to the Cartesian and state space collections and writing to the population / state / uncertainty estimates collection.

6.2.6 Monitor Solutions and Compare to Truth

This algorithm is always running by subscribing to the domain truth and state space collections while writing to the analytics collection. This algorithm is only run when evaluating system performance for a simulated example with known truth. It has a significant impact on the performance of the algorithms, but is necessary to quantitatively evaluate performance. The information it generates is the bases of the Normalized Squared Euclidean Distance Performance Visualization [57], the Error Performance Visualization, the Target Lock Visualization, and the Classification Analytics Visualization.

6.2.7 Simulated Observation Visualization

This is a visualization that displays the current observations overlaid on the truth, as shown in Figure 6.4. This algorithm subscribes to the domain truth collection and the measurement space collection. All visualizations display the information in near-real-time by leveraging the GPU [58] and OpenGL [48] on the MacBook Pro. In Figure 6.4, observations are plotted through this interface in real-time. The observations are colored by the probability that an object has been detected at a given location in the x-position dimension at a particular time. In this example there are only two types of observations, \( p = 0.25 \) (blue) and \( p = 0.75 \) (yellow). Both the time domain and x-dimension domain units have been bounded to -1 to 1 for simplicity.

6.2.8 State Space Samples Visualization

This is a visualization that displays the currently generated samples and the associated probabilities in real-time, as shown in Figure 6.5. This algorithm subscribes to the Cartesian and state space collection. The samples are colored by the probabilities leveraging the same color map as the observations visualization. When compared to the observation visualization it is apparent that the samples have been de-weighted relative to their uncertainty as described in previous chapters. This is why the colors appear less vibrant in the state space visualization.
6.2.9 State Space Cluster Visualization

This is a visualization that displays the currently generated samples and the cluster detection results of BattaliaInfinitum, as shown in Figure 6.6. This algorithm subscribes to the Cartesian and state space collection. This visualization is plotting the same samples as the State Space Samples Visualization, however it is coloring them by assigned cluster number rather than by probability. In this example there are clearly three separate clusters (cyan, purple, and red). They are surround by both gray and black samples. The gray samples are where the algorithms have been unable to determine if they are positive or negative samples. The black samples are samples that have been identified as belonging to a negative region in the state space.

6.2.10 Domain Knowledge and State Estimation Visualization

This is a visualization that displays the current domain knowledge and state estimates with associated uncertainties in the Cartesian domain, as shown in Figure 6.7. This algorithm subscribes to the population / state / uncertainty estimates collection. This visualization crudely maps the classifications back to the Cartesian space to enable better comprehension of the results. For each positive sample it maps a thin transparent green line. When stacked on top of each other this roughly approximates the probability density function in the Cartesian space. Each negative sample is also mapped behind the green probability density functions with a thin transparent blue line. Unfortunately, the GPU quickly saturates the stacking of both colors giving the false impression of higher probabilities and lower probabilities than is accurate. Ideally each cluster would be mapped back into a true probability density function. That said, this approach is
sufficient to communicate the performance of the results in real-time in conjunction with the additional visualizations.
6.2.11 Normalized Squared Euclidean Distance Performance Visualization

This visualization displays the time history of the cluster with the minimum Euclidean distance as compared to each truth state as seen in Figure 6.8. As a result there are three separate plots. The Euclidean distance is colored by the cumulative probability of the associated cluster. The dashed white line represents the $3\sigma$ line. While it is not optimal to approximate the accuracy of a PDF estimate by using a Gaussian assumption, it provides a simple metric to assess how well the algorithms are capturing the uncertainties within the domain. In general if the Euclidean distance is green and below the dashed white line, the uncertainties are well estimated. In addition, Figure 6.8 also displays the number of positive observations for each individual object (light blue) and all negative (orange) observations available over time. This is a quick way to communicate the amount of observations available to the algorithm at the time of the results.

6.2.12 Error Performance Visualization

This algorithm subscribes to the analytics and observations collections.

This visualization is conceptually similar to the Euclidean Distance Performance Visualization. There are only two differences. First, Figure 6.9 plots the errors in the estimates for each cluster. Second, instead of plotting the $3\sigma$ line, it plots the current search distance of the BattalliaInfinitum algorithm (white). In general it is unlikely that the algorithm will perform with less error than this search distance.
6.2.13 Target Lock Visualization

This algorithm subscribes to the analytics collection. Figure 6.10 also plots for each of the three truth objects over time. It is intended to communicate whether or not the algorithm as locked on a given target. For this visualization, the target lock simply means that the 1σ uncertainty for the minimum Euclidean distance cluster and the minimum error are equal and from the same cluster solution. When the two values are not equal they are plotted separately as different colors. The error 1σ is plotted as cyan, while the Euclidean distance is plotted as purple. When they are equal, they are plotted as pink. The value of this plot is to understand not only when a target lock was acquired, but also the uncertainty associated with that target lock.
6.2.14 Classification Analytics Visualization

This algorithm subscribes to the state space and analytics collections. This visualization provides three additional ways to "score" how well the algorithm is performing.

6.2.14.1 False Positive Clusters

The top plot in Figure 6.11 communicates the number of false positive clusters and their approximate significance. False positives are captured by checking each cluster solution to determine if it is within $3\sigma$ of any of the truth solutions. If not, then it is deemed a false positive. The total number of false positives is plotted as a purple point. It is helpful to understand if the false positives are relatively significant. An additional crude "weighted" metric was created and
Figure 6.10: Target Lock Visualization

plotted to approximate this for all false positives.

\[
Weighted = \sum_{cluster_{probability}} \frac{\text{size}(falsePosCluster)}{\min(\text{size}(truePosCluster))}
\]  

(6.2)

With this metric, if the false positives are large and high probability, the weighted number of false positives will be greater than the total. However the inverse will be true if they are small and low probability. The weighted metric is plotted in light green. If there are no false positives, nothing is plotted for that point in time.
6.2.14.2 Normalized Number of False Negative Points (Samples)

The middle plot in Figure 6.11 attempts to communicate the number of falsely assigned, negative samples for each truth cluster. This is displayed for each of the three clusters where cluster 1 is blue, cluster 2 is magenta, and cluster 3 is cyan. It is computed by first identifying the cluster with the closest Euclidean distance. Any negative samples within $1\sigma$ of that solution are counted. They are then normalized by the total number of negative samples in the state space. If none are found for a given cluster, then none are plotted.
6.2.14.3 Normalized Number of Points (Samples) Per Classification

The bottom plot in Figure 6.11 simply shows the breakdown of how many samples have been classified as positive (green), negative (red), or unknown, (yellow). The domain and the provided observations are the main driver for the values on this plot. However, the unknown (yellow) points are a convenient way to identify when the GODS authorized the MUSE to generate more samples.

6.3 Simulated Test Cases Description

The previous sections described the testing environment and the tools available for evaluating the performance of the system. Leveraging these capabilities the system was tested by simulating five different examples. The simulated examples are constructed to increase the difficulty until the method struggles to produce accurate estimates and domain knowledge. Each simulated test case is defined by seven different parameters, explained in more detail in the following subsections.

6.3.1 Defining Simulated Observation Uncertainty

One of the objectives of this research is to demonstrate the ability to handle observations of significantly diverse uncertainties. There are two parameters that are used to vary the diversity in each simulated example. The first parameter defines the maximum observation uncertainty, $Unrct_{max}$. This is a parameter that can range between zero and infinity. If the parameter is defined as $Unrct_{max} = 0.5$, all observations will have an uncertainty of 0.5 or less. The second parameter is a flag to enforce that all observations are of identical uncertainty or of variable uncertainty, $Unrct_{varFlag}$. If this flag is set to uniform, all observations will have an identical uncertainty equivalent to the maximum observation uncertainty. If the flag is set to variable uncertainty, then the observation uncertainties will be randomly distributed between zero and the maximum observation uncertainty, $Unrct_{max}$.

6.3.2 Defining Simulated Observation Probability

Another objective of this research is the ability to handle observations where the probability of object detection may be ambiguous. More specifically the objective is to handle observations that are not binary positive/negative detections. This is achieved in the simulation by associating a probability between zero and one for every observation to indicate the probability of a detected object. Similar to the parameters for defining uncertainty in the simulation, there are two conceptually equivalent parameters for defining the simulated probability. The first parameter defines the maximum range of probability, $P_{Max\Delta}$. This is a value that can range between zero and one. If $P_{Max\Delta} = 0.5$, then the probabilities of each observation will range between 0.25 and 0.75. The second parameter defines whether to generate uniform or variable probabilities,
\( P_{\text{varFlag}} \) If the flag is set uniform, then all positive detections will be generated with probability of \( 0.5 + \frac{P_{\text{Max}}}{2} \) and all negative detections will be generated with probability of \( 0.5 - \frac{P_{\text{Max}}}{2} \). If it is set to variable the probabilities will be randomly distributed between \( 0.5 - \frac{P_{\text{Max}}}{2} \) and \( 0.5 + \frac{P_{\text{Max}}}{2} \). Additionally, another goal of the research is to test the ability to detect objects in the presence of significant noise sources. To achieve this a third parameter is assigned to define the noise in the probability, \( P_{\text{MaxNoise}} \). This is a value that can range between zero and one. If set to zero, \( P_{\text{MaxNoise}} \) has no effect on the solution, however if it is set to any other value between zero and one, it will introduce randomly generated noise into the solutions. This is achieved by Equations 6.3 and 6.4. \( P_{\text{ObsClean}} \) is the noiseless probability generated by the first two parameters \( P_{\text{Max}} \) and \( P_{\text{varFlag}} \).

\[
P_{\text{noise}} = 0.5 + \frac{P_{\text{MaxNoise}}(2 \times \text{rand}(1) - 1)}{2} \quad (6.3)
\]

\[
p = \frac{P_{\text{ObsClean}}P_{\text{noise}}}{P_{\text{ObsClean}}P_{\text{noise}} + (1 - P_{\text{ObsClean}})(1 - P_{\text{noise}})} \quad (6.4)
\]

### 6.3.3 Defining Simulated Observation Minimum Data Delivery Rate

The ability to process observations as they are live streamed to the system is another core objective to be demonstrated. In addition, the goal is to demonstrate to process sparse observations with high latency or dense observations with low latency. Due to these objectives, it is necessary to define a parameter for the minimum data delivery rate. This is achieved by defining a maximum latency between new observations, \( \Delta t_{\text{max}} \). This is a value that can be defined as any number between zero and infinity. This simulator is constructed to randomize the latency between observations between zero and \( t_{\text{max}} \). It is important to note that, the simulator is constructed to provide observations out of time sequence that is another core objective of this thesis.

### 6.3.4 Defining Simulated Observation Data Volume

This research seeks to demonstrate a technique that is robust to sparse and dense data volumes. To demonstrate this it is necessary to define a parameter for the total number of observations to simulate, \( N_{\text{Obs}} \). This parameter can be any integer value between zero and infinity.

### 6.3.5 Defining Example Test Cases

While discussing the various parameters to be adjusted between example test cases, motivations for each parameter were highlighted. In this subsection, the motivation for the five examples tested, will be explored. These examples are documented in Table 6.1. Test case 1 is designed to provide a relatively pristine test case to assess the system performance to the first order. The second test case is designed to stress the system by increasing the data rate. The third test case is designed to further stress the algorithm by providing a very large data volume at an even
higher data rate. The fourth test case is designed to evaluate the performance when data of wide ranging uncertainties is provided. The last test case is designed to evaluate the ability to detect objects when the levels of noise are very high. Each test case will be discussed in detail.

Table 6.1: Definitions for Example Test Cases

<table>
<thead>
<tr>
<th>Test Case</th>
<th>Uncrt\textsubscript{max}</th>
<th>Uncrt\textsubscript{varFlag}</th>
<th>P\textsubscript{Max}</th>
<th>P\textsubscript{MaxNoise}</th>
<th>Δt\textsubscript{max}</th>
<th>N\textsubscript{obs}</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.075</td>
<td>Uniform</td>
<td>0.5</td>
<td>Uniform</td>
<td>0.00</td>
<td>10.0000</td>
</tr>
<tr>
<td>2</td>
<td>0.075</td>
<td>Uniform</td>
<td>0.5</td>
<td>Uniform</td>
<td>0.00</td>
<td>1.00000</td>
</tr>
<tr>
<td>3</td>
<td>0.075</td>
<td>Uniform</td>
<td>0.5</td>
<td>Uniform</td>
<td>0.00</td>
<td>0.0001</td>
</tr>
<tr>
<td>4</td>
<td>1.000</td>
<td>Variable</td>
<td>0.25</td>
<td>Variable</td>
<td>0.00</td>
<td>0.0001</td>
</tr>
<tr>
<td>5</td>
<td>1.000</td>
<td>Variable</td>
<td>0.10</td>
<td>Variable</td>
<td>0.25</td>
<td>0.0001</td>
</tr>
</tbody>
</table>

6.3.5.1 Test Case 1: Slow Data Delivery Rate of Uniform Uncertainty with No Noise Out of Temporal Sequence

The objective of this test case is to establish that the system performs as expected on a very simple technical challenge. The uniform noiseless observations are provided with significant latencies to demonstrate the ability to assess the domain and estimate states for the objects in the domain. For this test case 500 observations were made available for processing. They were populated into the database with a maximum interval between observations of less than 10 minutes. The processing is demonstrated for a total of 2171.17 minutes. During that time interval, the system provides a steady stream of solution updates. It constantly re-evaluates the state solutions, number of objects, and provides an assessment of where there are not any objects. In this use case there is no noise and only one fidelity of data provided with a binary probability assessment of 0.25 when no object is detected and 0.75 when an object is detected.
Figure 6.12 plots a static view of the final results when the simulation was terminated. For viewing the streaming results in movie format visit see Appendix A. It can be seen in the upper left that 500 observations have been made available, and then transformed by the **MUSE** into the Hough Space on the upper right. On the lower right **BattaliaInfinitum** has classified three clusters with significant numbers of samples (purple, cyan, and magenta samples), several additional clusters with low numbers of samples (yellow, red, orange, and green samples), regions where the system is confident there are no object states (black samples), and states where there is insufficient evidence that there are any objects at all (gray samples). On the lower left, these results in the Hough Space have been mapped back to the two dimensional domain. In this plot, it can be concluded that three primary states have been identified which correspond to the three truth objects. Due to the sparse sampling, there are several low confidence trajectories detected. Domain knowledge and uncertainty distributions are estimated as well. In general, it can be concluded that the process is working as anticipated. To analyze how well it is performing the comparison to truth for the three state estimates is discussed.
The most important metric to analyze is the Euclidean Distance of the clusters compared to truth. The reason for this is that the Euclidean Distance accounts for the accuracy of the uncertainty estimation as well as the state estimate. This is shown in Figure 6.13. From this plot there are a few primary conclusions. First, eventually the system successfully finds all three objects, with Euclidean Distances of less than one. Second, as more observations are made available, the more accurate the results become. Third, the system does not detect the third object as quickly as the first two objects. We note that observations did not arrive for the third object until approximately 2 hours into the simulation. This is a likely culprit for the delay in detection. When new observations arrive, they are prioritized over older observations for sampling, however it takes time for those samples to accrue into meaningful sample sizes. Eventually this does occur and all three objects are detected.
Similar conclusions can be drawn from the errors over time shown in Figure 6.14. As can be expected, the system has trouble with the sparse observations available for the first few hours. Due to the database write lock issues, it is unable to generate the volume of samples and BattaliaInfinitum revisits which would enable locking on all three objects sooner.
Figure 6.15: Test Case 1 Target Lock Results

After about 700 minutes of receiving observations and evaluating them in real time, the algorithm is able to lock on to all three objects as shown in Figure 6.15. It is arguable that Objects 1 and 2 locked on within the first 2 hours, however there are definitely instabilities in the solutions. It is particularly interesting that with only two observations on object 1, the system is very close to achieving a lock. This can also be seen in Figure 6.15.
In summary, overall the algorithm performed fairly well given the slow delivery of new observations and large number of possibilities to be considered. In the end it clearly locks on to all three objects with very low false negatives ratios. This can be seen in in Figures 6.12 and 6.16. Additionally, it is observed that there are still several false positives clusters. Given the sparsity of the data, it can be observed that the observations do not negate the possibility that the additional objects exist. Additionally, the weighted number of false positive clusters highlights, that the cumulative number of significant clusters is low. The next example will increase the speed of observation delivery to assess system performance against higher data rates.
6.3.5.2 Test Case 2: Moderate Data Delivery Rate of Uniform Uncertainty with No Noise Out of Temporal Sequence

Shown in Figure 6.17. In this use case only 500 observations are made available. They were populated into the database with a maximum interval between observations of less than one minute. The processing is allowed to run for a total of 440.55 minutes. During that time interval, the system provides a steady stream of solution updates. It constantly re-evaluates the state solutions, number of objects, and provides an assessment of where there are not any objects. In this test case there is no noise and only one fidelity of data provided with a binary probability assessment of 0.25 when no object is detected and 0.75 when an object is detected. In summary, this test case is identical to test case 1 except for the data rate and total run time.

Figure 6.17: Test Case 2 Results
It is important to point out that this test case appears to have outperformed the previous test case despite the shorter run time and high data rate while processing the same total data volume. This seems to indicate that better results will be produced when the system is provided more data quickly. In less than 1.5 hours it has successfully locked on to all three objects. This is shown in Figure 6.18, and then again in Figures 6.19 and 6.20. Figure 6.18 shows the minimum Euclidean Distance for each cluster. In less than 100 minutes, all three objects are well below the $3\sigma$ line.
The errors are also performing as expected as shown in Figure 6.19. They are roughly following the BattaliaInfinitum search distance, which represents a theoretical minimum possible error. Observing that the minimum Euclidean distance and the minimum error plot have both converged, as shown in Figure 6.20, can also evidence this. This is indicated by the change from purple and cyan, into pink.
Figure 6.21 shows that the false positives are in the single digits, and the weighted false positive metric indicates they are of little relative significance compared to the truth solutions. The false negative samples account for less than 1% of the total within two hours of the simulation.
A particularly interesting observation of these results happens in the first few minutes on Cluster 2 (middle cluster). With only 1 or 2 positive observations for each of the objects, the algorithm has successfully found object number 2. This can be seen in the middle plot of Figure 6.18. However, it is important to note that it does not have a strong target lock yet. That does not occur for another 30 minutes. This can be seen in Figure 6.20. In summary, in Figure 6.17 it is clear that the algorithm has found all three objects, however there are two low confidence additional objects. Note this false positive count is less than Test case 1. Since increasing the data rate appears to be beneficial, the next example will attempt to overload the system by streaming in the observations as fast as the computer can write to the database.
6.3.5.3 Test Case 3: High Data Velocity and Volume of Uniform Uncertainty with No Noise Out of Temporal Sequence

This test case is shown in Figure 6.22. In this example only almost 100,000 observations are made available. They were populated into the database with a maximum interval between observations of less than 0.0001 minute. The processing is allowed to run for a total of 457.617 minutes. During that time interval, the system provides a steady stream of solution updates. It constantly re-evaluates the state solutions, number of objects, and provides an assessment of where there are not any objects. In this use case there is no noise and only one fidelity of data provided with a binary probability assessment of 0.25 when no object is detected and 0.75 when an object is detected. As mentioned in the previous section, the purpose of this test case is to attempt to overload the system by making the combinatorial problem nearly impossible on a single laptop.
Figure 6.23: Test Case 3 Euclidean Distance Results

This is potentially the most counter intuitive result from all of the test cases. Typically, throwing this type of data volume and velocity at this type of problem would cause a combinatorial explosion of possibilities effectively crashing the system, as seen in Figure 6.23
However, in this system, it actually performs significantly better than in the previous two use cases, as seen in Figure 6.24.
While the data volume is bogging down the database it still ends up locking on to all three objects sooner. In fact, it locks on within minutes, as seen in Figure 6.25.
By the end of the simulation there are no false positives and less than 1% false negatives, as seen in 6.26. This result leads to the question, “Is more data always better?” More specifically, “Is it always better for this approach?” The next use case will test that theory by accepting data from any fidelity of data.
6.3.5.4 Test Case 4: High Data Velocity and Volume of Highly Variable Uncertainty with No Noise Out of Temporal Sequence

Shown in Figure 6.27. In this use case only 100,000 observations are made available. They were populated into the database with a maximum interval between observations of less than 0.0001 minute. The processing is allowed to run for a total of 482.35 minutes. During that time interval, the system provides a steady stream of solution updates. It constantly re-evaluates the state solutions, number of objects, and provides an assessment of where there are not any objects. In this test case there is no noise and wide variety of fidelities of data provided with a wide variety of probability assessments. The measurement fidelities vary between zero and a half of the domain.

Figure 6.27: Test Case 4 Results
Again, this wildly outperforms the first two use cases. The Euclidean distance rapidly locks on to all three objects, as seen in 6.28.
Similar performance is also evidenced in Figure 6.29. The errors drop very quickly, and begin to follow the curve for the search distance resolution limit. It is important to note that the accuracy is similar to the accuracies in the test case 3.
Figure 6.30 demonstrates that within minutes, all three objects are locked on.
Figure 6.31: Test Case 4 Classification Analytics Results

However, there is an apparent downside to this approach. There are significantly more false positives, as seen in 6.31. Fortunately the false positives have very little relevance. This can be also be seen in figure 6.27. The question now becomes, “What happens when there is noise in the data?” The next test case will vary the signal to noise ratio, to the point where the algorithm can find solutions that are not apparent to a human visually.
6.3.5.5 Test Case 5: High Data Velocity and Volume of Highly Variable Uncertainty with High Noise Out of Temporal Sequence

Shown in Figure 6.32. In this test case only 100,000 observations are made available. They were populated into the database with a maximum interval between observations of 0.0001 minute. The processing is allowed to run for a total of 812.683 minutes. During that time interval, the system provides a steady stream of solution updates. It constantly re-evaluates the state solutions, number of objects, and provides an assessment of where there are not any objects. In this use case there is significant noise and wide variety of fidelities of data provided with a wide variety of probability assessments.

Figure 6.32: Test Case 5 Results
When comparing this test case to the truth states, it also outperforms both test case 1 and test case 2 in the ability to quickly find the three objects as shown in Figure 6.33.
Figure 6.34: Test Case 5 Error Results

Similar performance can also be seen in Figure 6.34. It also noted that occasionally there are more commonly jumps in the level of error.
However as false alarms begin to grow in number, the system struggles to maintain a steady lock on the objects. It does successfully detect all of the objects, however the lock goes in and out. This can be seen in Figure 6.35.
Figure 6.36 shows that the approach begins whittling down the false positives, and only converges on the correct solutions. It is possible that with a faster database and computer this problem could be better mitigated due to more higher density sampling. Another possibility would be to increase the degree of wisdom, however no such simulations have been run in this research, and would fall under future work.
6.4 Results Summary

Overall the approach appears quite promising based on this simple example. Though high noise can affect solutions the algorithm appears to be highly robust to a variety of information types, even when the human mind is not capable of recognizing the solution. It is particularly interesting that it performs better at high data rates than low data rates. This makes a strong case for this type of approach being a possible alternative to traditional statistical filters in domains where an exact solution can be approximated over a finite space-time. This analysis does not assert that this approach is superior or inferior to more traditional statistical filter approaches. No such comparison has been performed. Rather, it only asserts that this type of an approach is possible and has some significant potential advantages.
Chapter 7

Modifications for Space Domain Application

In the previous chapters, the simplest example was used to both explain how this procedure works and to determine if the research shows promise on the simplest of domains. As was found in the previous chapter, the approach appears to merit additional investigation in more complex domains. The original motivation for this research is the space domain. This is a significantly more complex application even without including non-conservative forces. This chapter will detail the mathematics for applying this technique to the space domain by stepping through the processing flow and discussing the modifications required to apply the approach to the space domain.

7.1 Processing Modifications for the Space Domain

In previous chapters, it was explained how an object travelling in one dimension with constant velocity, can be described as a two-dimensional line in space-time, where the slope is defined by the velocity and the intercept is the x position at time equal to zero. The same philosophy can be applied to an object orbiting in an inverse square gravity field described by the following vector,

\[
\vec{X}_{obj} = \begin{bmatrix}
    x_{obj} \\
    y_{obj} \\
    z_{obj} \\
    \dot{x}_{obj} \\
    \dot{y}_{obj} \\
    \dot{z}_{obj}
\end{bmatrix}
\]  

(3.4)
and is subject to the following equations.

\[
\ddot{X}_{obj} = -\frac{\mu X_{obj}}{r_{obj}^3}
\]  \hspace{1cm} (3.5)

\[
r_{obj} = \sqrt{x_{obj}^2 + y_{obj}^2 + z_{obj}^2}
\]  \hspace{1cm} (3.6)

The line formed by an object orbiting in an inverse square gravity field in space-time can be completely described by six parameters. This can be done in any orbital element state space where only one parameter is time dependent, or a fast variable. For this research, Modified Equinoctial Elements is the preferred orbital element state space. The reasons for this will be discussed later in this chapter. The line through space-time of an orbiting object in a simple inverse square gravity field, can be completely described by 6 parameters:

1. semilatus rectum, \( p_{SR} \)
2. \( f \)-component, \( f = e \cdot \cos(\omega + I\Omega) \)
3. \( g \)-component, \( g = e \cdot \sin(\omega + I\Omega) \)
4. second equinoctial element, \( h = \tan(I) \cdot \cos(\Omega) \)
5. third equinoctial element, \( k = \tan(I) \cdot \sin(\Omega) \)
6. true longitude at a specified reference time, \( L(t_0) = \omega + I\Omega + \theta_{TL}(t_0) \)
7. where:
   - \( e \) denotes the eccentricity,
   - \( \omega \) denotes the argument of periapsis,
   - \( I \) denotes the retrograde factor:
     - \( I = +1 \) for posigrade orbits and \( I = -1 \) for retrograde orbits,
   - \( \Omega \) denotes the right ascension of ascending node,
   - \( i \) denotes the inclination, and
   - \( \theta_{TL} \) denotes true anomaly [59].

Similar to how the Hough transform is able to compound the probabilities of a line in three-dimensional space-time, \([x, v, t]^T\) down to a single two-dimensional state, \([\rho, \zeta]^T\), the same philosophy can be applied to the trajectory of an object orbiting in a simple inverse square gravity
field. The line of probabilities formed in \([x_{\text{obj}}, y_{\text{obj}}, z_{\text{obj}}, \dot{x}_{\text{obj}}, \dot{y}_{\text{obj}}, \dot{z}_{\text{obj}}, t_{\text{obj}}]^T\), can be compounded down to a single six-dimensional state, \([p_{\text{SNR}}, f, g, h, k, L(t_0)]^T\). There are several reasons for this transformation that are common to the two-dimensional Hough space motivations:

1. discrete description of intersecting lines,
2. low signal-to-noise-ratio (SNR) line detection, and
3. enables the ability to leverage negative and probabilistic observations.

It is possible to assess probability samples in the Cartesian space, thus eliminating the transformation to the modified Equinoctial element space. In fact, this is how some existing techniques perform data association. The problem with this approach is that it inherently leads to cross tagging. If two lines through space-time intersect, a cluster algorithm is likely to consider the trajectories as one possibility sample cluster, thus calling two objects one object. Figure 7.1 pictorially describes this comparison. In Figure 7.1(a) positive and negative samples of two intersecting lines are provided. If cluster detection is performed in this domain space, the results in Figure 7.1(b) are determined. There is only one positive cluster found, when in reality there should be two clusters. If cluster detection is instead performed in the state space, two distinct clusters are found corresponding to the two lines, with some samples belonging to both cluster. This is shown in Figure 7.1(c).

The second distinct advantage of the performing the cluster detection in the state space relates to Signal to Noise Ratio (SNR). SNR is the primary driver for being able to detect objects in any sensing modality. In Chapter 4 this was seen in Figures 4.5, 4.6, and 4.7. As SNR decreases, it becomes increasingly difficult to detect objects traversing a domain. However, by incorporating knowledge of how the objects must behave based on known physics, the SNR can be increased by transforming the measurements into a different state space. In the case of Figures 4.5, 4.6,
and 4.7 this is demonstrated by transforming into a Hough space to find two-dimensional states. As result, three states become obvious in Figure 4.7(b), whereas they are undetectable in Figure 4.7(a). The same process can be applied to the space domain to find objects that are undetectable in the measurement space. The key to achieving this objective is diverging from binary assessments of object detection. If measurements are instead provided as probabilistic assertions rather than as binary positive or negative assertions, those probabilities can be compounded from multiple sensors in the state space. This enables the computer vision technique to increase the SNR. The end result is the ability to detect much fainter objects than possible with a single sensor, by probabilistically fusing sensor measurements from multiple sensors in the state space prior to any binary assertions of existence.

This is a radically different approach to space domain information fusion. The approach is essentially turning the entire space domain into a six dimensional image, and performing simple computer vision techniques to fuse measurements from any modality.

This is the only major modification to the overall processing described for the simple case in previous chapters. The algorithms must be modified to generate samples from seven dimensions into six dimensions, instead of generating samples from three dimensions into two dimensions. This modification primarily affects the propagator, the simulator, the MUSE and BattaliaInfinitum. However, BattaliaInfinitum modifications are simple to implement.

### 7.1.1 Propagate Truth Modifications

The modifications required to migrate to the space domain are apparent. The current example is a three dimensional problem $X = [x, v, t]^T$, where as space is a seven dimensional problem (assuming no non-conservative forces), $X = [x_{obj}, y_{obj}, z_{obj}, \dot{x}_{obj}, \dot{y}_{obj}, \dot{z}_{obj}, t_{obj}]^T$. A propagator in the space domain must be used to forecast truth throughout the domain and to generate observations. This is a fairly straightforward task and a very well understood problem.

### 7.1.2 Simulate Observations Modifications

Simulating observations in the space domain is also a fairly well understood problem. However, there is a slight adjustment to typical simulations. Rather than only simulating observations on a single object, the simulator needs to simulate observations of the domain. For example, if a measurement is provided in a specific region, it should not only provide the measurements of objects but also probabilities where there were no objects. In fact, to fully challenge the algorithm, no binary assertion of object presence should be provided at all, simply a probability of existence.
7.1.3 The MUSE Modifications

When generating samples, the MUSE will need to use the proper conversions from the observations to the Modified Equinoctial Elements (MEE) state space. Observations will typically provide seven parameters, some of which may be unmeasured. However because this process is reliant on defining a domain boundary in all dimensions, any unmeasured quantities can be bounded. The MUSE will simply draw random samples within the bounds of the observations, to create a sample which is a single point in all seven dimensions. For example, if only one right ascension angle and one declination angle are provided with the site location and time of measurement, there are still multiple unknown terms in the measurement domain:

\[
\bar{S} = [r_{obs}, \dot{r}_{obs}, \theta_{obs}, \dot{\theta}_{obs}, \phi_{obs}, \dot{\phi}_{obs}, t_{obs}, p_{obs}]^T
\]  

(3.7)

The convenient reality of this approach is that the domain is bounded and thus these variables in bold, \([r, \dot{r}, \dot{\theta}, \dot{\phi}]\), can be bounded within the domain. It is also possible to assign a PDF to each parameter if a strict bound to the parameter is not acceptable. These bounds or PDF can be used to generate the necessary samples in the space domain. Once that is complete, the \(\bar{S}_{samp}\) can be converted to the \(\bar{X}_{samp}\). There is a simple process required to convert the sample into the state space where BattaliaInifinitum can analyze the data.

\[
\bar{X}_{samp} = [x_{Samp}, y_{Samp}, z_{Samp}, \dot{x}_{Samp}, \dot{y}_{Samp}, \dot{z}_{Samp}, t_{Samp}]^T
\]  

(7.1)

That sample can then be transformed into the six-dimensional space representation easily as shown in Algorithm 7.1. This sample is then placed into the database with the original probability for BattaliaInifinitum to process the sample along with all of the other samples. It might seem impossible to ever draw enough samples to sufficiently sample an observation like this in this way, but the results in the simple example showed that bringing in more diverse data and sampling as much of it as possible is more important than fully sampling a small number of observations for this technique.
Algorithm 7.1. The MUSE for Space

01. $\mathbf{X}_{\text{samp}} = [x_{\text{samp}}, y_{\text{samp}}, z_{\text{samp}}, \dot{x}_{\text{samp}}, \dot{y}_{\text{samp}}, \dot{z}_{\text{samp}}, t_{\text{samp}}]^T$
02. $\mathbf{COE}_{\text{samp}} = \mathbf{SV2COE}(\mathbf{X}_{\text{samp}})$
03. $n_{\text{samp}} = \sqrt{\frac{\mu}{a_{\text{samp}}}}$
04. if $e_{\text{samp}} < 1$
05. $E_{\text{samp}} = 2 \times \arctan\left( \sqrt{\frac{1-e_{\text{samp}}}{1+e_{\text{samp}}} \times \tan\left(\frac{T_{A_{\text{samp}}}}{2}\right)} \right)$
06. $M_{\text{samp}} = E_{\text{samp}} - e_{\text{samp}} \times \sin(E_{\text{samp}})$
07. elseif $e_{\text{samp}} > 1$
08. $F_{\text{samp}} = 2 \times \arctan\left( \sqrt{\frac{e_{\text{samp}}-1}{e_{\text{samp}}+1} \times \tan\left(\frac{T_{A_{\text{samp}}}}{2}\right)} \right)$
09. $M_{\text{samp}} = e_{\text{samp}} \times \sinh(F_{\text{samp}}) - F_{\text{samp}}$
10. elseif $e_{\text{samp}} == 1$ or $e_{\text{samp}} == 0$
11. $M_{\text{samp}} = T_{A_{\text{samp}}}$
12. end
13. $M_{\text{samp},t=0} = M_{\text{samp}} - n_{\text{samp}} \times t_{\text{samp}}$
14. $T_{A_{\text{samp}},t=0} = \text{MeanToTrueAnomaly}(M_{\text{samp},t=0})$
15. $\mathbf{COE}_{\text{samp},t=0} = [a_{\text{samp}}, e_{\text{samp}}, \text{incl}_{\text{samp}}, \omega_{\text{samp}}, RAAN_{\text{samp}}, T_{A_{\text{samp}},t=0}]^T$
16. $\mathbf{MEE}_{\text{samp},t=0} = \mathbf{COE2MEE}(\mathbf{COE}_{\text{samp},t=0})$

7.1.4 BattaliaInfinitum Modifications

The BattaliaInfinitum algorithms require no modifications from what was presented in Chapter 5. In the implementation of the algorithm there will simply be an additional four dimensions in the queries and distance calculations. The algorithms are specifically designed to be applicable to any cluster detection challenge with at least one dimension. That said, it would be best to change the computational architecture to prevent the significant delays caused by the database write-lock. This occurs when multiple parallel processes attempt to write to the database at the same time. When write-lock occurs, the database locks the database and introduces a deliberate latency to database write functions. The more observations and samples that are stored in the database, the longer it takes to write updates. This is contrary to the algorithms that perform computations at the same rate independent of the number of observations or samples stored. As a result, the probability of write-lock increases as more samples are generated. This is not a challenge unique to this approach and the primary impact is to the number of samples that can be generated and evaluated. This latency on the generation of new samples equates to a state estimate quality impact. Ideally, there would be hundreds or thousands of the BattaliaInfinitum workers processing the samples. This would enable the GODS to authorize the MUSE to generate significantly more samples, drastically improving the quality and usefulness of the results. The best computer architecture is beyond the scope of this thesis.
7.1.5 Generate State Estimates from Clusters Modifications

Similarly, there are no major challenges to computing the state estimates based on the results of BattaliaInfinitum. There are simply more dimensions to include when computing the weighted average of all samples in a cluster.

7.1.6 Visualization Modifications

The visualizations represent a significant challenge. In the simple examples from previous chapters, only two of the three dimensions are displayed, \([x, t]^T\). Visualizing two dimensions on a two-dimensional screen is trivial. However, in the space example there are now seven dimensions to consider. Most likely a two and a half dimensional approach will be taken to plot a three dimensional globe on a two dimensional screen for the position dimensions, \([x_{obj}, y_{obj}, z_{obj}]^T\), over a finite time window. Two and a half dimensional visualizations refer to the plotting of three dimensions on to two dimensional display, such as a typical monitor, projector, or TV screen. The visualization will likely be configured to iterate through a the time interval of the domain, similar to how weather radar maps iterate through a sequence. A similar approach is a possibility when applying this to the space problem.

The current examples from Chapter 6 use thin transparent lines to roughly capture and display the PDF, however it would be beneficial to rather estimate the PDF from the samples, map it back to the Cartesian domain, and then display the PDF rather than render every single sample. The reason for this is that the rendering becomes saturated such that most of the domain appears to be probability of zero or one. In reality there is significantly more structure to the PDF than is apparent in Figures such as Figure 6.7.

Visualizing a 7 dimensional PDF in a meaningful way will be a significant challenge in the space domain. As mentioned previously, one approach would be to display the position vectors and iterate over a finite time window. The bottom line is that this is an unsolved challenge, which will represent the biggest hurdle for communicating the value of this technique to the space problem. Visualizing space domain uncertainties in a real-time meaningful way is a difficult challenge with currently available technologies.

7.2 Space Domain Proof of Concept Example

While evaluating the entire system on a space domain is beyond the scope of this thesis, it is necessary to demonstrate the viability of computer vision for detecting space objects. Specifically demonstrating that the transform documented as a upgrade for the MUSE is able to find objects leveraging both positive and negative objects. Similar to the previous examples on a simple use case, three objects will be modeled as they traverse the space domain. Since the transform is valid on circular, elliptical, and hyperbolic orbit classes, one of each was chosen for this example.
The truth states are defined by the Classical Orbital Elements.

\[
\overrightarrow{COE} = \begin{bmatrix} a \\ e \\ i \\ \omega \\ \Omega \\ \theta_{TL} \end{bmatrix}
\]  
(7.2)

The states used for this demonstration are:

\[
\overrightarrow{COE}_1 = \begin{bmatrix} 1.75 \\ 0 \\ \frac{3\pi}{4} \\ \frac{3\pi}{4} \\ \frac{\pi}{10} \end{bmatrix}, \quad \overrightarrow{COE}_2 = \begin{bmatrix} -1.9 \\ 1.95 \\ \frac{\pi}{10} \end{bmatrix}, \quad \overrightarrow{COE}_3 = \begin{bmatrix} 1.5 \\ 0.85 \\ \frac{3\pi}{10} \end{bmatrix}
\]  
(7.3)

These states are converted to a Cartesian state:

\[
\overrightarrow{X}_{\text{obj}} = \begin{bmatrix} x_{\text{obj}} \\ y_{\text{obj}} \\ z_{\text{obj}} \\ \dot{x}_{\text{obj}} \\ \dot{y}_{\text{obj}} \\ \dot{z}_{\text{obj}} \end{bmatrix}
\]

that is subject to the following equations.

\[
\ddot{\overrightarrow{X}}_{\text{obj}} = -\mu \overrightarrow{X}_{\text{obj}} \frac{1}{r_{\text{obj}}^3}
\]  
(3.4)

\[
r_{\text{obj}} = \sqrt{x_{\text{obj}}^2 + y_{\text{obj}}^2 + z_{\text{obj}}^2}
\]  
(3.6)

where \(\mu = 2\) with no dimensions specified. This is done to demonstrate that the technique is not Earth specific. Taking all of this into account the three states when propagated over a finite time window, result in the tracks plotted in Figure 7.2 in Cartesian position space. Since this is all
hypothetical, no units have been specified for the simulation other than all angle measurements are in radians. Figure 7.2 illustrates how the trajectory of an object in an orbit is just a line in six dimensions, which has a 3-dimensional shadow over a finite window that maps to position space a simple curve. The goal of the technique is to find the line formed by measurements with uncertainties in the six-dimensional MEE space.

To demonstrate this concept, samples in the form of Equation 7.1 are generated as if state estimates with uncertainties have been provided. Simulating this type of information source vs. RADAR or optical measurements was chosen since it is the simplest example. If this example does not generate positive results, the technique has very little chance of success on other modalities. Also, this method is sufficient for demonstrating the validity of the transform. It is beyond the scope of this thesis to demonstrate the efficacy of the end to end system on the space example. The samples generated for this example are shown in Figure 7.3.

Since demonstrating BattaliaInfinitum on the space domain is beyond the scope of this thesis, another approach must be used to find the maxima in the distributions. This is done by following the same approach as a traditional Hough transform. Each sample will be transformed and then binned into a six-dimensional histogram in the MEE space. The two-dimensional equivalent would be a two-dimensional image or matrix. The positive samples will increase the value of the associated bin and the negative samples will decrease the value of the associated bin. Once this is done a simple thresholding algorithm can be applied to find the six-dimensional pixels that correspond to the most likely locations where there might be space objects. The benefit of this form of validation is that it is relatively simple to implement. The downside is that it introduces additional error and fundamentally limits the accuracy of the analysis. It also does not assert how many objects are in the domain, however this can be visually determined if the clusters are sufficiently distinct when mapped back into the three-dimensional Cartesian position.
For this transformation a histogram binning of 25 bins per each of the six-dimensions is lever-aged. This results in a total of $25 \times 25 \times 25 \times 25 = 9,765,625$ bins. Significant noise was applied to the samples from each state. Gaussian noise was applied with a one $\sigma$ of 0.05 to both position and velocity in their respective units. This represented $3-5\%$ of the domain with some samples as high as $15\%$ of the domain. Both negative and positive samples were injected into the system while the transformation updated with each iteration. The results are shown in Figure 7.4(a)-(d). A video version can also be found in Appendix A.

Figure 7.4 shows multiple two and a half dimensional visualizations of the same data just from different views. Looking at these plots it is clear that the simple six-dimensional histogram approach has indeed found three separate regions, which, in theory, a cluster algorithm could identify as three distinct objects. It can be observed that quantizing the solution space has resulted in artificial sub clusters within each cluster when in this view. These reasons are the motivation for eventually wrapping this transform into the full system.

While this method of looking at the results provides insight, it only displays a three dimensional shadow of a six dimensional result. Another way to visualize these results is with a spider plot. Spider plots can be used to display information for an unlimited number of dimensions when there is a common state across all of the dimensions. A spider plot enables plotting a multiple solutions to n-dimensional problems. In this case it is helpful to plot the results of the estimated Modified Equinoctial Elements. A single truth solution is plotted for the first truth object in Figure 7.5(a). Each solution is plotted on a single radial axis. Data is only ever plotted on those axis, however it is common on a spider plot to connect solutions on each neighboring axis with
a line. This enables quickly making associations between different solutions. For example, all three truth states are plotted in Figure 7.5(b). Each solution has a distinct shape which enables a quick means of making a more insightful conclusion versus the three dimensional shadow. Next, the solutions from the full run of the transform are plotted in Figure 7.6.

At first glance this is a very difficult chart to interpret if an unfamiliar plot, however there are several key insights which this provides. First, it is clear that there are three obvious solutions found. This is evidenced by the three distinct shapes formed by the lines stretching between neighboring dimensions. This means the transform successfully determined there to be three objects in the space domain. Next, there are dense clusters of white circles centered on each truth solution in each dimension, and these clusters all form associations with neighboring dimensions on the same object. This means that the orbit states of each object were successfully localized for all parameters. In addition, object three has significantly more uncertainty in true longitude as compared to all other dimensions and all other objects. This is likely a function of the orbit range during measurements and how the simulated noise was introduced into the original samples. Since it was applied in the Cartesian space it will not map uniformly to all other dimensions since the objects are of different orbit classes and only being sample during a partial revolution.
While this plot provides significant insight into potential solutions it is paramount to consider that this is still a shadow of the true complexity buried in the six dimensional volume. In this case, the spider plot does show all six dimensions, but oversimplifies the interactions of each dimension versus the other dimensions. It not only reduces the interactions down to a single linear relationship, but also reduces from one dimension versus six dimensions down to one dimension versus two dimensions. For example, no relationship is displayed between $f$ and $h$. This is a tool for identifying the uncertainty in each parameter individually. A six-dimensional probabilistic region as a function of time is required to most accurately estimate the uncertainties as a function of time.

7.3 Summary

Applying this system to the space domain is a fairly straightforward task from an algorithmic standpoint. There are no major hurdles, however this is not the case for designing the real-time visualizations and computational architecture. There are significant design trades that need to be considered prior to implementing the solution.
Figure 7.6: Spider Plot Results overlayed on Truth States
Conclusions

In Chapter 3 multiple technical challenges were outlined in Table 3.1. Each of these technical challenges required a new approach to providing real-time domain assessments based on sparse data. This chapter will discuss conclusions based on the framework outlined in Chapter 3, and then discuss potential future work.

8.1 Top Level Objectives

The overarching goal of this thesis was to generate real-time adaptive population distribution and state estimates for the regions of interest in the domain while scaling independently of the number of objects in the domain. For this thesis, real-time is defined as providing updates on a millisecond time-scale independent of the number of measurements, objects, or software architecture. Adaptive implies the ability to react to a changing environment. This could be a change in the number of objects, a maneuver, or new observations provided out of time sequence. Standard approaches leverage positive detections to make state estimates with uncertainties, but rarely include overall population assessments, and certainly not in real time. Population assessments are defined as probability distributions throughout the entire domain of whether or not there are objects in a given state or location. State estimates require the classification and discretization of these probabilistic regions. Lastly a domain is defined in this thesis as a finite region of interest that can be bounded. It can be defined in the measurement space, physical space, or a state space, but it must be finite.

8.1.1 Technical Approach

The unique solution identified in this research is the application of a new computer vision technique combined with a new computational intelligence algorithm to estimate the domain populations, to perform data association, initial orbit determinations, and state estimations across the entire domain in real-time. The computer vision technique developed leverages the con-
cept of a Hough transform, originally developed for line detection, but applied to a variety of other machine vision challenges. This is performed in the Modified Equinoctial Element space. It is used to create a probabilistic distribution of probability samples based on the measurements collected. Turning that information into data associations, initial orbit determinations, and state estimations required coupling the technique with a real-time probabilistic computational intelligence cluster algorithm. Furthermore, the real-time requirements drove the need for an additional computational intelligence algorithm to intelligently monitor system performance knowing nothing about the system itself. Then using that information to intelligently scale the samples in the modified equinoctial element Hough space to ensure the real-time performance. This computational intelligence algorithm also manages the samples to ensure the system learns when a change in the domain has occurred.

8.1.2 Scalability

As stated in the last section, a core objective is to develop a system that scales independently from the number of objects in the domain. Using traditional approaches, this is mathematically impossible without scaling processing requirements. Statistical filters will encounter a combinatorial challenge that grows with both the number of objects and the number of observations. Instead, by migrating to a computer vision approach to assess the entire domain, the mathematical complexity of the problem is unaffected by the number of objects in the domain. Whether there is only one object or forty-two googol number of objects, the processing time is identical. However, nothing is free. The ability to make this possible requires being flexible on the resolution of the estimates. With forty-two googols of objects, the resolution must increase to be able to identify discrete states. This does not affect the computer vision and population assessment, but impacts the ability to distinguish between discrete objects. With insufficient resolution in the computational intelligence algorithm, it is possible to produce a single state estimate with a probability distribution that encompasses multiple objects in similar orbits. A relevant example would be a debris cloud created by a satellite collision or break-up event.

Additionally, scalability to more processors for a linear increase in performance is also desired. This is achieved by ensuring both computational intelligence algorithms and the computer vision transform can be distributed on unlimited number of processors without requiring cross communication. As demonstrated in this research, this is complicated by the data design. A specific challenge is the issue of write-lock in MongoDB when multiple parallel algorithms attempt to write to the database at the same time. While all algorithms can be distributed to unlimited processors, a different solution is necessary for the database to fully realize this objective. Due to the surge of “big data” challenges over the last decade, there has been an explosive amount of advancements in database technologies that may solve this issue.
8.1.3 Data Association

In certain conditions, such as closely space objects or large state estimate uncertainties, it can become ambiguous as to which object an observation should be associated. In some cases, multiple objects can actually be associated with a single observation when occulting within the measurement uncertainties of a sensor. Due to these reasons it is necessary to consider all possibilities of association. Leveraging the computer vision approach enables never making a binary decision on association. Multiple objects can share the same measurement. Based on the results of the computational intelligence algorithm, a probability of association for any measurement to any object can be produced.

8.1.4 Fusion

Converting all observations from all sources into a common fusion engine enables maximizing the extraction of information content. This objective meant that the approach needed to map all measurements into a common state space prior to making any binary decisions. This is achieved by the computer vision transform of all measurement samples into the Hough space.

8.1.5 Multiple Modalities and Measurement Types

This objective drove two requirements. First, the ability to leverage data from any modality in a single fusion algorithm was required. This is achieved by sampling all measurements into a common state space. This is possible since the domain is mathematically bounded to achieve observability. It is performed by the computer vision and computational intelligence algorithm for generating samples from measurements. The second requirement is to ensure the ability to leverage positive and negative observations. This is a key pillar of this technique. The goal is to assess the domain in its entirety. This task is impossible by only asserting where objects are, and not where they are not. In fact, the majority of the information measured by sensors in the space domain is negative space. Not leveraging this information means not leveraging the vast majority of information measured. This is true for almost all modalities. This information not only enables an assessment of where it is improbable for there to be an object, but the information also enables the reduction of uncertainty for where there is an object. This is possible due to the computer vision transform.

8.1.6 Adaptability

In most domains, things change. If they do not, then there is little motivation to observe them to update states. To ensure relevance it is critical that any approach automatically adapt to changes in the environment. Leveraging statistical filters leads to challenges such as non-convergence and local minima that can require reprocessing previous measurements as new measurements arrive. This can be inefficient and can lead to incorrect results. The computational intelligence algorithm developed for this research is specifically designed to not be impacted by such challenges. It is
designed to perpetually “second-guess” all decisions. In effect, it is multiple algorithms “having an argument” or “battling” on a micro level to ensure that the macro level results and decisions are valid. If something changes in the domain the “tide of the battle” shifts to correct decisions that are no longer valid.

8.1.7 Initial Assumptions

When evaluating any domain it is paramount to understand not only what is known, but also what isn’t, and how well knowns are understood. Making assumptions that are over generalizations can lead to poor decision-making and misinformation. The net effect is higher cost and higher risk. Common statistical filters make assumptions about the uncertainties in a domain, such as Gaussian distribution assumptions. A core objective of this research was to develop a technique that does not assume anything about the distribution of uncertainties. This objective drove the design for the computational intelligence cluster algorithm to be agnostic to the cluster distribution.

8.1.8 Performance Summary

In general all initial objectives were demonstrated in the results from Chapters 6 and 7. However, there are still downsides to this approach. While the approach demonstrates potential significant improvements over common practice, it is not the complete solution. The primary objective was real-time with a trade between accuracy and cost. There are likely times when knowledge of a domain requires high-accuracy. Because of this, a joint approach likely makes the most sense. In most domains, such as the space domain, real-time awareness does not require high fidelity. A simple corollary example is the design of the human eye. While the field of view of the eye is relative large, the accuracy is not uniform across the entire field of view. In fact, it is seven times less accurate around the periphery versus the center of field of view. The full eye is constantly monitoring the full domain at low resolution in real-time watching for something interesting. When something interesting happens, the eye moves to track the event while the brain characterizes at higher-accuracies. In this example, this research is the real-time analysis of the full domain, while the high fidelity statistical filters are the in depth analysis of the center of the eye. This ensures efficient usage of resources for ensuring maximum levels of knowledge.

8.2 Future Work

8.2.1 Next Steps For the Space Domain

The next step is to develop and demonstrate the approach or real space domain data from multiple modalities. If truth is also provided, stress testing the process is necessary to understand the robustness of the approach, when it works, when it fails, and why. Finally, assuming that process does not raise any insurmountable technological challenges, the approach should be compared
on real data to other more traditional techniques. Before completing this process, it is only possible to know that the approach is possible or feasible. The entire testing must be completed to determine if the process has compelling advantages over other techniques. Once demonstrated on real-data, a re-design of the database solution to mitigate the issues with write-lock impacting the accuracy of the system. This is critical for ensuring scalability on large cloud infrastructures. Finally, identifying criteria for when higher fidelity algorithms should be cued based on the real-time analysis is paramount to demonstrate the ability to achieve high-accuracy when necessary.

8.3 Applicability to Other Domains

While this approach was motivated by the space domain awareness challenges, it is agnostic to the point where it could be applied to many different domains. The most obvious application is to any object travelling through any domain. In fact, two such examples are discussed in this thesis, however it is not limited to these examples. If an exact equation is available for roughly approximating motion over a relevant time space, the technique can be applied in a very similar construct.

The real-time computational intelligence algorithm can be applied generically to population distribution assessments. Understanding the population of objects is important in many domains. Whether tracking satellites or the migration patterns of schools of fish, measurements are made. In many cases there are significant uncertainties that need to be resolved by combining multiple measurements to assess the overall population without necessary tracking a single fish or single school of fish. The computational intelligence algorithm can be applied to any domain for identifying clusters of populations.

As robotic systems are presented with more sophisticated tasks in dynamic environments, making quick course classifications of objects in images becomes more important. This approach may provide value for building “instinctual” reaction times for mobile robotic systems. For example, if a baseball is about to hit someone in the head, the human mind does not need to know that it’s a baseball or it’s exact trajectory to know they should duck. This approach provides a means to potentially provide machines with the ability to react in real time.

A less obvious application of this technology is in the analysis of social networking. Assume a researcher has multiple metrics for capturing the degree of separation between patterns of behavior on social media for the goals of identifying undesirable or desirable behavior patterns. The objective is then to find clusters of common behavior and evolution of each cluster over time. For example, identifying groups of people posting or clicking on information related to terrorist organizations. The cluster algorithm developed in this research may provide significant value in real time on live data streams of this type.
From a mathematical perspective financial market analysis is similar in many ways to the social networking example. In this case the objective would be to understand and monitor groups of transactions to estimate trajectories and predict how they might react in real-time to the evolving market to prevent things such as market crashes or potentially other high-risk events. Also, to identify organizations that invest in specific industries.

The last example is real-time cyber traffic analysis. Again, a researcher would need to define a set of relevant metrics and then use the computational intelligence algorithm to determine groups of actors, and estimate the pattern trajectories of those groups to predict future actions.

8.4 Conclusions Summary

The approach developed appears to demonstrate significant promise to analyzing the space domain in real-time. However, it is not the complete solution. It should be combined with higher fidelity computationally expensive approaches as needed. While it was developed primarily for the space domain there are unlimited possibilities for how the approach can be applied to other domains. The most domain agnostic components of this research are the computational intelligence algorithms, however when an equivalent computer vision concept is possible for a domain, the coupling of that approach with the computational intelligence algorithms may provide significant additional value to information fusion.
Appendix: Videos of Live Results

See Test Case 1 at address:
https://www.youtube.com/watch?v=3WAnZ_zgsxA
See Test Case 2 at address:
https://www.youtube.com/watch?v=h5ObW59sTc
See Test Case 3 at address:
https://www.youtube.com/watch?v=9OFLS29IS0o
See Test Case 4 at address:
https://www.youtube.com/watch?v=Px7LUh_N7Uo
See Test Case 5 at address:
https://www.youtube.com/watch?v=-cr4DQqSaZk
See High Noise MEE Transform Results at address:
https://www.youtube.com/watch?v=6jqtR3sJDdA
References


[12] European Space Agency “Space Debris: Frequently Asked Questions” http://m.esa.int/Our_Activities/Operations/Space_Debris/FAQ_Frequently_asked_questions February 11, 2018
February 11, 2018


https://ntrs.nasa.gov/archive/nasa/casi.ntrs.nasa.gov/20150011452.pdf May 12, 2015

[18] National Aeronautics and Space Administration “Role of NAS and TPESP(1955-1956)”
https://history.nasa.gov/SP-4202/chapter6.html February 11, 2018

[19] United States Space Command “Space Surveillance”
http://www.au.af.mil/au/awc/awcgate/usspc-fs/space.htm February 11, 2018


https://www.globalsecurity.org/space/world/russia/space-surveillance.htm February 11, 2018

[22] Russian Academy of Sciences “International Scientific Optical Network (ISON) activities on highly elliptical orbit (HEO), geosynchronous orbit (GEO) and Near-Earth objects (NEO) observation and analysis in 2013”. 51st Session of STSC COPUOS, Vienna, February 10-21, 2014

https://www.globalsecurity.org/space/world/china/space-surveillance.htm February 11, 2018

[24] European Space Agency “Space Surveillance and Tracking - SST Segment”
http://m.esa.int/Our_Activities/Operations/Space_Situational_Awareness/Space_Surveillance_and_Tracking_-_SST_Segment February 11, 2018

https://exoanalytic.com/portfolio-items/espoc/ February 11, 2018

https://www.agi.com/comspoc February 11, 2018

[28] Bolden, Spencer, Pennsylvania State University “Serendipitous Acquisition of Space Situational Awareness From Astronomical Surveys (SASSAFrAS)”
Advanced Maui Optical and Space Surveillance Technologies Conference 2014

[29] Blake, Sanchez, Krassner, Sundbeck “Space Domain Awareness”
https://amostech.com/TechnicalPapers/2012/Data_Services/BLAKE.pdf
Advanced Maui Optical and Space Surveillance Technologies Conference 2012


[55] Curtis (1829), ”battalia”, The London encyclopaedia: or Universal dictionary of science, art, literature, and practical mechanics, comprising a popular view of the present state of knowledge, 3, Thomas Tegg, p. 666


