DISSERTATION

# BIG DATA DECISION SUPPORT SYSTEM

Submitted by

Tian J. Ma

Department of Systems Engineering

In partial fulfillment of the requirements

For the Degree of Doctor of Philosophy

Colorado State University

Fort Collins, Colorado

Fall 2022

Doctoral Committee:

Advisor: Edwin Chong

Steve Simske
Daniel Herber
Ali Pezeshki

ABSTRACT

BIG DATA DECISION SUPPORT SYSTEM

Each day, the amount of data produced by sensors, social and digital media, and Internet of Things is rapidly increasing. The volume of digital data is expected to be doubled within the next three years. At some point, it might not be financially feasible to store all the data that is received. Hence, if data is not analyzed as it is received, the information collected could be lost forever. Actionable Intelligence is the next level of Big Data analysis where data is being used for decision making. This thesis document describes my scientific contribution to Big Data Actionable Intelligence generations. Chapter 1 consists of my colleagues and I's contribution in Big Data Actionable Intelligence Architecture. The proven architecture has demonstrated to support real-time actionable intelligence generation using disparate data sources (e.g., social media, satellite, newsfeeds). This work has been published in the Journal of Big Data. Chapter 2 shows my original method to perform real-time detection of moving targets using Remote Sensing Big Data. This work has also been published in the Journal of Big Data and it has received an issuance of a U.S. patent. As the Field-of-View (FOV) in remote sensing continues to expand, the number of targets observed by each sensor continues to increase. The ability to track large quantities of targets in real-time poses a significant challenge. Chapter 3 describes my colleague and I's contribution to the multi-target tracking domain. We have demonstrated that we can overcome real-time tracking challenges when there are large number of targets. Our work was published in the Journal of Sensors.

ACKNOWLEDGEMENTS

I would like to express my deep appreciation to Dr. Edwin Chong his advice on my Ph.D. thesis. I also would like to thank Dr. Steve Simske, Dr. Daniel Herber, and Dr. Ali Pezeshki for their service on my dissertation committee. Finally, I would like to thank my co-authors for my journal publications.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

# Chapter 1: Big Data Actionable Intelligence Architecture[1]

## 1.1 Introduction

The amount of data produced by sensors, Internet of Things (IoTs), social and digital media, are rapidly increasing each day [1]. The International Data Corporation expects that there will be 175 zettabytes of data worldwide by 2025 [2]. There is significantly more information as compared to the number of people analyzing it. This becomes a potential problem, where lots of data could get overlooked. Data storage, retrieval, and maintenance can become extremely costly due to the explosion of data. At some point, it might not be financially feasible to store all the data that is received. Hence, if data is not analyzed as it is received, the information collected could be lost forever. Decision support in a dynamic real-time environment using large volumes of structured, unstructured, and semi-structured data can be a research challenge [1]. Many Big Data analytic techniques such as regression analysis [3] and machine learnings [4] have been available for many years. However, data mining and data analytics [5] are post-event processes [6][7][8], which are inadequate to support real-time decision making. Actionable intelligence is the next level of data analysis where data are analyzed in near-real-time to create insights that support decision making [1]. In this chapter, we will discuss a Big Data Actionable Intelligence (BDAI) framework that can quickly turn real-time streaming data from a variety of sources into actionable insights. Our framework architecture has demonstrated the ability to integrate disparate data sources from a variety of interfaces in near-real-time. Our platform addresses the National Spatial Data Infrastructure Executive Order 12906 concepts by providing "the technology, policies, standards, and human

[1] The material from this chapter was published in [41].

resources necessary to acquire, process, store, distribute, and improve utilization of geospatial data." [9]. This paper is organized as follow. Section "2.0" provides a discussion on the data sources and exemplar we used to demonstrate our architecture. Section "3.0" goes over any related work in current open literature. Section "4.0" discusses our approach to the BDAI problem. Section "5.0" provides a discussion of the results in our project. Section "6.0" goes over the conclusion of our research.

## 1.2 Exemplar Problem

### *1.2.1 Exemplar Description*

To demonstrate our capability of transforming Big Geospatial Data to Actionable Intelligence in near-real-time, we focused on an exemplar problem of generating Actionable Intelligence in regard to the traffic congestion in the city of Chicago. The traffic prediction problem is extremely complex, which makes it hard to accurately predict traffic condition based on off-line data (patterns, trends, road networks, etc.) or crowdsourcing applications such as Waze [9] due to the dynamic changes of real-time environment (i.e. accidents, sport events, weather changes, etc.). This exemplar highlights the importance of Actionable Intelligence. For example, first responders need to safely and expeditiously transport a victim to the hospital. Rapidly identifying the fastest route to a medical facility increases the survivability of the victim. Actionable Intelligence provides timely information such as heavy traffic, which allows the first responders to make important time saving transportation decisions.

### *1.2.2 Data Sources*

Table 1 provides the data sources used to test the BDAI framework. Figure 1 provides a high-level pictorial illustration of each data types. The data sources were extremely diverse, in terms of data types and data frequency. Most of the data interfaces provided ways to geospatially constraint the results within the Chicago city limits. One of the data sources included a 3-hour ground truth dash camera video experiment to validate actionable intelligence created from our framework.

**Table 1: Heterogenous Data Sources**

| Data Sources | Source Type | Frequency | Description |
|---|---|---|---|
| Twitter [11] | Live text | • Live. Query every 5 mins | • Decahose – Geo-tagged tweets within Chicago city limits |
| Travel Mid-west [12] | Various | • Traffic camera images every 15 min<br>• Vehicle Detection System (VDS) every 10 min<br>• Dynamic Message Sign (DMS) every 10 min<br>• Thousands of camera locations | • Traffic Cameras<br>• VDS - Vehicle Speeds, Vehicle Occupancy<br>• DMS – Traffic times, Lane Closures, Accidents |
| City of Chicago [13] | Various | • Traffic Segments every 10-15 mins<br>• Traffic Region every 10-20 mins<br>• Construction Moratorium - Infrequent | • Traffic Segments – Vehicle Speeds, Vehicle Occupancy<br>• Traffic Region - Vehicle Speeds, Vehicle Occupancy<br>• Construction Moratorium – Road closures |
| GDELT [14] | Various | Every 15 minutes | Global Knowledge Graph – provides context and feeling between people, organizations, and locations |

| | | | |
|---|---|---|---|
| | | | Event Mentions, Events |
| MapQuest [15] | Various | Every 5 minutes | Reported Incidents |
| Digital Globe [16] | Satellite Imagery | 1-3 images a day | Satellite Imagery (limited number of images) |
| Dash Camera | 3-hour Video | Field experiment | Dash Camera Video (Live Experiment and Validation) |

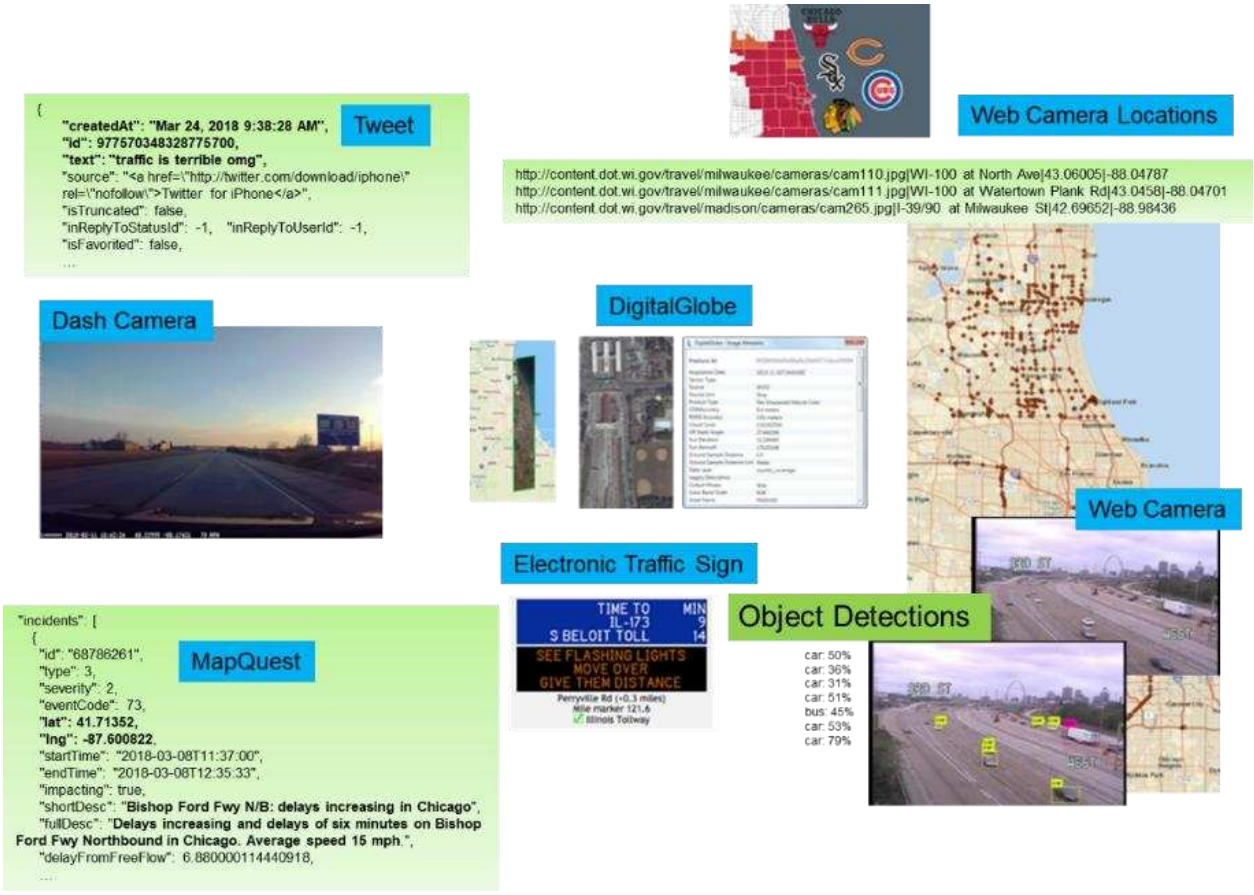



**Figure 1: Heterogenous Data Sources**

*1.2.3 System Requirement*

A summary of requirements and metrics that we used to evaluate our system is depicted in Table 2.

**Table 2: System Requirement**

| Requirements | Descriptions | Goal | Threshold |
|---|---|---|---|
| Scalability | Number of streaming location supported | 150 streaming location | 100 streaming location |
| Data Variety | Structured, Unstructured, Semi-structured | Structured, Unstructured, Semi-structured | Structured, Semi-structured |
| Average Throughput Per Location | Average data transfer rate per location | 1 Mbps per source Location | 0.50 Mbps per source location |
| Average Data Latency | Time measured from data creation to the time the data has arrived and indexed into our system | Less than or equal to the polling frequency | max (polling frequency, data update frequency) + 2 minutes |
| Data Management Guarantees | Level of guarantee on which message to be processed | Fully process each message | Drop message on failure |
| Traffic Classification Accuracy | Traffic Classification Accuracy | 95% accuracy on trained location | 90% accuracy on trained location |

*1.2.4 Assumption about Data*

We made the following general assumptions in regards to data:

1. Data can be referenced by time and geospatial extent.
2. Each data type may not follow a standardized format. Hence, architecture needs to accommodate needed flexibility to onboard new format.
3. Input data can come from variety of form (structured, semi-structured, or unstructured).
4. Data might not be immediately available for retrieval due to site restriction.
5. Data might not always be updated on a regular interval.

## 1.3 Related Works

Traffic prediction analysis is typically done in a crowd sourcing way, where location information from GPS apps are shared among users to help predict the fastest route [17]. Recently, improvement in traffic prediction accuracy using social media data has been demonstrated [18]. Despite many researches on traffic prediction [19], many existing research focuses on using few data sources for traffic prediction. Based on our research, we were not aware of any existing work utilizing a combination of data sources such as Twitter, web camera imageries, satellite imagery, dash camera video, Mapquest, and GDELT to support near-real-time traffic prediction. Our work uses seven disparate data sources as described in Table 1. Each data sources can be streamed from multiple locations. The web camera data in particular, involves the live streaming of over hundreds of camera locations around the City of Chicago. The traffic reports are received from hundreds of stations. Existing software architecture [20] typically focuses on acquisition, storage, and the retrieval of Big Data. However, our architecture focuses on Actionable Intelligence generations. Several data architecture has been proposed for network traffic monitoring applications [21][22][23], but our data architecture supports multiple disparate data sources. A general five-layer Big Data Processing and Analytics (BDPA) involves a collection layer, a storage layer, a processing layer, an analytic layer, and an application layer [24]. However, this architecture does not address actionable intelligence generation in their framework. In 2019, Zhu et al. states: “Currently, there are no widely accepted BDPA solution, especially a general-purpose solution fit for both traditional and internet industries [24].” Liu et al (2019) proposed a general multi-source framework [25] to map disparate data sources to a common unified data format for Big Data fusion. Their paper

suggested the benefits of combining heterogenous sources to provide a better solution, but it did not provide a solution on how this framework can be integrated with Big Data streaming sources. Hence, the motivation for our work focuses on using Big Geospatial Data to answer key customer geospatial and temporal questions. Big Geospatial Data is Big Data with geospatially tagged features and error estimates. As stated by the NIST Big Data Public Working Group (NBD-PWG), "Big Data consists of extensive datasets, primarily in the characteristics of volume, variety, velocity, and/or variability—that require a scalable architecture for efficient storage, manipulation, and analysis." [26]. While most Big Data information fusion solution focuses on social media data sources [27], our architecture accommodates a variety of geospatially tagged data sources at various velocities and veracities. Our traffic prediction exemplar allows us to test and validate key BDAI capabilities: handling heterogenous data sources, hosting data pipelines on distributed processing platforms, and running machine learning algorithms in near-real-time. The exemplar is not meant to compete with crowd sourcing GPS apps, but rather serve as a generic exemplar that can be extended to other Big Data Actionable Intelligence problems.

## 1.4. Methods

### *1.4.1 System Setup*

Our BDAI software was initially deployed to a bare metal system named "Ray". We deployed, configured, and tested the HORTONWORKS Data Platform (HDP) Apache Hadoop Distro [28] to the Ray cluster, composing of 120 computing nodes and 400 TB of Hadoop Distributed File System (HDFS) [29] storage. Since initial deployment, we have migrated our

BDAI software to run on a cloud infrastructure (Azure Stack [30]). Most of our custom data processing code is implemented in Java, [31] with some processing implemented in Python [32].

*1.4.2 BDAI Architecture Contributions*

A high level of our BDAI architecture is depicted in Figure 2. While a similar architecture has been proposed in open literature [20][24], these architectures focus on acquisition, storage and retrieval of Big Data, and on the use of specific datatypes [22][23]. The key question we want to answer in this paper is: Can we create a near-real-time data agnostic software architecture that can process many disparate sources while autonomously generate Actionable Intelligence? In order to combine and fuse disparate streaming data sources to produce actionable intelligence, we believe Big Data should be curated as it arrives to the system. Our main contributions to the Big Data Architecture field is listed as such: 1. Provide a general framework to map data from disparate data sources into a common frame of reference indexed by time and geo-spatial extent. This enables our architecture to stay data agnostic, which provides the possibility to quickly onboard new data sources that allows for agile responses to complete new and orthogonal scenarios. This method also provides the ability to ask questions generically over many disparate data sources, which minimizes the learning curve to perform meaningful fusion and analysis. 2. Provide a high-level description of our implementation in which our architecture uses a modern Big Data technology stack (depicted in Figure 3). This software stack is natively distributed and built for high-throughput streaming that allows us to tackle problems of mission-level magnitude. 3. Demonstrate and prove that our architecture

and technology stack are capable of supporting the streaming of disparate data sources to produce actionable intelligence.

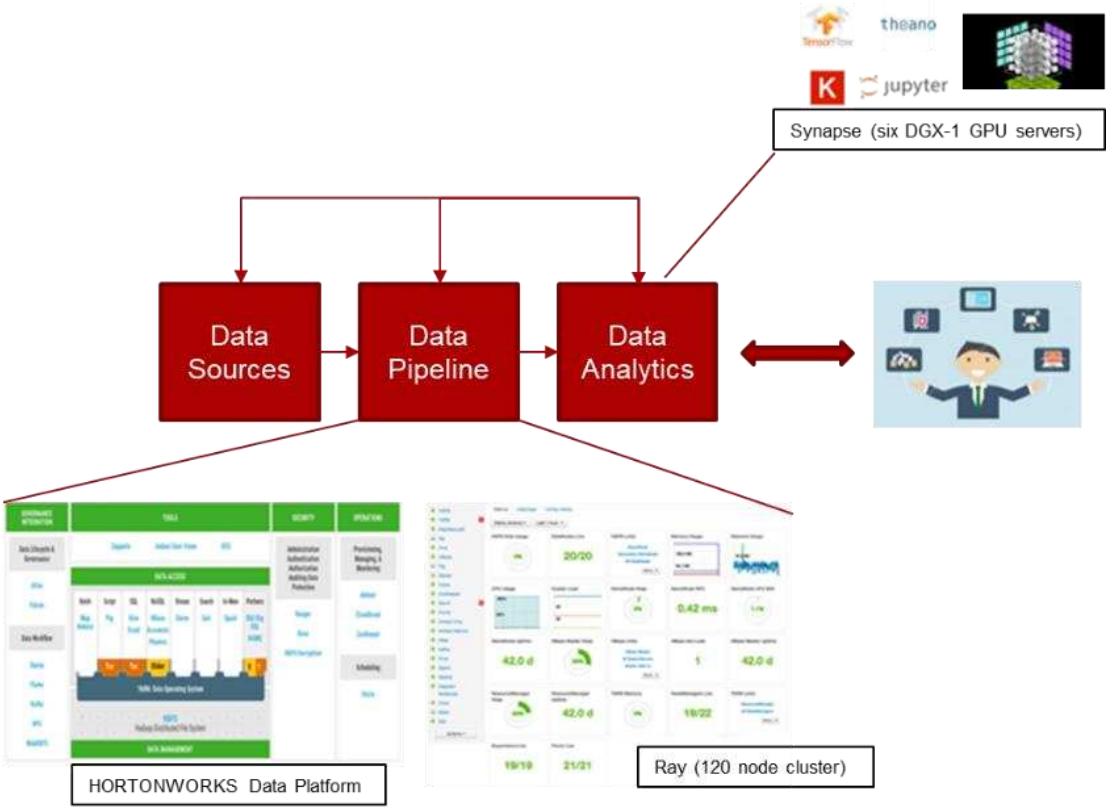



**Figure 2: BDAI SNL Architecture**


**Figure 3: Big Data Technology Stack**

*1.4.3 BDAI Architecture – algorithm workflow*

Our architecture contains four levels of processing: Data Source, Data Pipeline, Data Analytic, and Data Reporting. First, we set up a streaming interface connection for each data source. We utilized Apache Storm's topology [33] and Apache Kafka's [34] inter-process communication mechanism to implement our Data Pipelines because they are known to achieve a high level of scalability, low latency, fault-tolerant, and the data is guaranteed [35][36]. A general workflow of our data pipeline is depicted in Figure 4.

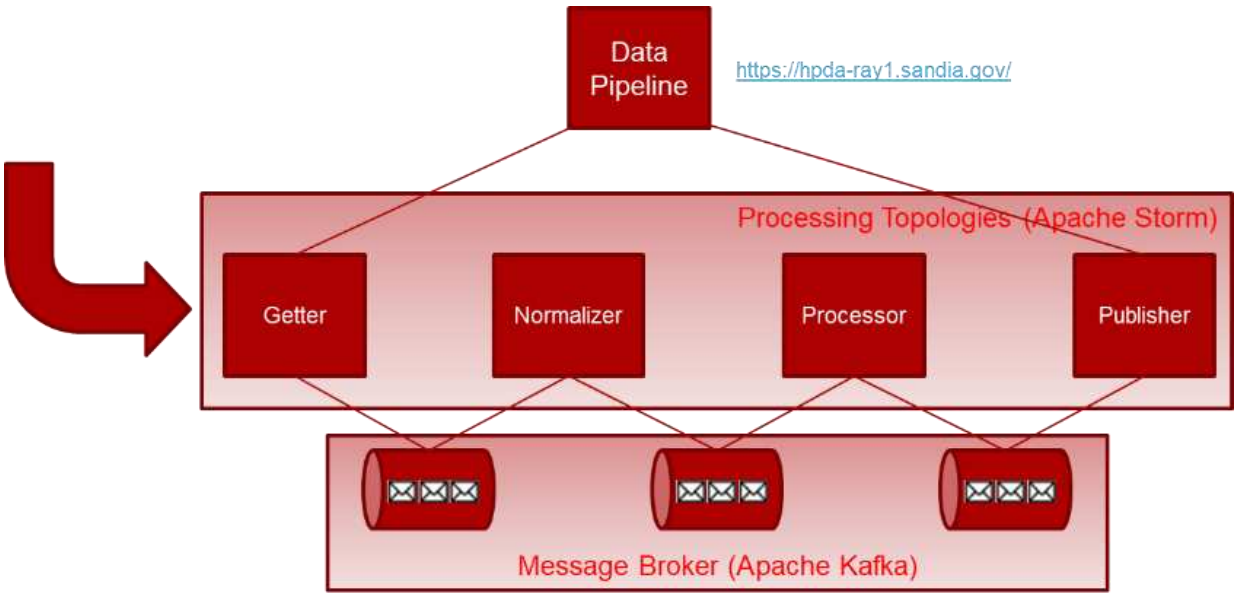



**Figure 4: Data Pipeline**

We created a separate processing Storm Topology [33] for each data type. Each topology follows a similar workflow of acquiring, normalizing, processing, and publishing the data (see Figure 4). Apache Kafka is used as a central messaging broker, connecting each step of the processing. For example, when incoming data arrives, it will first be placed in Kafka, and the "Getter" will be informed to obtain the data. The "Getter" is responsible for acquiring the data from an individual data source. The "Normalizer" is responsible for transforming the data by mapping out both raw data and metadata into a common event schema. A description of the event schema is depicted in Figure 5. The ontology mapping of each individual data source into a common event description is depicted Figure 6. The mapping of each individual data source into a common data schema is necessary to establish a common frame of reference for events that occurs at a given in time and space. This design makes searching for the events in a specific time or space to be easily accessible. All the data sources are "normalized" with the same common event schema, in which they are all "linked" by the time and its location. By

tagging the data in this manner, it ensures that the data can be discoverable by geospatial analytic processing in later steps.

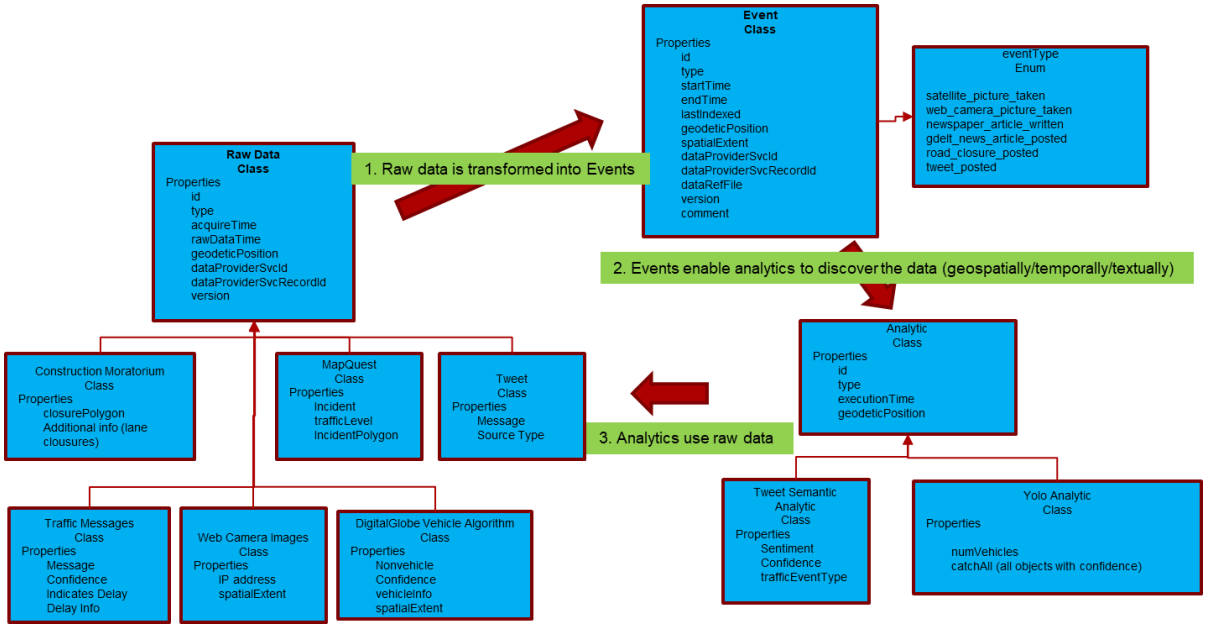



**Figure 5: Mapping Raw Data to generic Event Schema**

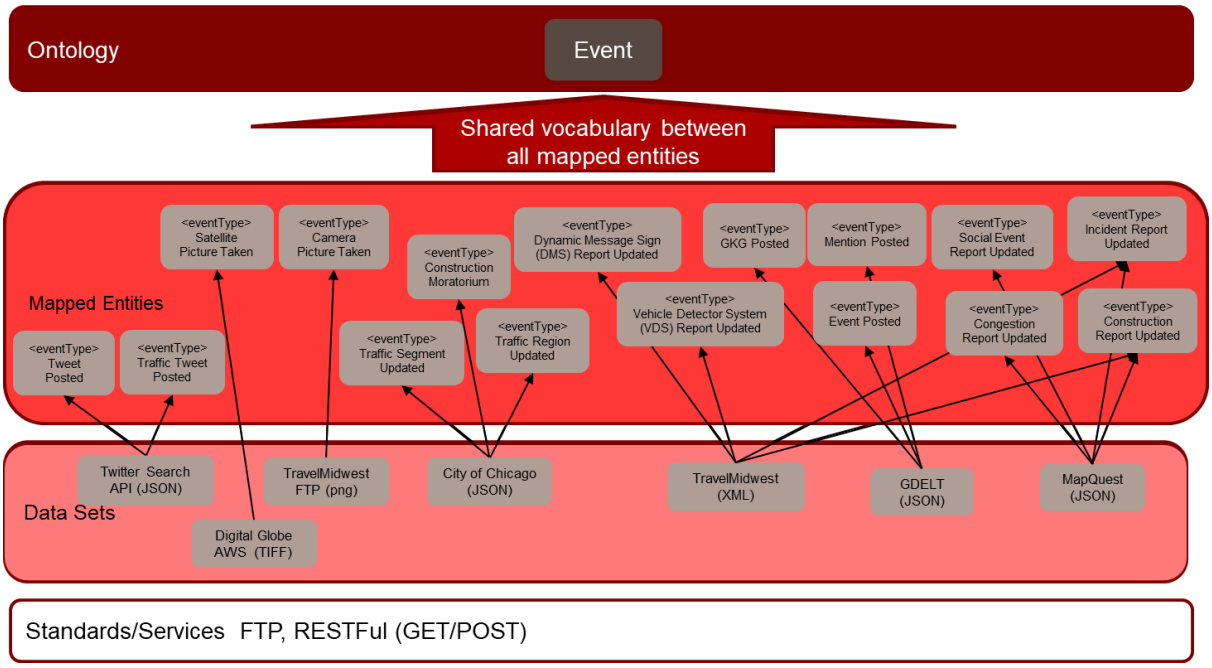



**Figure 6: General Event Schema**

The “Processor” is responsible for extracting events from raw sensor data and then populating its results in the event schema. The “Publisher” is responsible for “indexing” the data to enable search and discovery at the “Data Analytic” level. Apache Solr [37], an enterprise search engine, is used for both indexing and querying the geospatial and temporal data.

In our design, we developed a custom topology for each data type. The custom design provides flexibility to support different data types. An illustration of a web camera topology insertion is depicted in Figure 7. In this example, the “Processor” was built based on an object detection algorithm called You Only Look Once (YOLO)[38]. As depicted in Figure 8, the pre-trained YOLO processor did not yield good results. Hence, we labeled and re-trained YOLO using the web camera images from Travel Mid-West. Results of the re-trained YOLO processing are also depicted in Figure 8 as a comparison. The output of YOLO is used to determine the number of cars in each camera image. The event (i.e. number of cars at a location) generated from the YOLO topology is indexed by image time (when the image is captured) and image location (i.e. latitude and longitude of where the event occurred).

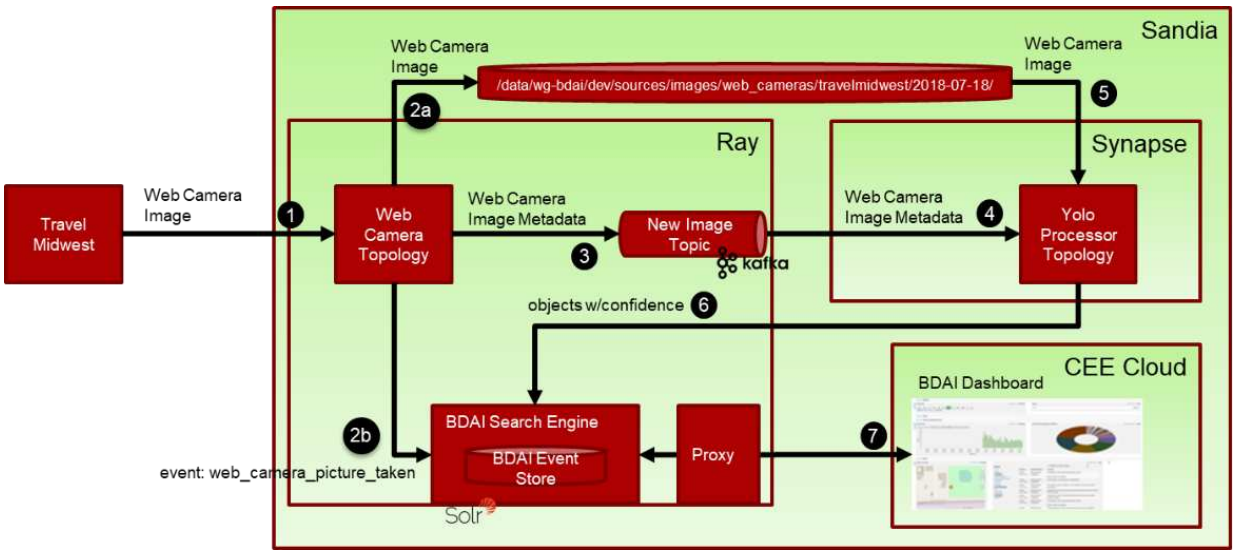



**Figure 7: Camera Topology Example**


**Figure 8: YOLO Results**

For the Tweeter Topology, we implemented a separate machine-learning "Processor" to process live tweets to generate traffic sentiment. Similarly, we indexed tweeted events by their time and location on where/when the events were tweeted. Following a similar workflow, we created separate topologies for each of the other data types as listed in Table 1.

*1.4.4 Muti-Source Data Fusion*

Information Fusion (IF) is a process of combining data or information to develop improved estimates or predictions of entity states[39]. Information obtained from a single

source can be unreliable or insufficient to make an accurate determination. For example, in one traffic scenario on the Dan Ryan Expressway Inbound between 87th St and 71st St on March 22, 2019, our YOLO topology had reported light traffic conditions because there were very few cars detected (see Figure 9). However, information received from our Tweet Processor indicated that the road was closed due to police activity (see Figure 10). Since the Tweet information had already been indexed by time and location, we could easily perform a geospatial query to obtain the Tweet's information to match the closest image time and location. Hence, the use of multiple data sources is necessary in order to improve the reliability and quality of the information provided to decision makers.

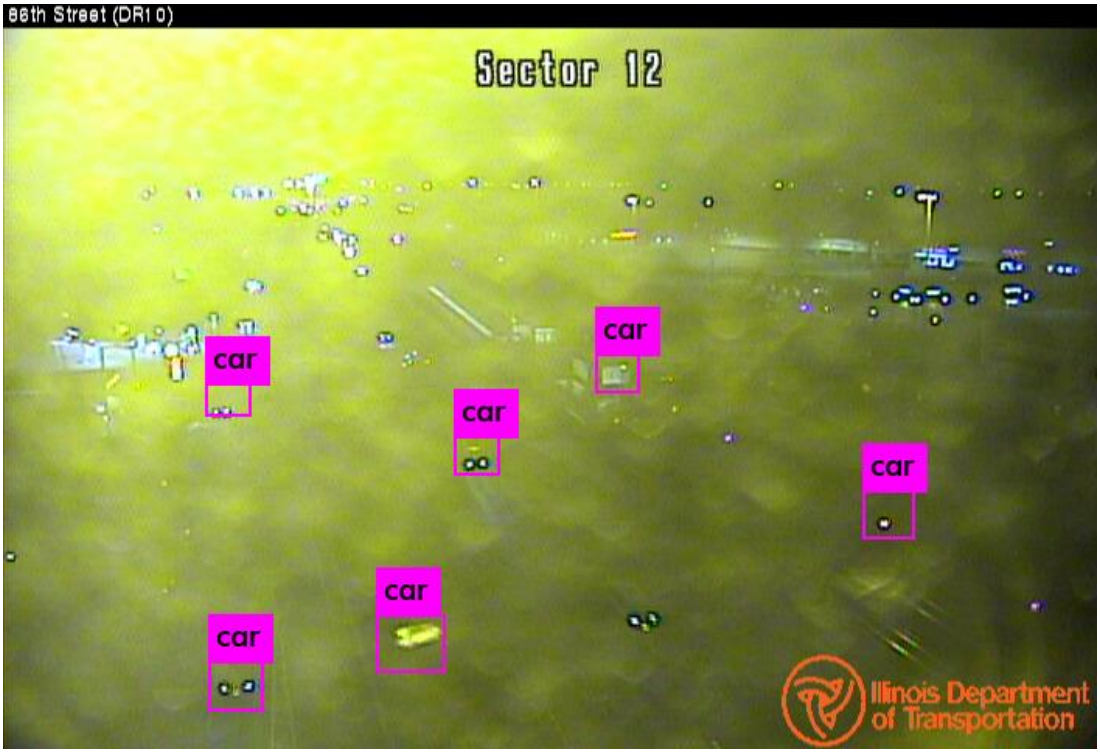



**Figure 9: Vehicle Detections reported by YOLO processor between 87th St and 71st**

```
tweet_traffic_posted:
      [
       {
        "comment": "Closed due to police activity in #DanRyan on Dan Ryan
Inbound between 87th St and 71st St #traffic #Chicago
https://t.co/GCesNZoWap",
        "dataProviderSvcId": "TTWN Chicago",
        "loc": "DanRyanat86thSt.",
        "endTime": "2019-03-22T08:44:23Z",
        "eventType": "tweet_traffic_posted",
        "id": "11090129394266685 45",
        "spatialExtent": "POINT (-87.62453 41.73621)",
        "dataRefFile": "1109012939426668545",
        "startTime": "2019-03-22T08:44:23Z",
```


**Figure 10: Tweets reported by Tweet processor between 87t St and 71st**

*1.4.4 BDAI Architecture Analytical Fusion Algorithm*

Our BDAI analytic seeks to combine event data from disparate sources to predict traffic congestion by improving the outcome beyond what could be done with a single source of information. At the data analytic level, we first query the normalized and curated data from all data sources by time and location. Then, we performed a data analytic on events occurring at similar times and locations. To demonstrate how machine-learning algorithm can be integrated into our architecture, we designed a Merged Neural Network (as depicted in Figure 11) to perform the traffic congestion classification. The algorithm takes input from all the normalized event data (related by time and location) to produce a traffic congestion probability. The output is a real-valued number between 0 and 1, as related to the level of traffic, where 0 is negligible traffic and 1 is a severe, complete standstill traffic jam.

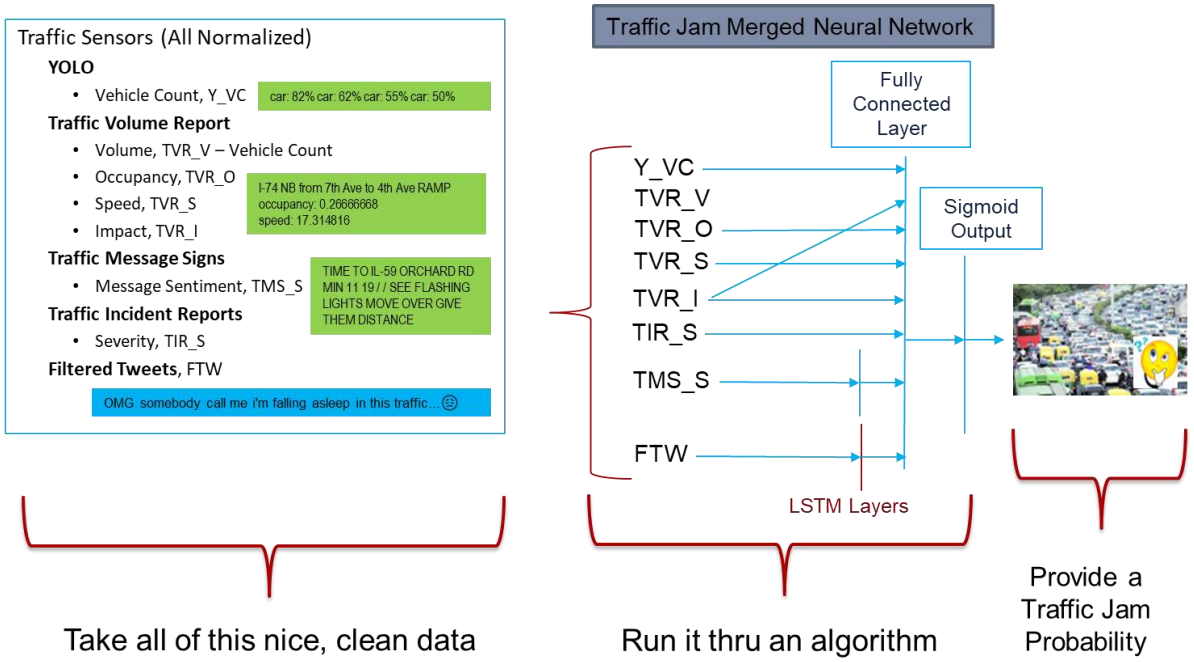
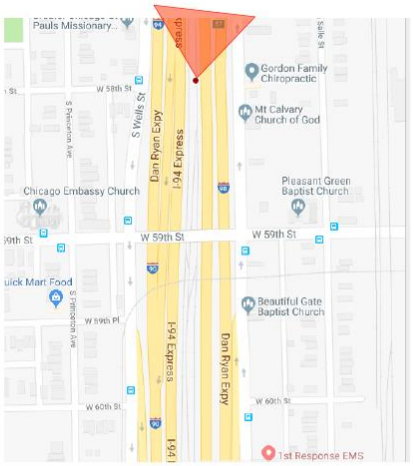



**Figure 11: BDAI Merge Neural Network**

## 1.5 Results and Discussion

### *1.5.1 Chicago Traffic Analytic – Multi-Source Analytical Fusion Demonstration*

A web camera image which captured the traffic condition on the Dan Ryan Expressway is depicted in Figure 12. At the corresponding time frame, our BDAI system was able to locate a tweet from the Total Traffic Chicago data source indicating that the road was closed due to an accident in the area (see Figure 13). At a similar time frame, the BDAI system had confirmed slow traffic through a traffic report from Mapquest (see Figure 14). However, Mapquest had reported that the West Dan Ryan Expressway had light traffic (see Figure 15). This information was also confirmed by the small number of cars detected (Figure 16) by our web camera topology. Taking into account all of the sources, BDAI was able to distinguish the traffic congestion level on both sides of the West Dan Ryan Expressway.



```
"id": "417dff01-921c-492b-a2b5-c2b45a4a2bfe",
"eventType": "camera_picture_taken",
"startTime": "2019-03-20T01:12:16Z",
"endTime": "2019-03-20T01:12:16Z",
"geodeticPosition": "POINT (-87.630990 41.788890)",
"spatialExtent": "POINT (-87.630990 41.788890)",
"version": "1.5",
"dataProviderSvcId": "travelmidwest",
"dataProviderSvcRecordId": "2019-03-20T01:12:16Z_DR6.jpg",
"dataRefFile": "/data/wg-bdai/dev/sources/images/web_cameras/travelmidwest/2019-03-20/2019-03-20T01:12:16Z_DR6.jpg",
"comment": "Dan Ryan at 58th St.",
"_version_": 1628485129289597000,
"lastIndexed": "2019-03-20T01:14:02.452Z"
```


**Figure 12: Camera Image Indicating Traffic Congestion on Dan Ryan**



```
"eventType": "tweet_traffic_posted",
"startTime": "2019-03-20T00:40:20Z",
"geodeticPosition": "POINT (-87.63108 41.7871)",
"dataProviderSvcId": "TTWN Chicago",
```

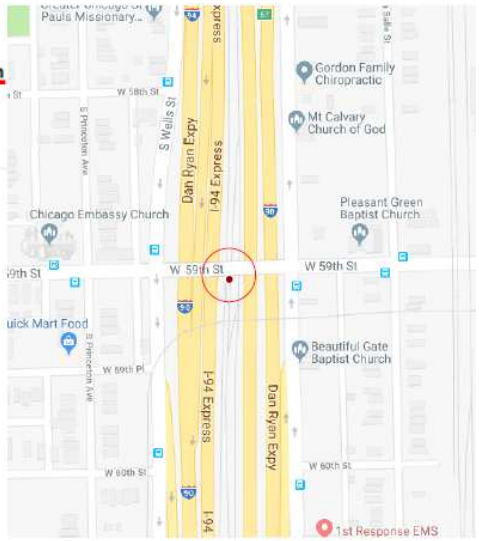
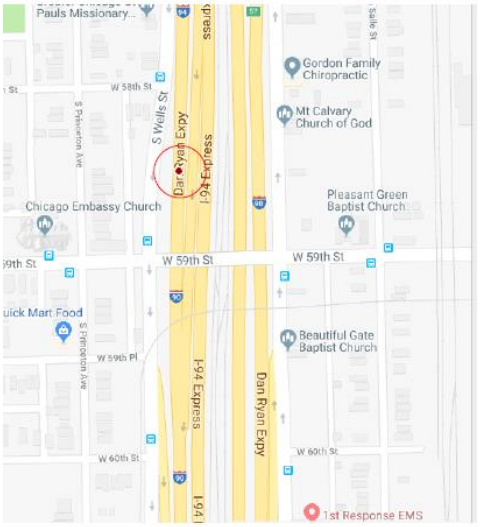



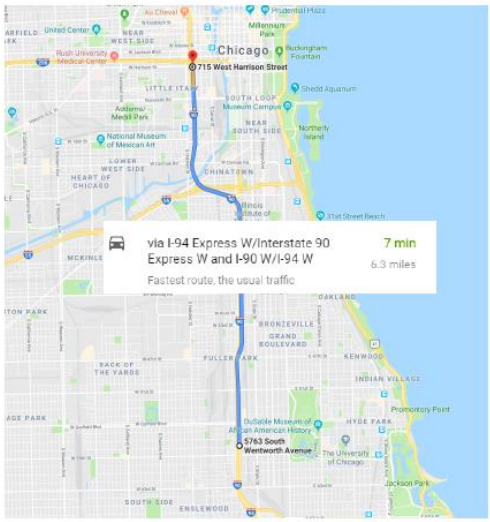

**Figure 13: Tweets indicated Dan Ryan Outbound at 59th St was closed due to an accident**



```
"startTime": "2019-03-20T00:31:33Z",
"eventType": "vds_report_updated",
"geodeticPosition": "POINT (-87.631590 41.788020)",
"occupancy": 19.971428,
"speed": 25.032557,
"volume": 368,
"streetName": "I-90",
"crossStreet": "59th St",
"trafficDir": "EAST_BOUND"
```

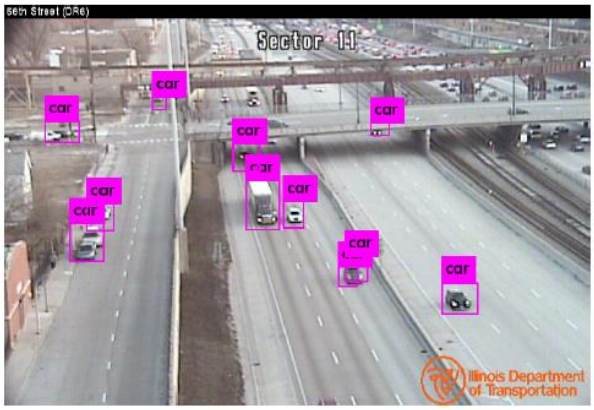

**Figure 14: Congestion Report from Map Request reported slow speed on EAST BOUND**



```
"startTime": "2019-03-20T00:40:11Z",
"eventType": "dms_report_updated",
"geodeticPosition": "POINT (-87.630840 41.789370)",
"occupancy": -1,
"speed": -1,
"volume": -1,
"streetName": "Dan Ryan Express Lane",
"crossStreet": "57th St",
"trafficDir": "WEST_BOUND",
"comments": "STOP THE TEXTS STOP THE WRECKS / 7 MINUTES TO JANE BYRNE INTERCHANGE"
```


**Figure 15: Congestion Report from Map Request reported light traffic**





```
"analyticType": "yolo object identification",
"analyticOutput": "car: 73% car: 65% car: 65% car: 56% car: 50% car: 47% car: 44% car: 40% car: 39% car: 35% car: 25%",
"processVersion": "yolo-v2"
```


**Figure 16: Small Number of Cars is detected East Bound Traffic**

*1.5.2 Traffic Classifier Performance*

Overall, the BDAI Merge Neural Network classifier performed extremely well on intersections where the network was trained. We also tested the BDAI Merge Neural Network classifier on intersections where it was not trained. As expected, the performance was not good. A summary of the performance of our classifier is depicted in Figure 17.

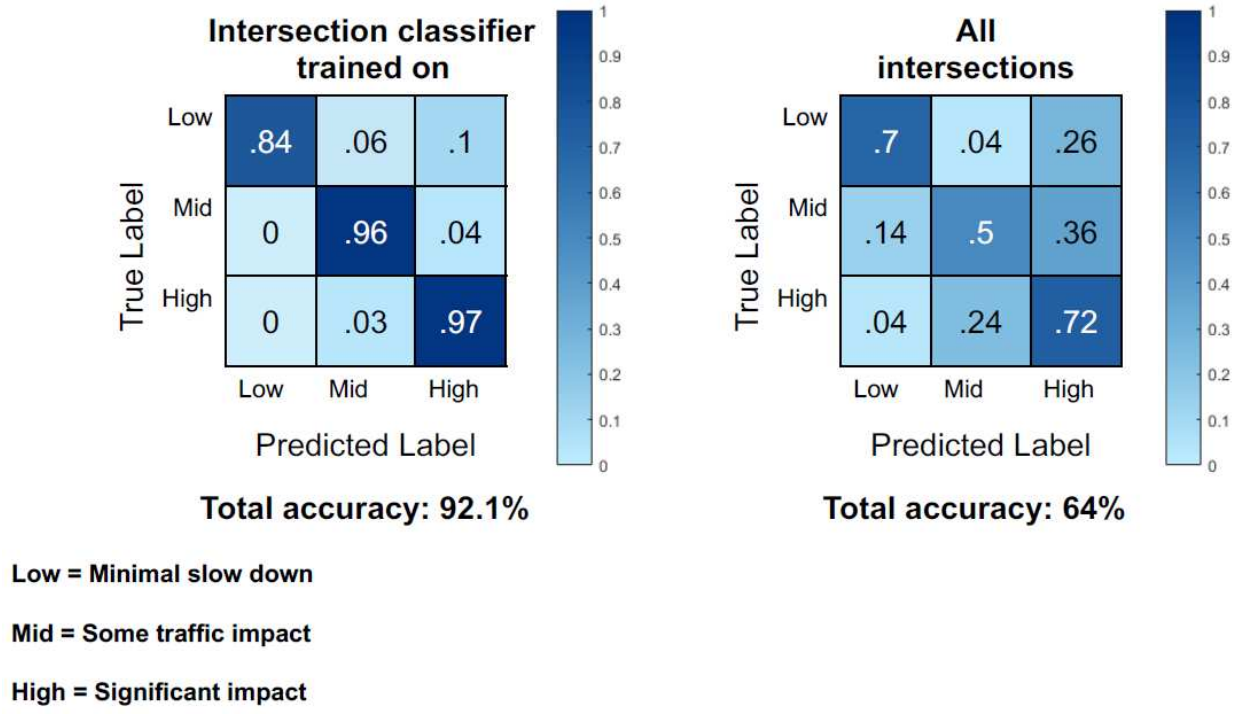



**Figure 17: Merge Neural Network Results**

*1.5.3 BDAI Dashboard*

The output of the BDAI system is visualized using a Banana Dashboard[40], as depicted in Figure 18. The BDAI Dashboard is back ended by an Apache Solr Cluster, which contains all event data. The map in the lower left represents the event records that were ingested in one of our data pipelines. The icons are the actual geospatial locations of the events. The event metadata is the table to right of the map.

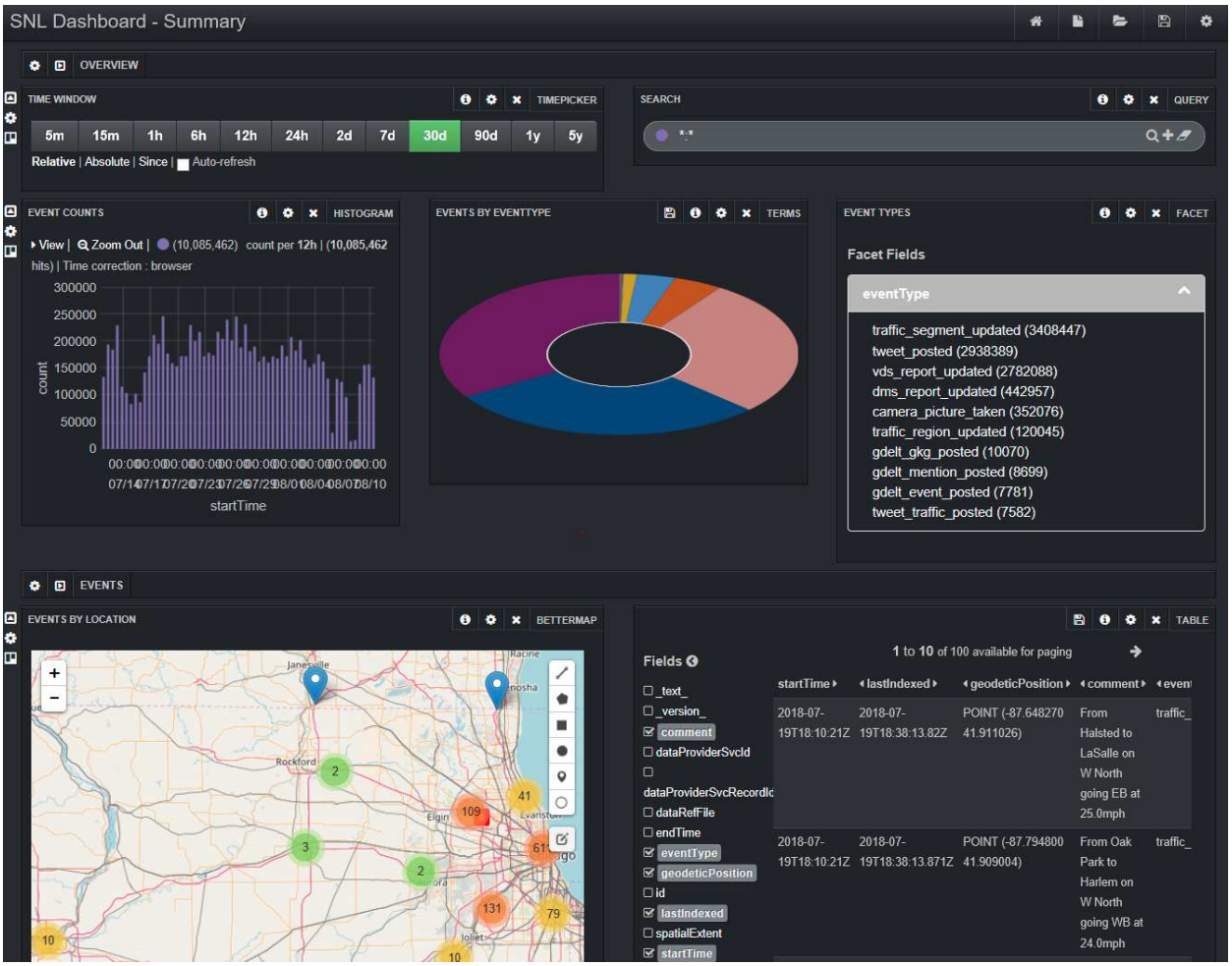



**Figure 18: BDAI Dashboard**

*1.5.4 System Performance*

Our BDAI software was deployed to a bare metal system named “Ray” (Section 4.1). A summary of the system performance from “Ray” for all event types is depicted in Table 3. We are not aware of any similar systems that are published in open literature to draw a direct comparison from our effort. The missing entries in the table are due to insufficient information in that particular event type to derive statistics. Each event type can have multiple sources as there may be multiple camera locations or traffic report stations active at a given time. Our data architecture supports concurrent streaming from each data sources. Each event type

restricts how frequent we can “poll” the data. Hence, “polling” is not done instantaneously when the event is available, but rather done at a fixed time interval, as permitted by the external source. This is not a limitation in our architecture, but rather a limitation set forth by an external data source. Latency in Table 3 is measured from the time an event happens, to the time that the event is curated and indexed into Solr. This does not account for any additional latency required by downstream analytic processing. Once the data is indexed into Solr, the data is immediately available to perform any sort of analysis. Some events, such as the web camera imagery requires additional processing (i.e. using the YOLO processor). The time for data processing highly depends on the specific type of algorithm implemented. Our Merge Neural Network (section 4.4) used for actionable intelligence generation performs a poll from “Solr” every 15 minutes. All information retrieved over the time interval are used to create actionable intelligence. The execution time for the Merge Neural Network is negligible (within a millisecond). The “polling” period is not a limitation in the architecture, but it is an adjustable parameter depending on the arrival time of each individual data sources. The polling rates for each topology is depicted in Table 4. The overall turnaround time for actionable intelligence generation is mainly driven by the availability of data sources and the frequency we poll the data since actual data processing is deemed negligible.

**Table 3: System Performance**

| eventType | avg num of locations | avg record size (bytes) | avg daily record total | total num of records | avg latency (min) | avg throughput (bytes/sec) |
|---|---|---|---|---|---|---|
| tweet_posted | 1 | 3835 | 132305 | 16875132 | 5.5 | 5873 |
| traffic_segment_updated | 818 | | 117957 | 4362857 | 29.9 | |
| vds_report_updated | 818 | 639 | 117250 | 2305651 | 4.7 | |
| gdelt_gkg_posted | | 109388 | 4254 | 1104740 | 3.0 | 5386 |
| gdelt_mention_posted | | 7509 | 4504 | 1034357 | 1.0 | 391 |
| camera_picture_taken | 150 | 350000 | 14041 | 516782 | 2.2 | 56879 |
| dms_report_updated | 150 | 2200 | 20906 | 508288 | 34.9 | 532 |
| gdelt_event_posted | | 1204 | 1793 | 217691 | 1.7 | 25 |
| traffic_region_updated | | | 4156 | 153555 | 37.3 | |
| tweet_traffic_posted | 1 | 3835 | 561 | 68124 | 5.8 | 25 |
| construction_moratorium | | | 1000 | 37000 | | |
| congestion_report_updated | | | 74 | 28092 | 28.6 | |
| incident_report_updated | | | 238 | 9346 | | |
| construction_report_updated | | | 99 | 800 | | |
| social_event_report_updated | | | | 225 | | |

**Table 4: Topology Polling Rates**

| Topology | Polling Rate |
|---|---|
| ChicagoTrafficTrackerTopology | no more frequently than 10 minutes (~between 10 and 12 minutes) |
| XmlTopology | no more frequently than 10 minutes (~between 10 and 12 minutes) |
| MoratoriumTopology | 24 hours |
| CamerasTopology | 15 minutes |
| GDELT | 15 minutes |
| MapQuestTopology | 5 minutes |

| | |
|---|---|
| TweetTopology | every 5 or 15 minutes subject to twitter rate limits |

*1.5.5 Performance vs Requirement Discussion*

In regards to the original requirement as depicted in Table 2, our system has achieved the scalability and flexibility needed for Big Data processing. We have demonstrated that our system is horizontally scalable to hundreds of locations. For example, the data we ingested include: traffic segments received from 818 stations, vehicle detection system reports received from 818 stations, images received from 150 camera locations, and dynamic message signs reported from 150 stations. We ingested an average of 132,000 tweets a day, 14,000 camera images a day, and 10,000 posts from Gdelt. A comparison breakdown of the statistics for requirement analysis is depicted in Table 5. The majority of the data met our requirement specification. The only exception is the dynamic message sign report topology. The larger latency was associated with an inconsistent update interval provided in the server rather than the actual latency in our system. As depicted in Table 5, the overall latency performance of each data types are largely driven by external site restrictions on how frequent we are allowed to query the data. Despite this restriction, most data sources had an average latency less than the "polling" time. It is possible that the latency can be further reduced if the data can be pushed to the consumer at a higher rate. Evaluation of this architecture using a different application exemplar with real-time accessible data would be left for future exploration.

**Table 5: Latency Performance vs Requirement**

| Event Type | Avg Record Per Day | Num of Source Station | Query Freq (min) | Polling Freq (min) | Average throughput (bytes/sec) | Average Latency (Measured) | Status |
|---|---|---|---|---|---|---|---|
| Tweets | 132K | 1 | 5 or 15 (subject to rate limit) | 5 or 15 minutes subject to rate limit) | 5873 | 5.5 min | Met Goal |
| Tweet Traffic Posts | 0.5K | 1 | 5 or 15 (subject to rate limit) | 5 or 15 (subject to rate limit) | 25 | 5.8 min | Met Goal |
| Camera Images | 14K | 150 | 15 | 15 | 56879 | 3 min | Met Goal |
| Gdelt Global Knowledge Graphs | 4.2K | 1 | 15 | 15 | 5386 | 3 min | Met Goal |
| Gdelt Mention Posts | 4.5K | 1 | 15 | 15 | 391 | 1 min | Met Goal |
| Gdelt Event Posts | 1.7K | 1 | 15 | 15 | 25 | 1.7 min | Met Goal |
| Dynamic Message Sign Report | 20.9K | 150 | 15 | 10-12 | 532 | 34.9 min | Failed |

## 1.6 Conclusion

In conclusion, our big data architecture provides a framework for machine-learning algorithms to learn and analyze streaming data (e.g. near real-time analytics) from heterogenous data sources (texts, signal waveforms, images, videos) to turn them into actionable information for decision makers. Our data-agnostic solution is accomplished by mapping different data types into a common frame of reference that requires both temporal

and geospatial metadata. We have demonstrated through a traffic prediction exemplar that our architecture can support actionable intelligence generation in near-real-time using disparate data sources. Our traffic prediction exemplar allowed us to test and validate key BDAI capabilities: handling heterogenous data sources, hosting data pipelines on distributed processing platforms, and running machine learning algorithms in near-real-time. Our BDAI platform was designed with flexibility in mind, allowing us to quickly onboard new data sources and apply machine learning algorithms. Our data platform's agility and common frame of reference allows us to rapidly provide Actionable Intelligence to our customer's mission relevant problems. The framework architecture is a generalized architecture that can enable solutions for other BDAI problems with similar data diversity and data volume. The BDAI architecture has been fully implemented into a software system that is currently running and is hosted at Sandia National Laboratories for over a year. Our work has been featured on the local news media [1]. The current BDAI system can produce first order of data analytics (i.e. combining data from multiple source to assess what is happening at current time). In the future, we plan to further develop statistical techniques such as minimum variance to optimize the resultant estimate. In addition, we plan to extend BDAI's capability to include a second order of analytics by providing the decision maker with a list of suggested actions, based on the assessment of the current situation using multiple data sources.

# Chapter 2: Remote Sensing Detection Enhancement[2]

## 2.1 Introduction

Big Data in the area of Remote Sensing has been growing rapidly. Remote sensors are used in surveillance, security, traffic, environmental monitoring, and autonomous sensing. Data from remote sensors can be fused with other data sources to generate actionable intelligence for decision makers [41]. However, real-time detection of low signal-to-noise-ratio (SNR) small moving targets using a remote sensor has been an ongoing, challenging problem. Since the object is located far away, the object often appears too small on the sensor. The object's SNR is often very low. Occurrences such as camera motion, moving backgrounds (e.g. rustling leaves), low contrast and resolution of foreground objects makes it difficult to segment out the targeted moving objects of interest. Due to the limited appearance of the target, it is difficult to obtain the target's characteristics such as its shape and texture. Without these characteristics, filtering out false detections can be a challenging task. Detecting these targets, would often require the detector to operate under a low detection threshold. However, lowering the detection threshold could lead to an increase of false alarms. In this paper, the author will introduce a new method that improves the probability to detect low SNR objects, while decreasing the number of false alarms as compared to using the traditional baseline detection technique. This chapter is organized into the following sections: Section 2.2 provides a discussion on related research work published in open literature; Section 2.3 focuses on the

[2] The material from this chapter was published in [99].

data used for our experimental results; Section 2.4 is a detailed discussion on our methods used; Section 2.5 goes over our results; and Section 2.6 is a summary conclusion of the paper.

## 2.2 Related Works

Recent artificial intelligence (AI) based object recognition methods uses a deep learning approach to segment objects in motion [42-44]. Deep learning algorithms can now achieve human-level performances on a variety of difficult tasks. Success on these methods generally require large training datasets [45] with quality features [45,46]. However, deep learning methods have several drawbacks; including understanding its reasoning in making decisions. These methods are unclear to a human observer [47,48] and requires large training data sets [45]. This can often lead to unexpected results when real data does not resemble those in the training datasets [49]. Moreover, studies have shown that these methods are vulnerable to adversarial attacks [50], fooling deep learning models to mis-classify objects [50,11]. Resolving these challenges are especially important in time critical security surveillance application, where failure to detect a target could result in high consequences.

Traditional change detection methods rely on background subtraction [52]. The background is continuously updated as each new frame is received. For each new frame that comes in, the estimated background will be subtracted from the new frame to produce a "Difference Image". Thresholding can then be applied on the Difference Image to identify changes in the scene [53-56]. Though these methods do not require pre-trained labels, they generally have a higher false alarm rate. Traditional background estimation can typically be categorized into two categories: pixel-based approaches and dimension reduction approaches.

The popular pixel-based approach - Gaussian Mixture Model (GMM) [57], is known for its simplicity in implementation. Dimensional reduction approaches such as Principle Components of Pursuit (PCP) [58] and other subspace estimation approaches [59] are known for their robustness on handling camera jitters and light illumination.

Despite research advances in background modeling and AI object recognition approaches, these methods alone are inadequate to detect low SNR targets. Low SNR targets with limited features and training examples are not ideal for machine learning problems. Lowering the threshold using traditional background subtraction detection, could cause an increased number of false alarms.

Thus far, Velocity Matched Filter (VMF) techniques have been introduced [60,61] and it appears to be extremely effective in improving the detection and tracking of low SNR targets. These methods enhance the target's signal and noise ratio (SNR) by integrating the target's energy over a period of time. It also matches the target's true velocity and direction of where it is traveling. Numerous Track-Before-Detect (TBD) algorithms [62-66] have evolved using VMF. TBD algorithms have been studied in a variety of remote sensors such as Radar [63,64 ,66 ,67], Sonar [68], and Infrared (IR) [69]. While earlier publications have demonstrated the effectiveness of using TBD algorithms in single target scenarios, it has also been used for tracking multiple targets, [65,70] generally requiring the number of targets to be known ahead of time. Dynamic programming algorithms for performing VMF without a known target velocity has also been developed [66,71]. While dynamic programming techniques seem to overcome run-time performance limitation, it tends to degrade on maneuvering targets due to the lack of motion modeling. More recently [72], motion models have been incorporated in TBD

techniques to better handle target maneuver, but the challenge remains on how to initialize tracking. Many of the published techniques are only demonstrated with simulated data.

Though VMF or TBD type techniques have shown to be theoretically appealing in low SNR target tracking scenarios, there are several challenges that make these approaches difficult to apply in the real world. First, the algorithm assumes a priori knowledge in which the number of targets are known or a priori knowledge to initialize the track. This knowledge is often not provided in real-time surveillance application. Recent advances made to [62,73] eliminate these problems are by inserting an additional detection process before the TBD processor. However, initiating a low SNR target track is a challenge for this framework. If the target is not detectable by the threshold set by the initial detector process, then TBD will not start. On the other hand, setting a low detection threshold to force TBD to initiate could lead to many false alarms. Secondly, to gain the maximum benefit of the TBD approach, it assumes the noise is white [61]. Nonstationary changes such as jitters or sensor noises which are not removed, can introduce correlated noises that can potentially degrade VMF's performance. Finally, VMF assumes that the objects can be seen and that finding the best matched hypothesis is always available. However, in a real-world situation, objects may not always be observable by the sensors. For example, a person's movement can be observed by the sensor, but not when the person is walking behind a big tree. In this situation, finding the best matched filter becomes difficult because all the match hypotheses could be invalid.

## 2.3 Experimental Setup

To evaluate our technique in a real-world scenario, a video camera [Table 6] was placed at the top of Sandia Mountain [Table 7] to collect live traffic data. The distance of the camera to the target on the ground was approximately 4000 feet. Since the image is large, [Figure 19] only shows a cropped off portion of the image. The targets were barely visible, and they appeared like small dots. The size of the target(s) in the image ranged from 4 pixels to 20 pixels. The camera video is subjected to jitter motion naturally induced by the wind. This setup allowed us to evaluate the challenges of remote target detection under a real-world natural setting.

**Table 6: Camera Specifications**

| Video Camera | Frame Rates | Image Resolution | Lens focal length |
|---|---|---|---|
| Mysterium X | 24 frames per second | 3072x1620 | 72mm |

**Table 7: Experiment Location**

| | |
|---|---|
| Camera Location | Peak of Sandia Mountain, Albuquerque, New Mexico |
| Camera Location Elevation | 10,379 feet |
| Ground Target Elevation | 6060 feet |

**Figure 19: A cropped image captured by the video camera**

2.4 Methods

To overcome the limitations of existing TBD methods, this paper provides the following key contributions. 1. An ideal “Normalized Difference Frame” calculation to perform VMF enhancement; and 2. A novel Constrained Velocity Matched Filter (CVMF) that combines known

physical constraints with the target's dynamic motion constraints to enhance its SNR. Our processing workflow is summarized as shown in [Figure 20].

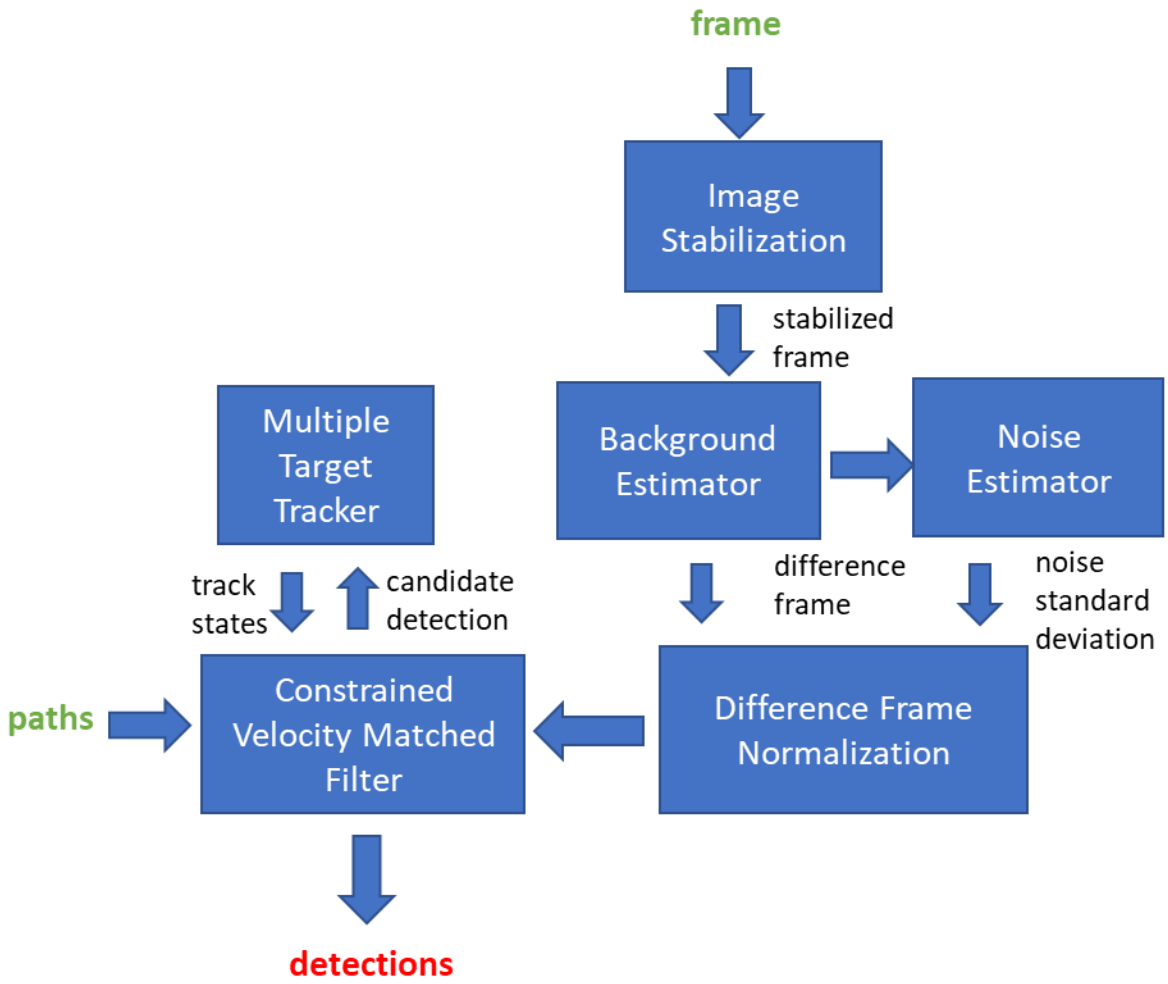



**Figure 20: Processing Workflow**

*2.4.1 Image Stabilization*

To eliminate motion jitters on the camera induced by wind, we used the first frame of the video as a reference frame and registered subsequent frames from the video onto the reference frame. This was accomplished by using a frame-to-frame registration technique as described in [74] to create a stabilized frame.

*2.4.2 Background Estimation*

The stabilized frame was then fed to a temporal background estimator so that the background subtraction could be performed. The process of background subtraction can be expressed mathematically using the following equation:

$$D(t) = F_s(t) - B(t-1) \quad (1)$$

where $D(t)$, corresponds to the difference frame at time $t$, $F_s(t)$ corresponds to the stabilized frame at time $t$, and $B(t-1)$ corresponds to the background computed in the previous time step. For simplicity in implementation, the popular Gaussian Mixture Modeling (GMM) background estimation method [57] was used in our processing. However, it is important to note that our method can also be applied to other temporal background estimation methods such as the Principal Component of Pursuit [58], and Subspace Tracking techniques [59].

*2.4.3 Noise Estimator*

In general, a background cannot be perfectly estimated regardless of which background estimation method used. Hence, it is important to model a deviation of background models. To model the estimated background deviation, we estimated the temporal variance $v$ of frame pixel location $(i, j)$ at each time step $t$ using an Infinite Impulse Response (IIR) filter with the following equation:

$$v(i,j,t) = (1-\gamma)\, D(i,j,t)^2 + \gamma\, v(i,j,t-1) \quad (2)$$

where $\gamma$ is the variance update rate [0,1]

The temporal standard deviation for pixel $(i, j)$ at time $t$, is obtained using the following equation:

$$\sigma(i,j,t) = \sqrt{v(i,j,t)} \tag{3}$$

*2.4.4 Difference Frame Normalization*

Pixels in different parts of an image can have different temporal standard deviation, depending on factors such as the environment and the scene structure. For example, the temporal standard deviation of the pixels in the waterfall region with constant running water is much higher than the pixels of an empty field. Hence, it is important to normalize the Difference Frame with respect to its temporal noise estimation before any thresholding is applied. The Normalized Difference Frame $N_d$ for frame pixel location $(i,j)$ in time $t$ is expressed as follow:

$$N_d = \frac{D(i,j,t)}{\sigma(i,j,t-1)} \tag{4}$$

While numerous existing methods attempt to detect objects on difference frames [53-56], our method attempts to find objects on the Normalized Difference Frame.

2.4.5 Constrained Velocity Matched Filter

The Constrained Velocity Matched Filter (CVMF) uses a combination of physical constraint and motion estimation constraint to find, match, and integrate target signals along a motion path to enhance the target's SNR. Performing operation on the Normalized Difference Frame is more ideal because it reduces the risk of enhancing the noise on high noise region areas (e.g. high scene contrast region, waterfalls, etc.). For detecting vehicles in this video, a physical road constraint is imposed in the CVMF processing. However, for other applications,

other constraints can be used, such as railroads for trains, pathways inside a building. A summary of the CVMF method is depicted in [Figure 21].

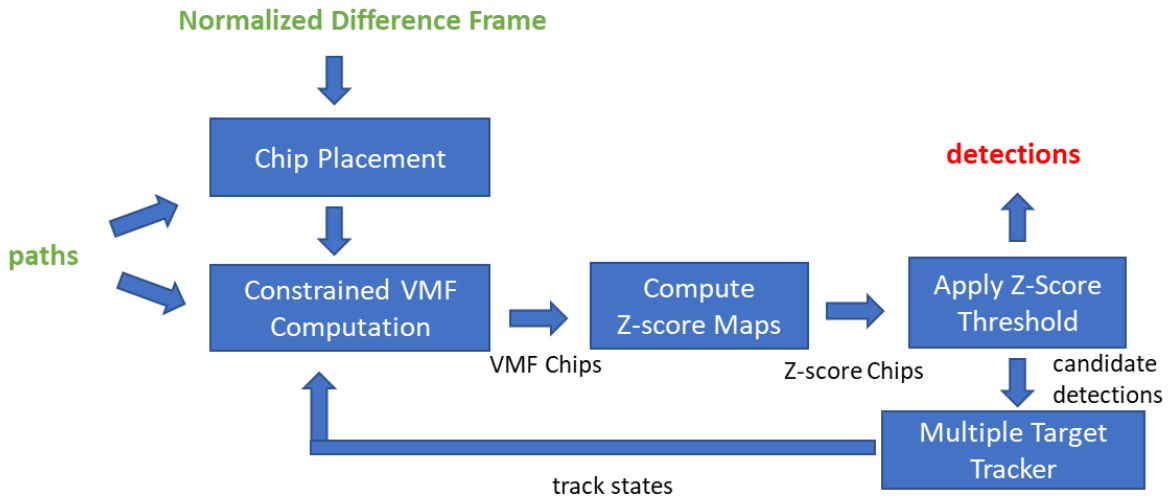



**Figure 21: Constrained Velocity Matched Filter Process**

Given the road constraints, we divided the path into different numbers of processing region (called "chips") along the road in the Normalized Difference Frame. An illustration is shown in [Figure 22]. The size of each "chip" used was 65x65 pixels. In general, the size of the "chip" should be selected based on the knowledge of the target's size and the path. For example, the region selected should be big enough to cover the width of the path with enough margin to account for path uncertainties. In addition, the region should be large enough to include non-targeted areas.


**Figure 22: Constraint Processing Illustration (processing region is denoted by red box, road path is denoted by green line)**

The continuous VMF process [60,61] can be implemented in a discrete form, by shift-and-add operation with different velocity hypotheses along the path region in both forward and backward direction. For instance, suppose an object's movement is within the camera's view over a sequence time step as illustrated in [Figure 23]. For each processing chip, we can perform a range of shift-and-add operation for a range of velocities in attempt to match the target's movement over a period of time [Figure 24].

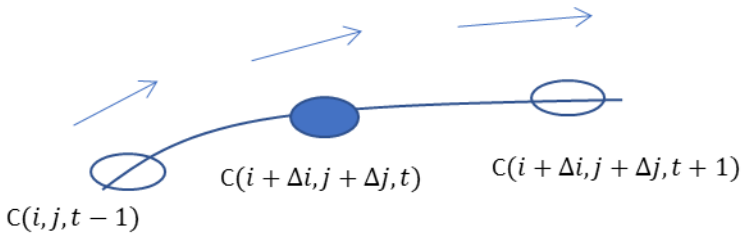



**Figure 23: Example of an object's movement multiple time steps**

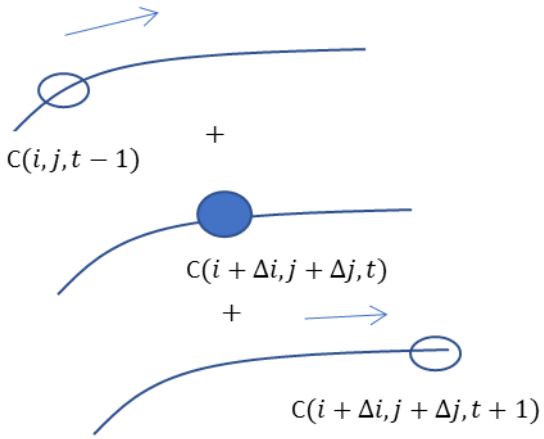



**Figure 24: Shifting and Adding Operation**

The "sum chip" is the summation of individual chips over the temporal window.

Mathematically, this can be expressed as the following:

$$S_k(i,j,t) = C(i+\Delta i, j+\Delta j, t-w) + \cdots C(i,j,t) + \cdots C(i+\Delta i, j+\Delta j, t+w) \quad (5)$$

where $S$, is the summation of the pixel $(i,j)$ across multiple frames. $(\Delta i, \Delta j)$ corresponds to the shift positions, and $w$, represents the frame window for the summation, and $k$ corresponds to the index of the matched hypothesis. The total number of matched hypothesis $K$ can be expressed as:

$$K = M * N \quad (6)$$

where $M$ is the number of directional hypotheses and $N$ is the number of velocity hypotheses. Since the movement of the individual targets are constrained in a pre-determined path, $M$ is 2 in most cases (either forward or backward direction). $M$ can be greater than 2 when the chip is at the intersection. The number of velocities depends on the target's speed. The units of the velocity in the target's movement can generally be described in fractions of pixels per frame.

We started with an initial set of velocities and allowed for further refinement once a track has been established.

To find the detection in the sum chip $S$ for a given hypothesis $k$, we first normalized the sum chip to form a Z-score chip. We can do this by computing the mean $\mu_s$ and standard deviation $\sigma_s$ of the sum chip $S$. For dense target scenarios, it is recommended that a trim mean is used instead, to avoid high SNR targets inflating the mean estimates.

$$\mu_s = \frac{1}{p}\sum_{p=1}^{P} S(p) \tag{7}$$

$$\sigma_s = \sqrt{\frac{1}{P}\sum_{p=1}^{P}(S(p) - \mu_s)^2} \tag{8}$$

Then, we compute the $Z$ score of the sum chip $Z_s$ for each pixel $(i, j)$ using the following equation:

$$Z_s(i,j) = \frac{S(i,j) - \mu_s}{\sigma_s} \tag{9}$$

The following thresholding logic is applied to perform detection.

If $(|Z_s(i,j)| \geq T)$, then pixel $(i, j)$ is a candidate detection

Pixel detected locations are generated from all hypotheses. They are consolidated to eliminate redundant detections from each chip. Adjacent pixel detections are clustered to represent a single target.

The centroid of the target's cluster is then fed to the Multiple Target Tracker (MTT) for association and tracking. To simplify, MTT is implemented using a simple 4-state constant

velocity model [75]. An object's dynamic movement can be expressed mathematically using the following equations:

$$\boldsymbol{x}(t) = \boldsymbol{A}\,\boldsymbol{x}(t-1) + \boldsymbol{q}(t-1), \qquad \boldsymbol{q}(t) \sim N(0, \boldsymbol{Q}) \tag{10}$$

$$\boldsymbol{y}(t) = \boldsymbol{H}\,\boldsymbol{x}(t) + \boldsymbol{r}(t), \qquad \boldsymbol{r}(t) \sim N(0, \boldsymbol{R})$$

where $\boldsymbol{x}$ corresponds to the state vector, $\boldsymbol{y}$ corresponds to the output vector, $\boldsymbol{A}$ corresponds to the system matrix, and $\boldsymbol{H}$ corresponds to the output matrix. The system includes additive process noise $q$ and measurement noise $r$, which are modeled as white noise gaussian with zero mean**.** The constant velocity model can be expressed in the following form:

$$x_1(t) = x_1(t-1) + \Delta T x_3(\mathrm{t}-1) + \mathrm{q}_1 \tag{11}$$

$$x_2(t) = x_2(\mathrm{t}-1) + \Delta T x_4(t-1) + q_2$$

$$x_3(t) = x_3(t-1) + q_3$$

$$x_4(t) = x_4(t-1) + q_4$$

where $x_1$, $x_2$ represents the positions of the object, and $x_3$, $x_4$ corresponds to the velocity state of each position component, and $\Delta T$, corresponds to delta time changes between the state update.

In matrix form, this can be expressed as:

$$\boldsymbol{x}(t) = \begin{bmatrix} 1 & 0 & \Delta T & 0 \\ 0 & 1 & 0 & \Delta T \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \boldsymbol{x}(t-1) + \boldsymbol{Q}, \tag{12}$$

$$\boldsymbol{y}(t) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \boldsymbol{x}(t) + \boldsymbol{R}$$

where $\boldsymbol{Q}$, is the process noise matrix, and $\boldsymbol{R}$, is the measurement noise matrix. Kalman Filtering can be used to predict and update the state estimates and its covariance estimate $\boldsymbol{P}$ at each time step.

Prediction Steps:

$$\hat{\boldsymbol{x}}(k|k-1) = \boldsymbol{A}\,\hat{\boldsymbol{x}}(k-1|k-1) \tag{13}$$

$$\boldsymbol{P}\,(k|k\text{-}1) = \boldsymbol{A}\,\boldsymbol{P}(k-1|k-1)\boldsymbol{A}^{\boldsymbol{T}} + \mathbf{Q}$$

Update Steps:

$$\boldsymbol{K}(k) = \boldsymbol{P}(k|k-1)\boldsymbol{H}^{\boldsymbol{T}}(\boldsymbol{H}\boldsymbol{P}(k|k-1)\boldsymbol{H}^{\boldsymbol{T}} + \boldsymbol{R})^{-\mathbf{1}} \tag{14}$$

$$\hat{\boldsymbol{x}}(k|k) = \hat{\boldsymbol{x}}(k|k-1) + \boldsymbol{K}\,(\boldsymbol{y}(k) - \boldsymbol{H})\hat{\boldsymbol{x}}(k|k-1)$$

$$\boldsymbol{P}(k|k) = (\boldsymbol{I} - \boldsymbol{K}(k)\boldsymbol{H})\boldsymbol{P}(k|k-1)$$

As the target(s) are being tracked, the state vectors $\hat{\boldsymbol{x}}$ associated with covariance $\boldsymbol{P}$ (motion constraint) are fed back to the CVMF process to fine tune the pre-defined velocity bins and improve the accuracy of matching. Having feedback from a tracker to CVMF also adds robustness to maintain the tracking of moving objects in a temporary occlusion (e.g. a car temporarily obscured by a tree). The tracker's state is capable of propagating to the next time step, assuming the target is traveling in a similar speed without the need to re-initialize VMF filters. Different applications might require a more sophiscated modeling of dynamic behavior such as the target's acceleration [75].

## 2.5 Results and Discussion

We compared our method with our baseline processing method as depicted in [Figure 25]. The baseline processing workflow is the same as the one in [Figure 20] but without the

additional CVMF component. The same video was used as an input to both processing methods. Both methods were evaluated over the same detection areas in the same regions of the image. Valid detections in the frames were manually labeled to provide assessment for the probability of detection and false alarm measures.

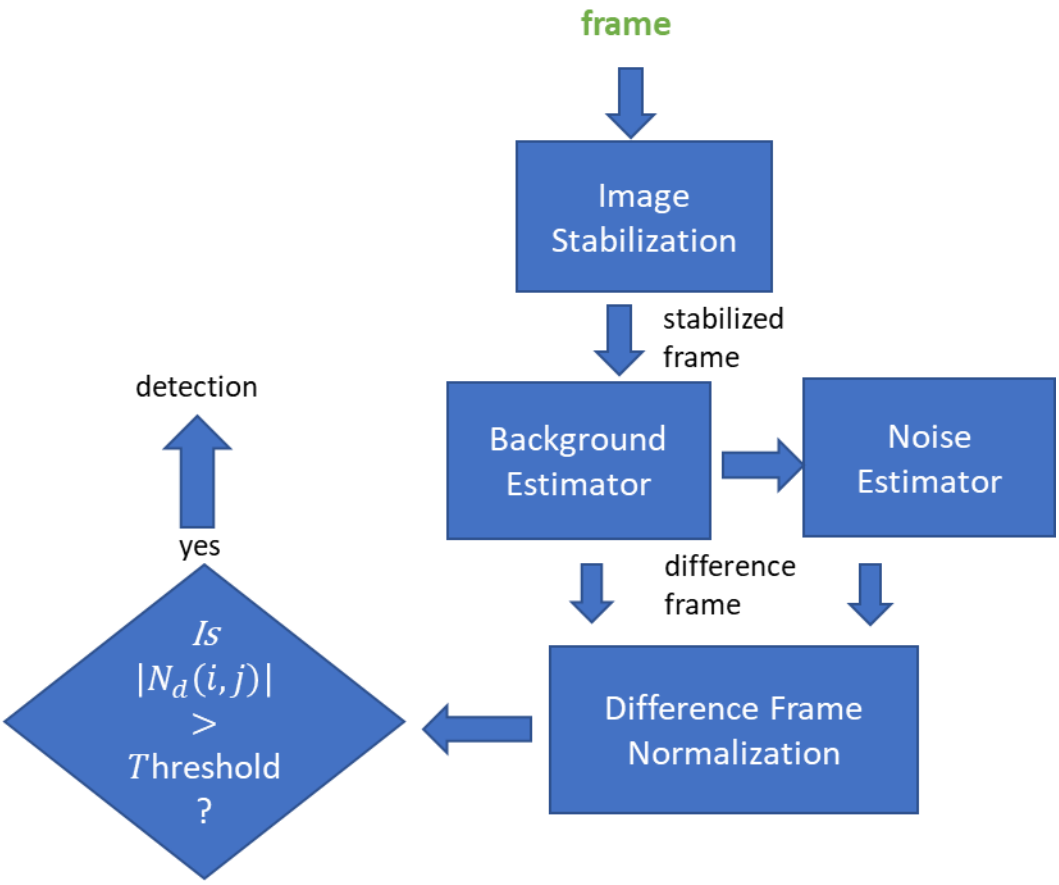



**Figure 25: Baseline Detection Processing**

[Figure 26] shows a comparison of the Receiver Operating Characteristics (ROC) data curve as a baseline, along with different window of frames used in the CVMF calculation. The ROC curve improves as the window of frames increases; however, it reaches an asymptotic state at 7 frames. A more sophisticated motion model is probably needed to integrate the target's motion over a longer framed window. An example of a qualitative comparison is

depicted in [Figure 27]. The baseline Normalized Difference Frame shows a very low SNR target at around 4.0. After incorporating the CVMF enhancement, the target's SNR increased to 8.0.

To support our claim of performing VMF operations on Normalized Difference Frames instead of operating on Difference Frames, [Figure 28] shows the ROC curve comparison of a baseline single Difference Frame thresholding versus using the CVMF method operating on Difference Frames. An example of a qualitative comparison is depicted in [Figure 29]. Though CVMF is also effective in boosting SNR on the Difference Frame domain, it does not perform as well on the Normalized Difference Frame. A direct ROC curve comparison of CVMF operating on Difference Frame versus CVMF operating Normalized Difference Frame is shown in [Figure 30]. As shown, operation on Normalized Difference Frames significantly outperforms operation on Difference Frames.

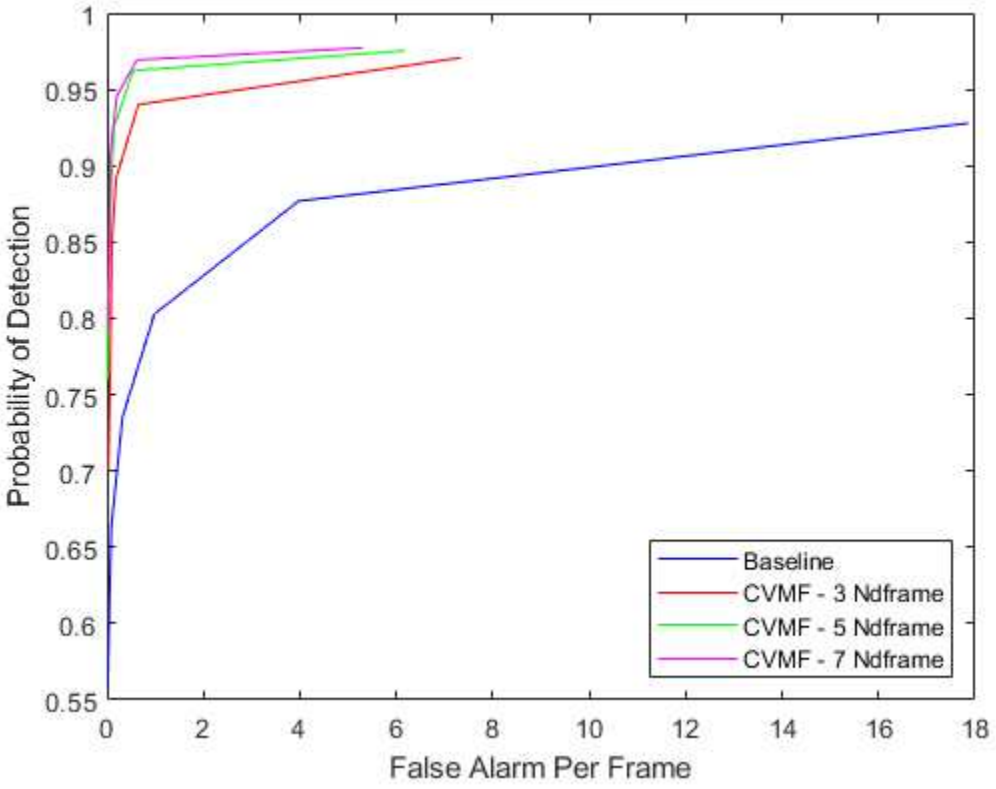



**Figure 26: ROC Curves Comparison - Normalized Difference Frame**

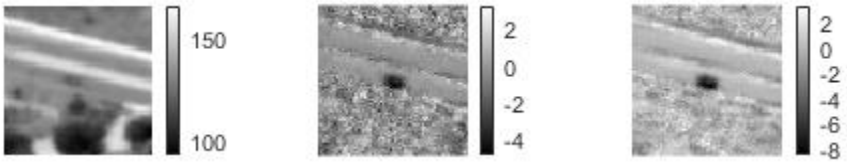



**Figure 27: Target Enhancement (Left –original, center- baseline normalized difference, Right – CVMF 5-frame Z-scores)**

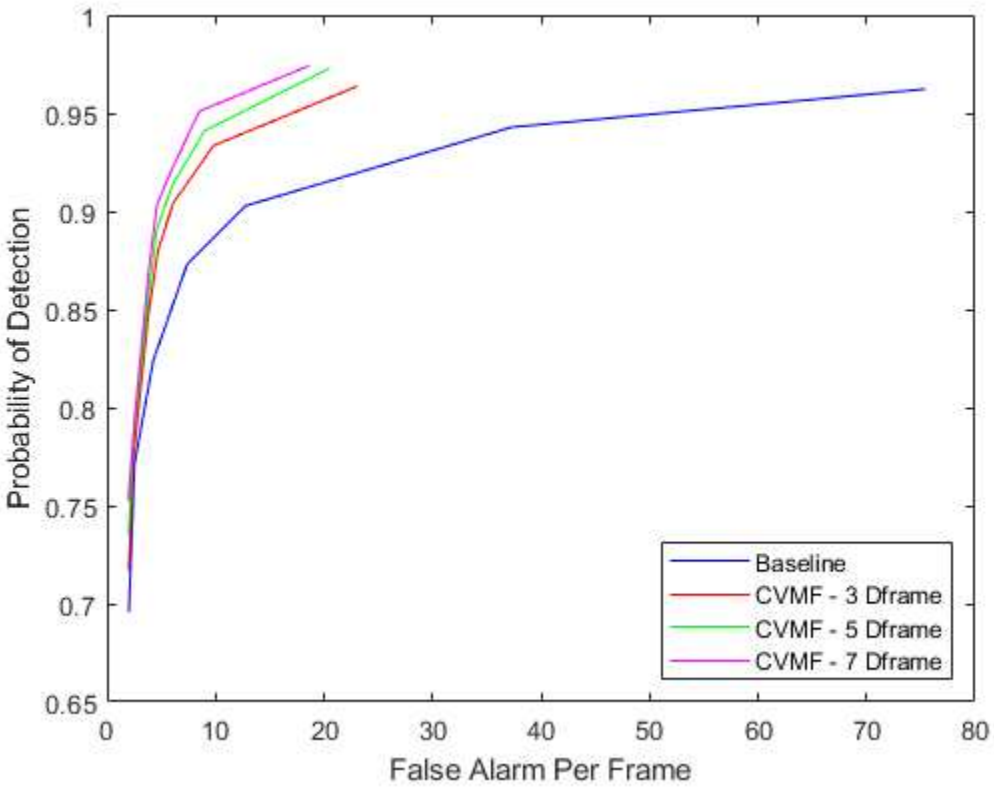



**Figure 28: ROC Curve Comparison - Difference Frame**

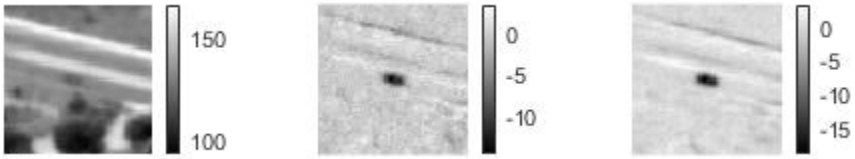



**Figure 29: Target Enhancement (Left –Original Frame, Center - Baseline difference, Right – CVMF 5-frame difference)**

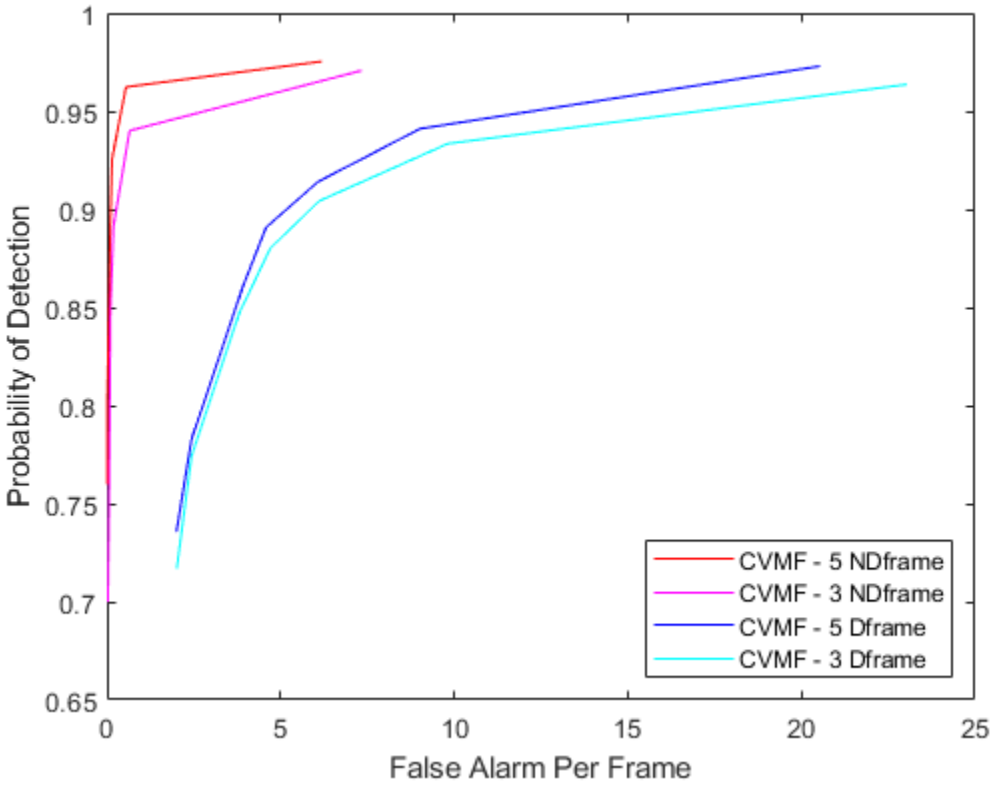



**Figure 30: CVMF Normalized Difference vs CVMF Difference**

## 2.6 Conclusion

In this chapter, we have provided a new method to enhance detection of low SNR targets. The innovation of this work is evident by the issuance of a U.S. patent [76]. Our CVMF method incorporates physical constraints from known roads and dynamic motion constraints obtained from a Kalman Tracker to accurately find, match, and integrate target signals over multiple frames to improve the target's SNR. We have demonstrated our results using real data collected by an actual sensor. Our method has established a significant improvement over baseline traditional detection techniques. In addition, we have proven that our technique can achieve better performance if the CVMF operations were performed under the Normalized

Difference Frames. Currently, our CVMF performance converges to a steady state at 7 frames. In the future, we plan to incorporate a more sophisticated motion model (e.g. constant acceleration model) to support a longer frame integration window. A more sophisticated motion model can potentially provide a more accurate estimation of the target's motion over a longer period of time.

# Chapter 3: Performance Study of Distance-Weighting Approach with Loopy Sum-Product Algorithm for Multi-Object Tracking in Clutter[3]

## 3.1 Introduction

Since probabilistic data association (PDA) was introduced for tracking in a cluttered environment [77], engineers have been trying to optimize the data association (DA) technique for implementing in a Kalman filter (KF) [78] tracker. With the inception of the sum-product algorithm (SPA) [79], a new graphical approach for KF tracking was developed that is more efficient in dealing with the complex problem of multiple target tracking (MTT). To develop a robust algorithm that is also scalable for tracking multiple objects in clutter, in this paper we examine the performance of the distance-weighting probabilistic data association (DWPDA) [80] in conjunction with the loopy sum-product algorithm (LSPA) [81]. The problem of DA is finding the correspondence between the targets and the measurements of uncertain origins. There are several approaches to tackle this problem, such as the nearest neighbor (NN), PDA, and joint probabilistic data association (JPDA) [77,82,83]. The NN approach is one of the easiest and uses at any time only the nearest measurement to the predicted measurement as if it were the one originated from the target of interest [82]. The NN approach is suboptimal and can only work well in case of widely spaced targets, accurate measurements, and few false alarms. Finding some association between all the targets and measurements is computationally expensive. Therefore, a gate is formed around the predicted measurement based on some predefined threshold called the validation region. The PDA algorithm obtains the probability of each measurement lying inside the validation region as being the correct measurement and

[3] The material from this chapter was published in [100].

updates the state estimates according to an appropriately modified tracking filter, called PDAF [77]. One major drawback of the PDA algorithm is that it treats every measurement inside the validation region of a target of interest as if it were originated from the said target or as a Poisson-distributed false alarm. PDA does not take into account the possibility that a validated measurement for one target might be a measurement originated from another nearby target. The JPDA algorithm improves this deficiency of the PDA algorithm by computing the association probabilities for the validated measurements from the joint likelihood functions corresponding to all feasible joint events such that no more than one measurement originates from each target [83]. However, with an increasing number of targets and/or clutter, the JPDA algorithm becomes impractical for real-time applications due to its combinatorial complexity because it considers all feasible joint events of measurements to targets to obtain the joint association probabilities [84].

PDA and JPDA are zero-scan algorithms, meaning that all hypotheses are combined after computation of the probabilities, for each target at each time step [77,83]. Alternatively, multiple hypothesis tracking (MHT) is a deferred decision logic algorithm. In the case of a conflicting target-measurement association, the MHT algorithm formulates alternative data association hypotheses instead of choosing the best-combined hypothesis. The ambiguities between the alternative hypotheses are resolved using the measurements that arrive later into the future [85]. JPDA is shown to produce reasonable results compared to the computationally expensive multi-scan MHT algorithm [83–96]. Both the PDA and the JPDA algorithms utilize known targets for forming validation gates in the measurement space to compute the posterior probabilities and are therefore categorized as target-oriented approaches. There exist

measurement-oriented algorithms, such as the one described in [87] and others, where hypotheses are formed for each measurement to have originated from a known target, a new target, or clutter. Both sets of algorithms have shown to yield an equivalent expression for posterior probabilities with appropriate assumptions [83,87,88]. As we realize that calculating DA probabilities for MTT is a process that involves a complicated global function of many variables that can be broken down into a product of local functions each of which depends on a subset of the variables, we look for alternate methods that can exploit this trait. SPA is one such method that operates in a factor graph [89] and attempts to compute, either exactly or approximately, various marginal functions associated with the global function [79]. SPA operates by passing messages, called beliefs, between the nodes of a factor graph, i.e., variables and local functions, that involve summations and products of factors. Implementation of a factor graph for the MTT DA problem requires a loopy sum-product solution which is neither guaranteed to converge nor produce the correct marginal functions if convergence occurs. A simultaneous target- oriented and measurement-oriented factor graph formulation of the DA problem has been shown that is guaranteed to converge and results in accurate association probabilities [81,90]. A simplified implementation of LSPA results in a significant reduction in computational complexity without loss of accuracy [90,91], which makes LSPA more appealing than PDA or JPDA for DA.

In recent years, the use of LSPA for DA in MTT has been gaining traction. A belief- propagation approach to multi-target multi-sensor tracking is proposed in [92] by formulating a detailed factor graph where every single target and data-association variable is modeled as an individual node. To tackle the problem of tracking an unknown number of targets, where targets appear

and disappear, an LSPA-based method is proposed in [93] that creates augmented target states for keeping track of existing and non-existing tar- gets. A more comprehensive derivation of the message-passing algorithm for multi-sensor multi-target tracking is given in [94,95]. An extension of the LSPA-based MTT framework for target estimation is given in [96] that exploits additional target information provided by a classifier. All these efforts highlight the potential of SPA for solving the complicated problem of DA in MTT.

The data association probabilities obtained using PDA are a crucial part of the beliefs being passed between nodes during the convergence of LSPA. Modifying the DA probabilities according to a weighting scheme based on distances between the predicted and validated measurements has been shown to enhance the tracking accuracy of PDAF while tracking a single target in a densely cluttered environment [80]. In this paper, we want to explore whether this distance-weighting approach for PDA, when integrated with LSPA, would improve the performance of LSPA even further. To evaluate this possibility, we formulate a distance weighting LSPA (DWLSPA) and compare its performance in terms of tracking accuracy and computation time against DWPDA, JPDA, and LSPA for tracking multiple targets crossing at a small angle in different density clutter. Our results turn out to be contrary to expectations.

The main contribution of this performance study is to explore the idea of modifying one of the building blocks for a state-of-the-art data-association algorithm [91] for multi- target tracking and to compare the tracking accuracy of the modified algorithm to that of the original algorithm. The distance-weighting modification analyzed in this paper is based on a recent successful implementation of a similar modification to one of the earliest data-association frameworks for tracking targets in cluttered environments [90]. In doing so, we develop the

mathematical formulation for each data-association filter being considered and evaluate its performance in terms of tracking accuracy and computation time over a wide range of easy-to-replicate multi-object-tracking scenarios.

We introduce our notations and the target tracking system in Section 3.2. In Section 3.3, we describe the problem of PDA, DWPDA, JPDA, and LSPA. We present simulation results for these methods and our analysis in Section 3.4. In Section 3.5, we conclude the paper with our observations and final remarks regarding the performance of these methods.

## 3.2 Target Tracking Dynamic System Model and Assumptions

We describe the classic data association problem in which a single sensor surveils a large number of targets. The number of targets under surveillance is assumed to be known and is denoted by $N_T$. The measurements are comprised of possible target detections and false alarms. A target is detected with a known probability of detection $P_D$ and is independent of time. The false alarms, modeled according to the Poisson point process with a known spatial density $\lambda$, are uniformly distributed in the measurement space. A validation region, with the threshold $\gamma$ corresponding to certain gate probability $P_G$, is set up at every sampling time around the predicted measurement and possibly several measurements fall in it. Each algorithm differs in how these measurements are used (or not) in the estimation of the state of the target. We assume that each target can generate at most one measurement and each measurement can have only one source.

We denote by $x_i(k)$, $i \in \{1, \ldots, N_T\}$, the state of $i$-th target of dimension $n_x$ at time $k$. The complete set of target states at time $k$ is denoted by $X(k) = (x_1(k), \ldots, x_N T(k))$. At time $k$, the

total number of measurements is denoted by $M(k)$. We denote by $z_j(k)$, $j \in \{1, \ldots, M(k)\}$, the value of $j$-th measurement of dimension $n_z$. The complete set of measurements at time $k$ is denoted by $Z(k) = (z_1(k), \ldots, z_{M(k)}(k))$, and the complete set

of measurements up to and including time $k$ is denoted by $Z^k = (Z(1), \ldots, Z(k))$.

The state and measurement equations are assumed linear with additive zero-mean

white noise with known covariances. The state of target $t$ evolves in time according to the equation

$$x_t(k+1) = F(k)x_t(k) + G(k)u(k), \tag{1}$$

and the true measurement for target $t$ is given by

$$z_t(k) = H(k)x_t(k) + w(k) \tag{2}$$

where $u(k)$ and $w(k)$ are zero-mean mutually independent white Gaussian noise sequences with known covariances $Q(k)$ and $R(k)$, respectively. Functions $F(k)$, $G(k)$, and $H(k)$ are known matrices for state transition, noise gain, and sensor, respectively. The past information (through time $k$-1) about the target $t$ is assumed to be known and summarized approximately by the Gaussian posterior

$$p[x_t(k-1)|Z^{k-1}] = \mathcal{N}[x_t(k-1); \hat{x}_t(k-1|k-1), P_t(k-1|k-1)] \tag{3}$$

where $\hat{x}_t(k-1|k-1)$ and $P_t(k-1|k-1)$ are the state estimate and covariance for target

## 3.3 Algorithm Description

### *3.3.1 Probabilistic Data Association Filter*

The PDA algorithm calculates the association probabilities for each validated measurement at the current time for the target of interest. PDA assumes that all the validated measurements are generated by either the target of interest or clutter. The association prob- abilities are used for calculating the mean squared error (MSE) estimate and covariance of the target's state. An appropriately modified KF, called PDAF, is used to account for the uncertainty of origins for the validated measurements while estimating the state of the target. The algorithm can be given as follows.

#### 3.3.1.1 Prediction

The prediction of the state and measurement of target *t* at time *k* is done as in the KF, i.e.,

$$\hat{x}_t(k|k-1) = F(k-1)\hat{x}_t(k-1|k-1) \quad (4)$$

$$\hat{z}_t(k|k-1) = H(k)\hat{x}_t(k|k-1). \quad (5)$$

The covariance of the predicted state for target *t* is

$$P_t(k|k-1) = F(k-1)P_t(k-1|k-1)F(k-1)' + G(k-1)Q(k-1)G(k-1)'. \quad (6)$$

Here, $\widehat{x_t}$(*k*-1 | *k*-1) and $P_t$(*k*-1|*k*-1) are available from Equation (3). The innovation covariance of the target *t* (for the correct measurement) is

$$S_t(k) = H(k)P_t(k|k-1)H(k)' + R(k). \quad (7)$$

3.3.1.2 Measurement Validation

The validation region for target *t* at time *k* is the elliptical region given by

$$\mathcal{V}_t(k,\gamma) = \{z \in Z(k) : [z - \hat{z}_t(k|k-1)]' S_t(k)^{-1} [z - \hat{z}_t(k|k-1)] \leq \gamma\}. \quad (8)$$

Here, *γ* can be obtained according to

$$V(k) = c_{n_z} |\gamma S(k)|^{1/2} \quad (9)$$

where $V(k)$ is the volume of the validation region given by Equation (8) and $c_{n_z}$ is the volume of the unit hypersphere of dimension $n_z$ [88]. The validated measurements for target *t* according to Equation (8) are

$$Z_t(k) \stackrel{\Delta}{=} \{z_i(k)\}_{i=1}^{m_t(k)} \quad (10)$$

where $m_t(k)$ is the number of validated measurements for target $t$ at time $k$.

3.3.1.3 Data Association Probabilities

The association probability $\beta_i^t$ for each validated measurement of target *t* is obtained as

$$\beta_i^t(k) = \begin{cases} \frac{\mathcal{L}_i(k)}{1 - P_D P_G + \sum_{j=1}^{m_t(k)} \mathcal{L}_j(k)} & i = 1, \ldots, m_t(k) \\ \frac{1 - P_D P_G}{1 - P_D P_G + \sum_{j=1}^{m_t(k)} \mathcal{L}_j(k)} & i = 0 \end{cases} \quad (11)$$

where *i* = 0 indicates probability of associating none of the validated measurements to the target. In Equation (11), $L_i(k)$ the likelihood ratio (LR) of measurement $z_i(k)$ originating from the target *t* vs. from clutter and is obtained as:

$$\mathcal{L}_i(k) \triangleq \frac{\mathcal{N}[z_i(k); \hat{z}_t(k|k-1), S_t(k)]P_D}{\lambda}, \quad z_i(k) \in Z_t(k). \tag{12}$$

3.3.1.4 State Estimation

The state estimation for target t is done according to PDAF by

$$\hat{x}_t(k|k) = \hat{x}_t(k|k-1) + W_t(k)v_t(k) \tag{13}$$

where the combined innovation for target *t* is

$$v_t(k) = \sum_{i=0}^{m_t(k)} \beta_i^t(k)v_i^t(k), \tag{14}$$

and

$$v_i^t(k) = z_i(k) - \hat{z}_t(k|k-1) \tag{15}$$

is the innovation of measurement for $z_i(k) \in Z_t(k)$. The filter gain is calculated as

$$W_t(k) = P_t(k|k-1)H(k)'S_t(k)^{-1}. \tag{16}$$

The covariance estimation for target *t* associated with the updated state is

$$P_t(k|k) = \beta_0(k)P_t(k|k-1) + [1 - \beta_0^t(k)]P_t^c(k|k) + \tilde{P}_t(k) \tag{17}$$

where the covariance of the state updated with the correct measurement, $P^c$, and the innovation spread, $P$, for target *t* are given by

$$P_t^c(k|k) = P_t(k|k-1) - W_t(k)S_t(k)W_t(k)' \tag{18}$$

and

$$\tilde{P}_t(k) = W_t(k)\left[\sum_{i=1}^{m_t(k)} \beta_i^t(k) v_i^t(k) v_i^t(k)' - v_t(k) v_t(k)'\right] W_t(k)', \tag{19}$$

respectively.

The estimated state in PDAF from Equation (13) is for a single target $x_t$. To estimate the state of every target $x_t$, $t \in \{1, \dots, N_T\}$, we need to apply PDAF to the targets one by one sequentially. This means that for each target $x_t$, $t \in \{1, \dots, N_T\}$, we perform state and measurement prediction, form a validation region around the predicted measurement and prune off potentially unrelated measurements, obtain data-association probabilities for the validated measurements, and, finally, update the state estimate according to Equation (13). The order in which PDAF is applied to the targets is irrelevant because the outcome is independent of the order.

*3.3.2 Distance-Weighting Probabilistic Data Association Filter*

In the PDA algorithm, the association probability $\beta$ is calculated as the likelihood ratio of a validated measurement to have originated from a target vs. from clutter. Chen et al. [80] have pointed out that a true measurement from a target of interest is more likely to be nearthe target's predicted measurement. Since false alarms are uniformly distributed in the measurement space, the measurement nearest to that of the predicted measurement from a target of interest should carry more weight while calculating the association probabilitiesfor the target. The distance weight proposed in [4] is calculated as

$$\Delta_i^t(k) = \frac{1/\delta_i^t(k)}{\sum_{i=1}^{m_t(k)} 1/\delta_i^t(k)} \tag{20}$$

where $\delta_i^t(k)$ is the Mahalanobis distance between validated measurement *i* and predicted measurement of target *t* at time *k*. The Mahalanobis distance is calculated as the norm of the innovation squared and is given by

$$\delta_i^t(k) = v_i^t(k)' S_t(k)^{-1} v_i^t(k) \tag{21}$$

where $v_i^t(k)$ is the innovation as defined in Equation (15).

The new data association probabilities for target t at time k are calculated as

$$\mathcal{B}_i^t(k) = \beta_i^t(k)\Delta_i^t(k) \quad i = 1, \ldots, m_t(k) \tag{22}$$

and are normalized according to

$$\mathcal{B}_i^t(k) = \mathcal{B}_i^t(k) / \sum_{j=0}^{m_t(k)} \mathcal{B}_i^t(k) \quad i = 0, 1, \ldots, m_t(k). \tag{23}$$

The data association probabilities obtained in Equation (23) are used for estimating target state as explained previously in PDAF Section 3.3.1. Similar to PDAF, the DWPDA filter is designed for tracking a single target. For the purpose of tracking multiple targets, we need to update the states of the targets one by one.

*3.3.3 Joint Probabilistic Data Association Filter*

The PDA algorithm is designed for tracking a single target in clutter. Because the PDA algorithm assumes all the incorrect measurements in the validation region of a targetof interest are clutter, it is susceptible to scenarios where these incorrect measurements might have originated from another nearby target. Situations may arise in MTT where thevalidation regions of nearby targets

may overlap for several time frames in a row and cause persistent interference that can lead to track deterioration or track loss altogether with PDA. JPDA improves on PDA by calculating joint association probabilities at each time step. Marginalized association probabilities for each target can be obtained using these joint association probabilities, which are then used for estimating the state of each target.

3.3.3.1 Measurement Validation

A major difference between the PDA and the JPDA algorithm is that no individual validation gates will be assumed for the various targets. A uniform validation region for all targets is obtained by taking a union of individual validation gates. This way each measurement is assumed to be validated for each target and false alarms will be assumed to be uniformly distributed across the entire validation region. Predictions for each target are done at each time step similar to the PDA algorithm using Equations (4)–(7).

Once the validated measurements are obtained for each target using Equations (8) and (10), the combined validated measurements at time k are given according to

$$Z_{N_T}(k) = \bigcup_{i=1}^{N_T} Z_i(k). \tag{24}$$

The number of combined validated measurements is $m_{N_T}(k)$ at time $k$.

3.3.3.2 The Validation Matrix

A validation matrix Ω of size $m_{N_T} \times (N_T + 1)$ with binary elements is created using the combined validated measurement.

$$\Omega(k) = [\omega_{jt}] \quad j = 1, \ldots, m_{N_T}(k); \quad t = 0, 1, \ldots, N_T \tag{25}$$

where

$$\omega_{jt} = \begin{cases} 1 & \text{if } z_j(k) \in Z_t(k) \\ 0 & \text{otherwise.} \end{cases} \tag{26}$$

The first column of Ω(*k*) corresponding to $t$ = 0 is all unity indicating that eachmeasurement could be a false alarm.

3.3.3.3 The Feasible Joint Events

The joint association events for time k are given by

$$\theta(k) = \bigcap_{j=1}^{m_{N_T}} \theta_{jt_j} \quad j = 1, \ldots, m_{N_T}(k); \quad t = 0, 1, \ldots, N_T \tag{27}$$

where in the event of $\theta_{jt}$ measurement $j$ is originated from target $t_j$. An event matrix $\hat{\Omega}$ consisting of the units in Ω corresponding to the association in $\vartheta$ is used to represent a joint association event $\vartheta$.

$$\hat{\Omega}(\theta(k)) = [\hat{w}_{jt}(\theta(k))] \quad j = 1, \ldots, m_{N_T}(k); \quad t = 0, 1, \ldots, N_T \tag{28}$$

Where'

$$\hat{\omega}_{jt}(\theta(k)) = \begin{cases} 1 & \text{if } \theta_{jt} \in \theta(k) \\ 0 & \text{otherwise.} \end{cases} \tag{29}$$

Not all joint association events are feasible joint events. As per our target tracking model assumptions, feasible association events are only those where no more than one measurement is associated with each target. Therefore, a feasible association event is the one which satisfies the following two conditions:

1. A measurement can have only one source, i.e.,

$$\sum_{t=0}^{N_T} \hat{\omega}_{jt}(\theta(k)) = 1 \quad \forall j. \tag{30}$$

2. Each target can generate at most one measurement, i.e.,

$$\delta_t(\theta(k)) \triangleq \sum_{j=1}^{m_{N_T}} \hat{\omega}_{jt}(\theta(k)) \leq 1 \quad t = 1, \ldots, N_T. \tag{31}$$

$\delta_t(\vartheta(k))$ in Equation (31), called the target detection indicator, indicates if a measurement is associated with target $t$ in event $\vartheta(k)$. Similarly, we define the measurement association indicator $\tau(\vartheta(k))$ to indicate if measurement $j$ is associated with a target in event $\vartheta(k)$.

$$\tau_j(\theta(k)) \triangleq \sum_{t=1}^{N_T} \hat{\omega}_{jt}(\theta(k)) \quad j = 1, \ldots, m_{N_T}(k). \tag{32}$$

We can obtain the number of false alarms in event $\vartheta(k)$ as

$$\phi(\theta(k)) = \sum_{j=1}^{m_{N_T}(k)} [1 - \tau_j(\theta(k))]. \tag{33}$$

3.3.3.4 Joint Data Association Probabilities

The joint data association probabilities are calculated as

$$P\{\theta(k)|Z^k\} = \frac{1}{c}[Z_{m_{N_T}(k)}(k)|\theta(k), m_{N_T}(k), Z^{k-1}]P\{\theta(k)|m_{N_T}(k)\} \tag{34}$$

where P indicates the probability mass function (PMF) and c is the normalization constant. The marginal association probabilities are obtained from the joint DA probabilities by summing over all the joint events according to Equation (29). The marginal association probabilities at time k are then given as

$$\beta_j^t(k) = \sum_{\theta(k)} P\{\theta(k)|Z^k\}\hat{\omega}_{jt}(\theta(k)) \quad j = 1, \ldots, m_{N_T}(k); \quad t = 0, 1, \ldots, N_T \tag{35}$$

$$\beta_0^t(k) = 1 - \sum_{j=1}^{m_{N_T}(k)} \beta_j^t(k) \quad t = 0, 1, \ldots, N_T \tag{36}$$

where the probability ${\beta_j}^t$ represents that the measurement j is associated with target t and t indicates that none of the validated measurements is associated with target t. The

expression for the joint association probabilities in Equation (35) can be given in terms of the variables defined in JPDAF Section 3.3.3 as

$$P\{\theta(k)|Z^k\} = \frac{1}{c_1} \prod_{j=1}^{m_{N_T}(k)} \{\lambda^{-1}[\Lambda_j^{t_j}(k)]\}^{\tau_j(\theta(k))} \prod_{t=1}^{N_T} (P_D)^{\delta_t(\theta(k))}(1 - P_D)^{1-\delta_t(\theta(k))} \tag{37}$$

where ${\Lambda_j}^{t_j}$ is the Gaussian density of measurement j associated with target of index $t_j$ given by

$$\Lambda_j^{t_j}(k) = \mathcal{N}[z_j(k); \hat{z}_{t_j}(k|k-1), S_{t_j}(k)] \tag{38}$$

and $c_1$ is the new normalization constant. The marginal association probabilities obtained in Equations (35) and (36) are used in conjunction with the state estimation equations given in PDAF Section 3.3.1.4 for estimating state of each target separately.

### 3.3.4 Loopy Sum-Product Algorithm

Graphical models can be used for representing the joint probability distributions of many variables efficiently by exploiting factorization. For SPA, factor graphs are utilized for representing the DA relations between multiple targets and their validated measurements. SPA is then conducted on the resulting loopy factor graph for obtaining the marginal DA probabilities. The algorithm for the loopy-SPA can be given as follows.

#### 3.3.4.1 Belief Propagation in Factor Graphs

Kschischang et al. [79] demonstrated how factor graphs can be used to interpret a variety of algorithms such as the KF, the Viterbi algorithm, the Hidden Markov Model, etc. A factor graph is a standard bipartite graphical representation of a mathematical relationship between random variables and local functions. While formulating a factor graph to express the structure of the factorization of a global function of many variables into several local functions, each node represents each random variable $n \in \aleph$, each factor represents each local function $f \in \Im$, and an edge connects a node $n$ to a factor $f$ if and only if $n$ is an argument of $f$.

The SPA algorithm, also called as Belief Propagation (BP), passes messages, called beliefs, between nodes and factors in an iterative manner for conducting optimal inference on a tree-

structured factor graph. We denote by ψ·(·) the joint probability distribution function, by $\mu_{n\to f}(x_n)$ the message sent from node $n \in \eta_f$ to factor $f$, by $\mu_{n\to f}(x_n)$ the message sent from factor $f$ to node $n \in \eta_f$, by $\eta_f \subseteq \aleph$ the set of neighboring nodes of $f$, and by $\eta_n = \{f \in \Im \mid n \in \eta_f\}$ the factors involving node $n$. The message computation performed by SPA can be given as

$$\mu_{n\to f}(x_n) = \prod_{\xi\in\eta_n\setminus\{f\}} \mu_{\xi\to n}(x_n) \tag{39}$$

$$\mu_{f\to n}(x_n) = \sum_{\sim\{x_n\}} \left( \psi_f(x_{\eta_f}) \prod_{\xi\in\eta_f\setminus\{n\}} \mu_{\xi\to f}(x_\xi) \right) \tag{40}$$

where $x_n$ denotes the summation over all arguments of $f$ except $x_n$. For factorization involving continuous variables, the summation is replaced with an integral taken over a Lebesgue measure. The algorithm is known as sum-product because the steps involved are summations (or integrals) and products of factors and messages.

SPA extended to loopy graphs is called LSPA. LSPA simply requires repeated ap- plication of SPA until convergence occurs. Practically, this means computing messages continuously using Equations (39) and (40) until the maximum error between subsequent messages is less than a pre-set threshold. However, LSPA is neither guaranteed to converge to the right answer nor to converge at all.

3.3.4.2 Factor Graphs for Data Association

We consider the DA problem involving known and fixed $N_T$ targets and their combined validated measurements $ZN_T$ (k) at time k obtained as explained in JPDAF Section 3.3.3.1. The

number of combined validated measurements are $mN_T$ (k) at time k. Formulating a factor graph for the DA problem that is guaranteed to converge according to [81,90,91], we need the following two sets of association variables:

1. Target oriented association variable (a): Create an association variable $a_i(k) \in \{0, 1, \ldots, mN_T$ (k) for each target i 1, . . . , $N_T$ . The value assigned to $a_i(k)$ is an index to the measurement with which target $i$ is hypothesized to be associated at time k (zero if the target is hypothesized to not have been detected). The complete set of target oriented association variables at time k is denoted by $a$(k).

2. Measurement oriented association variable (b): Create an association variable $b_j$(k) ∈ 0, 1, . . . , $N_T$ for each measurement $j$ 1, . . . , $mN_T$ (k) . The value assigned to $b_j$(k) is an index to the target with which measurement $j$ is hypothesized to be associated

at time k (zero if the measurement is hypothesized to be clutter). The complete set of measurement oriented association variables at time k is denoted by $b$(k).

Given one set of association variables, the other set can be perfectly reconstructed. As shown in [81,90,91], this use of seemingly redundant information while forming a factor graph leads to the remarkable result of guaranteed convergence of LSPA computed on the said factor graph. A bipartite graphical model formed using the association variables $a$(k) an $b$(k) is shown in [Figure 31].

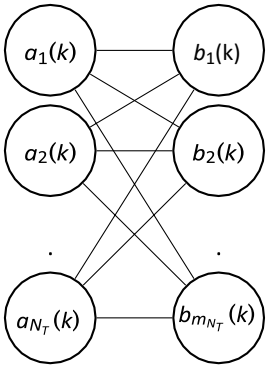



**Figure 31: Bipartite graphical model formulation for data association at time k. The value assigned to $a_i(k)$ is an index to the measurement with which target i is hypothesized to be associated at time k and the value assigned to $b_j$(k) is an index to the target with which measurement j is hypothesized to be associated at time k.**

$$f(X(k), a(k), b(k)) \propto \prod_{i=1}^{N_T} \psi_i(x_i(k), a_i(k)) \prod_{j=1}^{m_{N_T}(k)} \psi_{i,j}(a_i(k), b_j(k)) \tag{41}$$

where

$$2\psi_i(x_i(k), a_i(k)) = \begin{cases} [1 - P_D(x_i(k)) f(x_i(k)|Z(k-1))] & \text{if } a_i(k) = 0 \\ [P_D(x_i(k)) f(z_{a_i(k)}|x_i(k)) f(x_i(k)|Z(k-1))]/\lambda(z_{a_i(k)}) & \text{if } a_i(k) \neq 0 \end{cases} \tag{42}$$

and

$$\psi_{i,j}(a_i(k), b_j(k)) = \begin{cases} 0 & \text{if } a_i(k) = j,\ b_j(k) \neq i \text{ or } b_j(k) = i,\ a_i(k) \neq j \\ 1 & \text{otherwise.} \end{cases} \tag{43}$$

Here, $\propto$ indicates equality up to a normalization factor, $\psi_i(x_i(k), a_i(k))$ indicates the dependence of the factors $\psi_i(\cdot)$ on Z(k), $\psi_{i,j}$ $(a_i(k), b_j$(k)) enforces consistency of the redundant association variables $a_i(k)$ and $b_j$(k) describing the same association configuration,

$P_D(x_i)$ gives the probability of detection for target $x_i$, and $\lambda(z_j)$ gives the PDF value of measurement zj occurring as a result of false alarm with a Poisson point process.

The factorization structure in Equation (41) can be represented by a factor graph. As an example, for $X(k) = (x_1(k), x_2(k))$ and $Z(k) = (z_1(k), z_2(k))$ , the factor-graph representation of Equation (41) is shown in Figure 2. Since the target states $x_i(k)$ in [Figure 32] are leaf nodes, we can marginalize these and replace the factor $\psi_i(x_i(k), a_i(k))$ with $\psi_i(a_i(k)$. In a factor graph, each parameter variable is represented by a variable node (circle node), and each factor is represented by a rectangle node, as shown in [Figure 32]. Each variable node and each factor node are connected by an edge if the variable is an argument of the factor. For each node, certain messages are calculated according to Equations (39) and (40). Message passing is started at variable nodes with only one edge (which pass a constant message) and/or factor nodes with only one edge (which pass the corresponding factor). Finally, for each variable node, a belief (posterior PDF value) is calculated as the product of all incoming messages (passed from all adjacent factor nodes) followed by a normalization. For a tree-structured factor graph, these beliefs are exactly equal to marginal posterior PDF values. For a loopy factor graph, the beliefs are in general only approximations of the respective marginal posterior PDF values. A detailed description of the messages passed between each variable node and factor node is given in [95]. A sample MATLAB program for implementing one iteration of loopy SPA is given in [90].

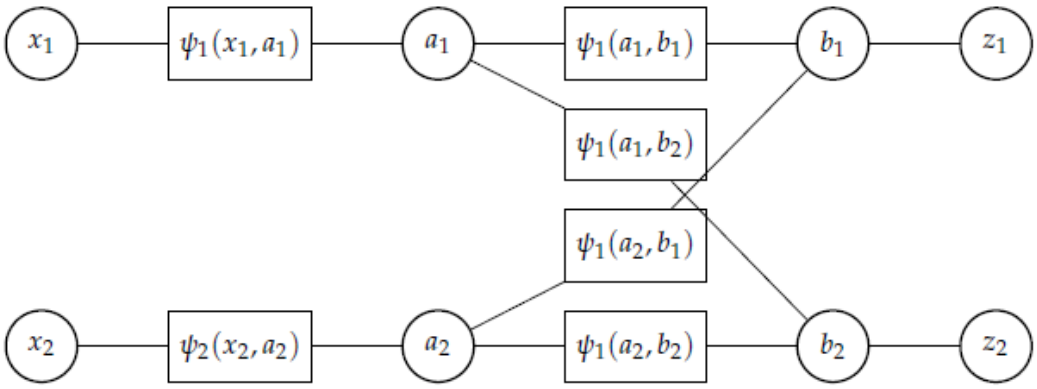



**Figure 32: Factor graph representing the factorization of the joint posterior probability density function (PDF) $\boldsymbol{f(x_1, x_2, a_1, a_2, b_1, b_2 \mid z_1, z_2)}$ according to Equation (41), depicted for one time step. For simplicity, the time index k is omitted.**

3.3.4.3 Joint Data Association Probabilities

Once the combined validated measurements are obtained as given in Equation (24), the joint data association probabilities under the stated assumptions at time k are calculated as

$$p(a(k), b(k)|Z^k) = \prod_{i=1}^{N_T} \psi_i(a_i(k)) \prod_{j=1}^{m_{N_T}(k)} \psi_{i,j}(a_i(k), b_j(k)) \tag{44}$$

where

$$\psi_{i,j}(a_i(k), b_j(k)) = \begin{cases} 0 & \text{if } a_i(k) = j,\ b_j(k) \neq i \text{ or } b_j(k) = i,\ a_i(k) \neq j \\ 1 & \text{otherwise} \end{cases} \tag{45}$$

enforce consistency of the redundant association variables $a_i(k)$ and $b_j$(k) describing the same association configuration, and $\psi_i$ ($a_i(k)$ = 0) = 1 and $\psi_i$ ($a_i(k)$ = $j$ > 0) are the (unnormalized) single target data association probabilities obtained as explained in PDAF Section 3.3.1.3.

Equation (44) can be expressed as the formulation of SPA for a bipartite model illustrated in Figure 1 in which all target association variables ai(k) are connected to all measurement association variables $b_j$(k). In this case, LSPA may be implemented

via two half iterations, alternating between the two sets of messages $\mu_{a_i(k)\to b_j(k)}((b_j(j))$ and $\mu_{b_j(j)\to a_i(k)}(a_i(k))$. The message updating equations according to Equations (39) and (40) can be given as

$$\mu_{a_i(k)\to b_j(k)}(b_j(k)) = \sum_{a_i(k)} \psi_i(a_i(k))\psi_{i,j}(a_i(k), b_j(k)) \prod_{j'\neq j} \mu_{b_{j'(k)}\to a_i(k)}(a_i(k)) \tag{46}$$

$$= \begin{cases} \psi_i(j) \prod_{j'\neq j} \mu_{b_{j'(k)}\to a_i(k)}(j) & \text{if } b_j(k) = i \\ \sum_{a_i(k)\neq j} \psi_i(a_i(k)) \prod_{j'\neq j} \mu_{b_{j'}(k)\to a_i(k)}(a_i(k)) & \text{if } b_j(k) \neq i \end{cases} \tag{47}$$

and

$$\mu_{b_j(k)\to a_i(k)}(a_i(k)) = \sum_{b_j(k)} \psi_{i,j}(a_i(k), b_j(k)) \prod_{i'\neq i} \mu_{a_{i'}(k)\to b_j(k)}(b_j(k)) \tag{48}$$

$$= \begin{cases} \prod_{i'\neq i} \mu_{a_{i'}\to j}(i) & \text{if } a_i(k) = j \\ \sum_{b_j(k)\neq i} \prod_{i'\neq i} \mu_{a_{i'}\to j}(b_j(k)) & \text{if } a_i(k) \neq j. \end{cases} \tag{49}$$

These messages can be further simplified as shown in [14,15]. Upon convergence of LSPA, the approximate marginal association probabilities can be given as

$$p(a_i(k) = j|Z^k) = \frac{\psi_i(j)\mu_{b_j(k)\to a_i(k)}(a_i(k))}{\sum_{j'=0}^{m_{N_T}(k)} \psi_i(j')\mu_{b_{j'}(k)\to a_i(k)}(a_i(k))} \tag{50}$$

$$p(b_j(k) = i|Z^k) = \frac{\mu_{a_i(k)\to b_j(k)}(b_j(k))}{\sum_{i'=0}^{N_T} \mu_{a_{i'}(k)\to b_j(k)}(b_j(k))} \tag{51}$$

where $\mu_{b_j(k)=0\to a_i(k)}(a_i(k)) \triangleq 1$ and $\mu_{a_i(k)=0\to b_j(k)}(b_j(k)) \triangleq 1$. The marginal association probabilities are used to estimate target states as per PDAF Section 3.3.1.4.

### 3.3.5 Distance-Weighting Loopy Sum-Product Algorithm

Since the factor-graph representation for the MTT DA problem contains loops, and the convergence process of calculating the marginal data-association probabilities using LSPA is governed by some heuristically determined preset threshold, different initiation messages can lead to different final beliefs. These initiation messages for the convergence process happen to be the (unnormalized) single-target data-association probabilities. These probabilities directly influence the marginal data-association probabilities at the end of the convergence process and, consequently, the estimation of the target state. We would expect that a more accurate initial set of single-target probabilities leads to either more accurate final beliefs, a faster convergence to the final beliefs, or both. The distance-weight- based association probabilities have been proven to be more accurate for tracking a single target in densely cluttered environments [80]. We would expect this adjustment to the initial condition to change the calculation of the association probabilities and hence the overall tracking process in terms of tracking accuracy or computation time, and therefore worth exploring.

Here we formulate a modification for the joint data association probabilities calculated using LSPA under the stated assumptions as described in Section 3.3.4. The distance-weight based joint data association probabilities at time k can be given as

$$p(a(k), b(k)|Z^k) = \prod_{i=1}^{N_T} \psi_i(a_i(k)) \prod_{j=1}^{m_{N_T}(k)} \psi_{i,j}(a_i(k), b_j(k)) \tag{52}$$

where

$$\psi_{i,j}(a_i(k), b_j(k)) = \begin{cases} 0 & a_i(k) = j,\ b_j(k) \neq i \text{ or } b_j(k) = i,\ a_i(k) \neq j \\ 1 & \text{otherwise} \end{cases} \tag{53}$$

enforce consistency of the redundant association variables $a_i(k)$ and $b_j(k)$ describing the same association configuration, and $\psi_i(a_i(k)$ = 0, and $\psi_i(a_i(k)$ = j > 0). $\beta^t{}_i(k)$ are the (unnormalized) distance-weight based single target data association probabilities obtained in Equation (22). The approximated marginal association probabilities are obtained from the joint DA probabilities as described in Section 3.3.4.

## 3.4 Simulation and Analysis

### 3.4.1 The Dynamic Model

We assume a two-dimensional system where the target state vector consists of position and velocity in each of the two coordinates. For target $i$ at time k, the target state $x_i$(k) = [x(k), $\dot{x}$(k), y(k), $\dot{y}$(k)] has four components: The first and second components are the horizontal location and velocity, respectively, while the third and fourth components are the vertical location and velocity, respectively. The system is equipped with the nearly constant velocity (NCV) model (also sometimes called the constant velocity (CV) model). The system is described by Equations (1) and (2) with

$$F(k) = \begin{bmatrix} 1 & T & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & T \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{54}$$

$$G(k) = \begin{bmatrix} T^2/2 & 0 \\ T & 0 \\ 0 & T^2/2 \\ 0 & T \end{bmatrix} \tag{55}$$

$$H(k) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \tag{56}$$

$$Q(k) = \begin{bmatrix} q & 0 \\ 0 & q \end{bmatrix} \tag{57}$$

$$R(k) = \begin{bmatrix} r & 0 \\ 0 & r \end{bmatrix} \tag{58}$$

where $T$ is the sampling interval.

### 3.4.2 Simulation Parameters

For our simulations, we set the probability of detection $P_D$ = 0.9, the gate threshold γ = 9.21 corresponding to gate probability $P_G$ = 0.99, sampling interval $T$ = 1 s, the process noise variance q = 0.05, and the measurement noise variance r = 5. To better evaluate the accuracy of different algorithms under multiple conditions, we varied the clutter density generated according a Poisson point process from $\lambda = 1.0 \times 10^{-4}$/scan/m2 to $\lambda = 5.0 \times 10^{-4}$//scan/m2. We also varied the number of tracked targets from 1 to 6. The initial state of the first target is always set at $x_1(1)$ = [100 m, 30 m/s, 100 m, 30 m/s]. We set the initial state of each consecutive target by $x_i$ (1) = [100 m, 30 m/s, $(100 - i \times 100 \times c(i))$ m, $(30 - i \times 30 \times c(i))$ m/s] where $i$ is the target index and $c(i)$ is a single uniformly distributed random number in the interval (0, 1) independent of each other. To compare the performance, we performed 500 Monte Carlo simulations on MATLAB (Natick, MA,USA) [97].

### 3.4.3 Results and Discussion

When choosing an optimal tracking algorithm, there is typically a tradeoff between tracking accuracy and computation time. Maintaining a high-level accuracy in complex scenarios where multiple targets need to be tracked simultaneously and the environment is particularly noisy requires significant computation time. Traditionally, the tracking accuracy of an estimator is calculated in terms of a miss-distance, or localization error, between a reference value and its estimated value. In our context, we are also interested in evaluating missed detections and false alarms. The generalized optimal sub-pattern assignment (GOSPA) metric has been designed to reflect this performance [22]. Informally, the GOSPA metric can be defined as

$$\text{GOSPA} = \sum \text{localization error} + \frac{\text{cutoff distance}}{2}(\#\text{ of missed detections} + \#\text{ of false alarms})$$

(for a precise detailed description of GOSPA, see [98]). The localization error is for pairs of true targets and target estimates that are sufficiently close. A missed detection is declared if there is no corresponding target sufficiently close to it, and a false alarm is declared if there is no corresponding true target sufficiently close to it. To evaluate the performance of the different tracking algorithms described in Section 3.3 in terms of the miss-distance, the GOSPA metric, and computation time, we considered multiple scenarios with varying levels of complexities.

[Figure 33 a,b] depict two such scenarios where the trajectories of three targets are shown for clutter densities $\lambda = 1 \times 10^{-4}/m^2$ and $\lambda = 5 \times 10^{-4}/m^2$, respectively. The scenario depicted in [Figure 33b] is more demanding because of the increased number of false alarms. [Figure 34 a,b] compares the performance of LSPA and DWPDA in terms of tracking accuracy,

calculated using the root mean square (RMS) position error, over a period of 100-time intervals for the above two scenarios, respectively. We see that the RMS position errors for LSPA stay close to 0 over the entire period while they are always increasing for DWPDA

as time progresses. We see similar results while tracking six crossing targets with clutter densities $\lambda = 1 \times 10^{-4}/m^2$ and $\lambda = 5 \times 10^{-4}/m^2$, as depicted in [Figure 35 a,b]. [Figure 36 a,b] show that the computation times for the two clutter scenarios as the number of targets increases from one to six. We see that the computation times required for LSPA are only slightly higher than those of DWPDA. Finally, [Table 8] and [Table 9] summarize the results for the

performance in terms of the GOSPA metric based on the Euclidean distance with a cutoff parameter of 30. [Table 8] shows the average GOSPA errors as we increase the number of targets from 1 to 6 while keeping the clutter density constant at $\lambda = 3 \times 10^{-4}/m^2$. We can see that irrespective of the number of targets that are being tracked, GOSPA errors for LSPA are only a fraction of the errors for DWPDA. We see a similar pattern in [Table 9] where we increase the clutter density from $\lambda = 1 \times 10^{-4}/m^2$to $\lambda = 5 \times 10^{-4}/m^2$while keeping the number of targets fixed. The poor performance of DWPDA is expected, since the algorithm is ill-equipped to deal with the problem of DA in MTT, and GOSPA appropriately penalizes any missed detections and false alarms. From the above results, it is evident that LSPA is superior to DWPDA in terms of tracking accuracy in all scenarios without trading off much computation time. This superior tracking accuracy can be attributed to the reduction in the association probabilities of false measurements in the overlapping validation regions from multiple targets and, at the same time, the increase in the association probabilities for actual target

measurements. From the above results, we know that LSPA can improve the tracking performance for MTT in densely cluttered environments.

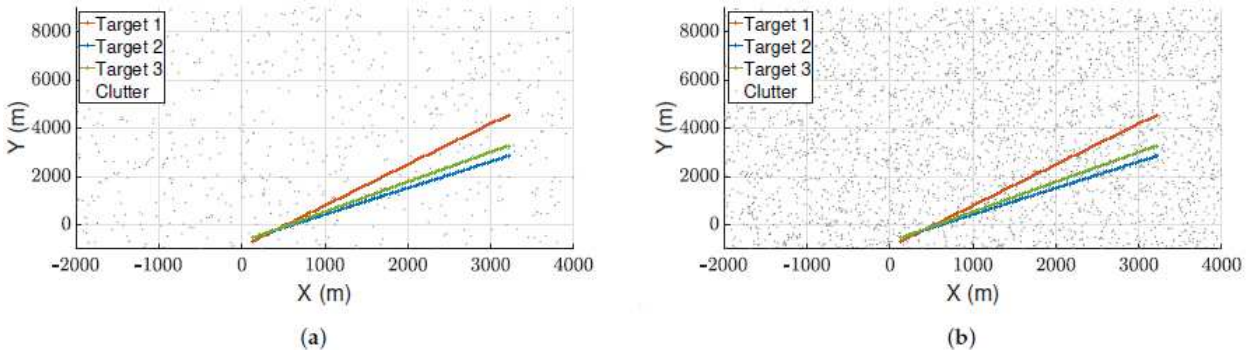



**Figure 33: True target positions for three crossing targets with different clutter densities. (a) Clutter density $\lambda = 1 \times 10^{-4}/m^2$; and (b) clutter density $\lambda = 5 \times 10^{-4}/m^2$**

**Table 8: Average generalized optimal sub-pattern assignment (GOSPA) errors for tracking multiple crossing targets with clutter density $\lambda = 1 \times 10^{-4}/m^2$.**

| | Number of Targets | | | | | |
|---|---|---|---|---|---|---|
| | 1 | 2 | 3 | 4 | 5 | 6 |
| LSPA | 0.4571 | 0.9103 | 1.5033 | 1.9659 | 2.3441 | 2.9000 |
| DWLSPA | 0.4435 | 0.8787 | 1.5243 | 1.8601 | 2.2664 | 2.8253 |
| JPDA | 0.6751 | 2.5626 | 4.6425 | 6.1505 | 9.3851 | 11.5240 |
| DWPDA | 4.7454 | 9.1838 | 14.2879 | 18.5087 | 23.3401 | 27.4684 |

**Table 9: Average GOSPA errors for tracking three targets with different clutter densities.**

| | $\lambda$ | | | | |
|---|---|---|---|---|---|
| | $1 \times 10^{-4}$ | $2 \times 10^{-4}$ | $3 \times 10^{-4}$ | $4 \times 10^{-4}$ | $5 \times 10^{-4}$ |
| LSPA | 1.4280 | 1.6058 | 1.4056 | 1.6628 | 1.7690 |
| DWLSPA | 1.3718 | 1.5652 | 1.3388 | 1.5289 | 1.6257 |
| JPDA | 3.5367 | 4.2030 | 4.3475 | 4.4828 | 5.2677 |
| DWPDA | 7.4128 | 11.8464 | 14.4591 | 15.0590 | 15.8318 |

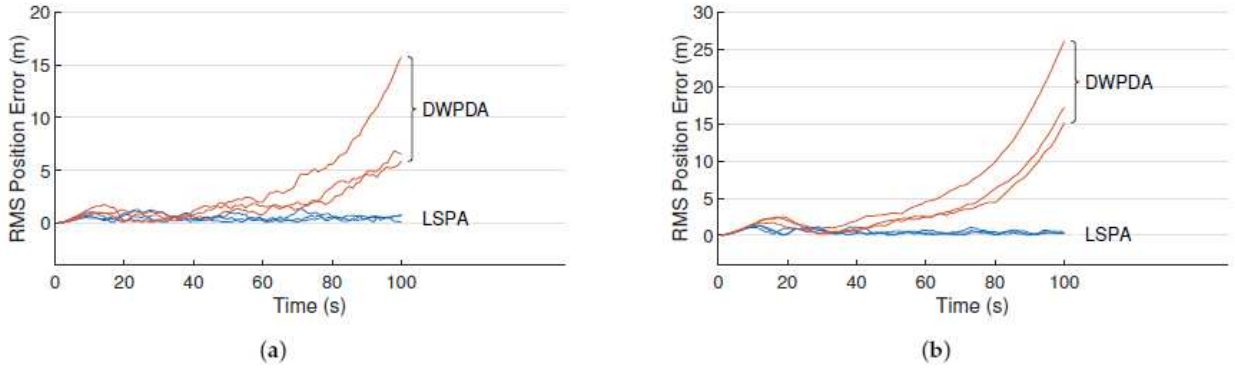



**Figure 34: RMS position error for three crossing targets using DWPDA and LSPA with different clutter densities. (a) Clutter density $\lambda = 1 \times 10^{-4}/m^2$; and (b) clutter density $\lambda = 5 \times 10^{-4}/m^2$.**

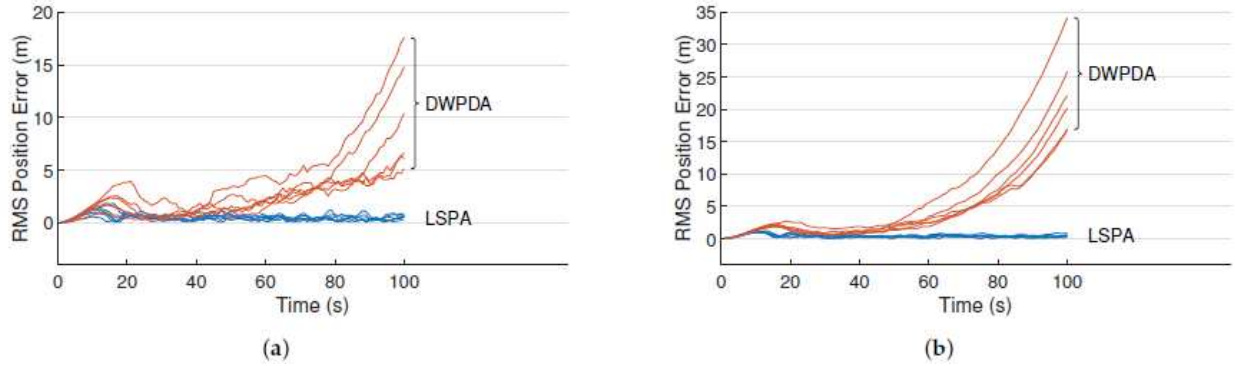



**Figure 35: RMS position error for six crossing targets using DWPDA and LSPA with different clutter densities. (a) Clutter density $\lambda = 1 \times 10^{-4}/m^2$; and (b) clutter density $\lambda = 5 \times 10^{-4}/m^2$.**

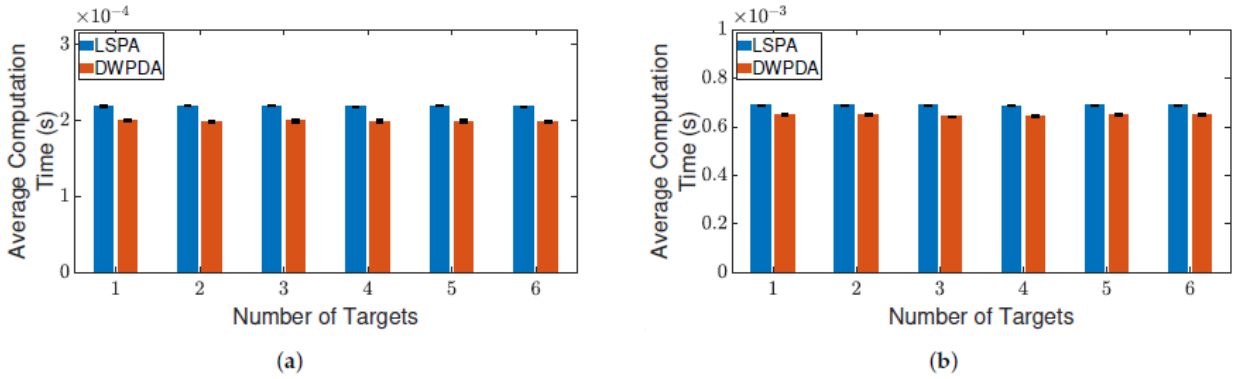



**Figure 36: Average computation time for obtaining association probabilities using DWPDA and LSPA for tracking multiple crossing targets with different clutter densities. (a) Clutter**

**density λ = $\lambda = 1 \times 10^{-4}/m^2$; and (b) clutter density $\lambda = 5 \times 10^{-4}/m^2$. Error bars indicate 95% confidence intervals.**

Next, we compare the performance of LSPA with JPDA, in terms of computation time and RMS position error, as the number of targets increases from one to six. [Figure 37] and [Figure 38] show the results of this comparison for the two clutter scenarios. We can clearly see that the average computation time of LSPA is much smaller than that of JPDA, so much so that the LSPA computation times are not even visible in [Figure 37a] and [Figure 38a]. However, in contrast to the comparison of LSPA and DWPDA in terms of the RMS position error, in the case of LSPA compared with JPDA, [Figure 37b] and [Figure 38b] clearly show that there is little difference in RMS position error. The same observation applies in terms of computation time and RMS position error in [Figure 39 a,b] respectively when we compare LSPA with JPDA in scenarios involving a fixed number of targets as the clutter density increases from $\lambda = 1 \times 10^{-4}/m^2$to $\lambda = 5 \times 10^{-4}/m^2$. While there is little difference between LSPA and JPDA in terms of RMS position errors, [Table 8] and [Table 9] show that GOSPA errors for LSPA are less than 1/3rd of GOSPA errors for JPDA across almost all scenarios. These relatively higher GOSPA errors for JPDA can be explained by a few missed detections when multiple target paths overlap. These missed detections are rightly penalized in the GOSPA metric. These results show that LSPA dominates JPDA in terms of computation time while maintaining a high level of tracking accuracy. This dramatic reduction in computation time for LSPA can be explained by the implementation of the loopy factor graph, resulting in simultaneous updating of the joint association probabilities for multiple targets during each iteration of LSPA. The results show that LSPA scales well for real-time applications involving complex tracking scenarios.

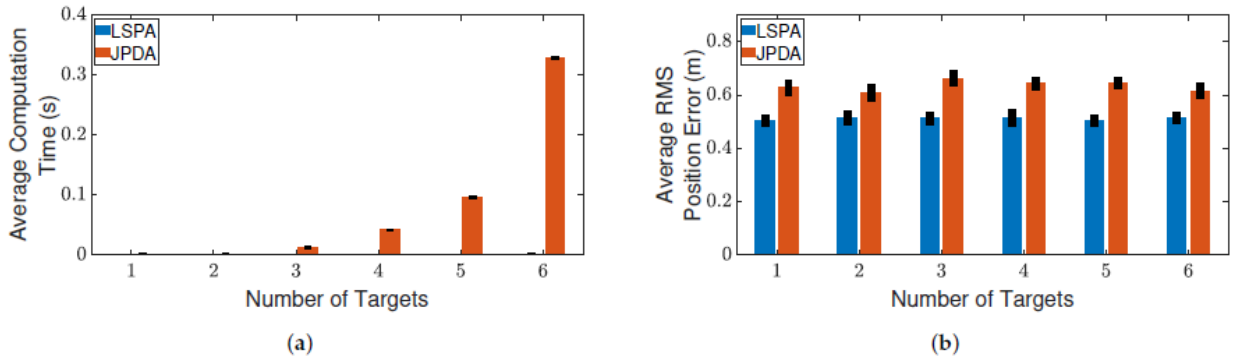



**Figure 37: Tracking multiple crossing targets using LSPA and JPDA with clutter density $\lambda = 1 \times 10^{-4}/m^2$. (a) Average computation time for obtaining association probabilities; (b) average RMS position error. Error bars indicate 95% confidence intervals.**

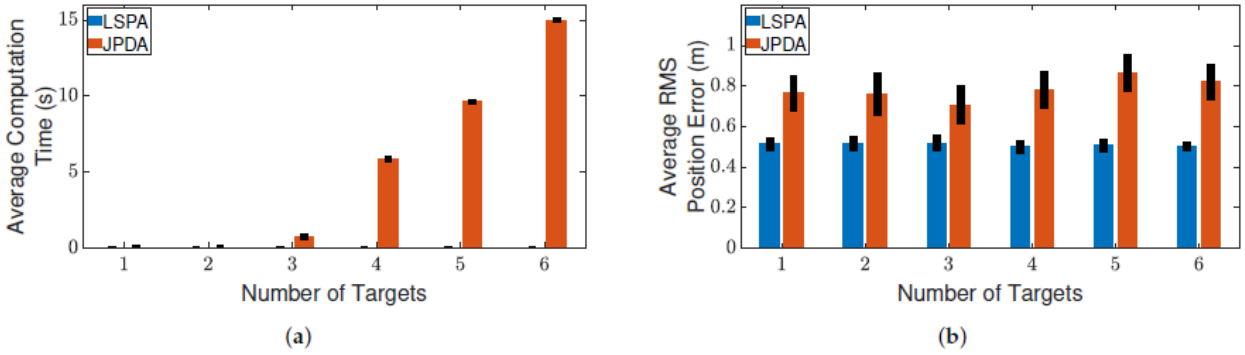



**Figure 38: Tracking multiple crossing targets using loopy sum-product algorithm (LSPA) and joint probabilistic data association (JPDA) with clutter density $\lambda = 5 \times 10^{-4}/m^2$. (a) Average computation time for obtaining association probabilities; (b) average root mean square (RMS) position error. Error bars indicate 95% confidence intervals.**

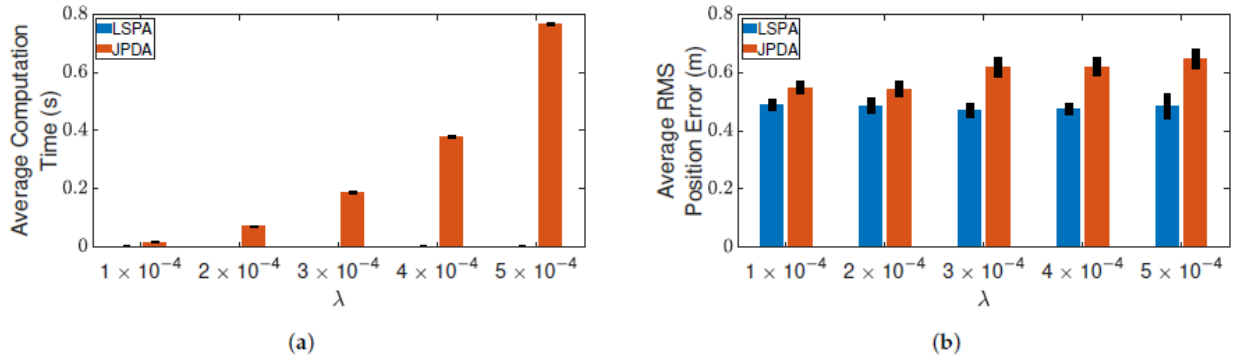



**Figure 39: Tracking three crossing targets using LSPA and JPDA. (a) Average computation time for obtaining association probabilities; (b) average RMS position error. Error bars indicate 95% confidence intervals**

Finally, to see whether the integration of LSPA with the distance-weighting scheme has any effect on its performance, we compare LSPA with DWLSPA in terms of the same two metrics as above. [Figure 40] and [Figure 41] show the results of this comparison for two clutter scenarios respectively, as the number of targets increases from one to six. There is little difference between LSPA and DWLSPA in terms of computation times, as shown in Figure 40a] and [Figure 41a]. Similarly, we can see that the RMS position errors for LSPA and DWLSPA are almost identical in Figure 40b] and Figure 41b]. [Figure 42 a,b] show that the difference between LSPA and DWLSPA remains negligible, in terms of both computation time and RMS position error, for scenarios with a fixed number of targets and varying clutter densities. [Table 8] and [Table 9] show that GOSPA errors for LSPA and DWLSPA are comparable across all tested scenarios. This means that in addition to the localization errors, the missed detections and false alarms remain consistent between LSPA and DWLSPA, and the potential advantage of DWLSPA with the additional distance-weighting information is not apparent. Surprisingly, LSPA and DWLSPA perform equally well in every scenario in terms of both tracking accuracy and computation time. The unchanged performance of DWLSPA can be explained by the initiation of LSPA with small improvements in single-target association probabilities having insignificant effect on joint association probabilities calculated at the end of a large number of iterations. However, this unexpected lack of improvement contrasts sharply with results reported in [78] showing significant improvement when introducing distance weighting relative to PDA. This result is interesting and useful because we can see that distance weighting does not always lead to

better performance. Moreover, the following important observation remains: LSPA is reliable and efficient for tracking multiple objects in cluttered environments.

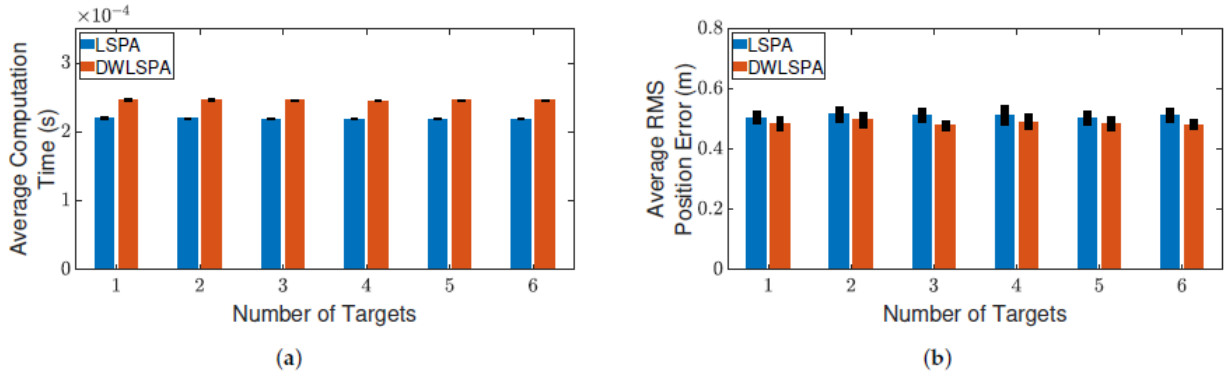



**Figure 40: Tracking multiple crossing targets using LSPA and DWLSPA with clutter density $\lambda = 1 \times 10^{-4}/m^2$. (a) Average computation time for obtaining association probabilities; (b) average RMS position error. Error bars indicate 95% confidence intervals.**

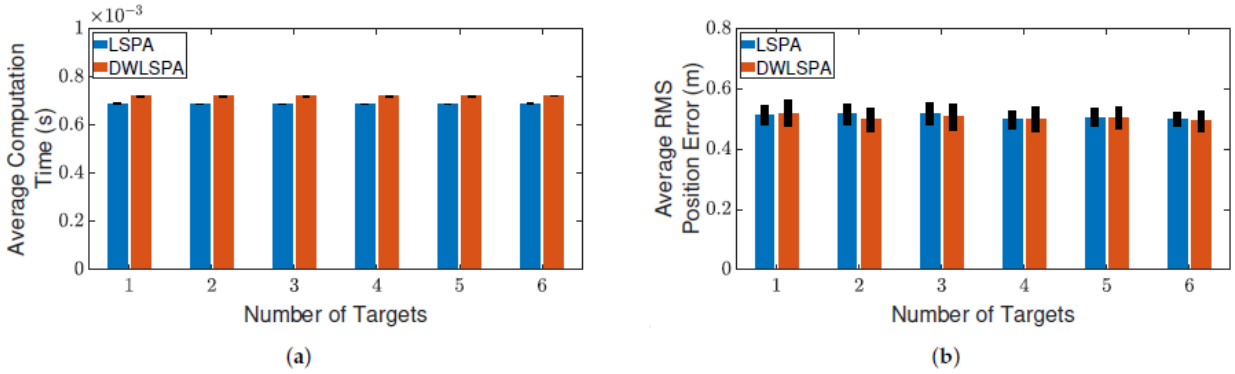



**Figure 41: Tracking multiple crossing targets using LSPA and DWLSPA with clutter density $\lambda = 5 \times 10^{-4}/m^2$. (a) Average computation time for obtaining association probabilities; (b) average RMS position error. Error bars indicate 95% confidence intervals.**

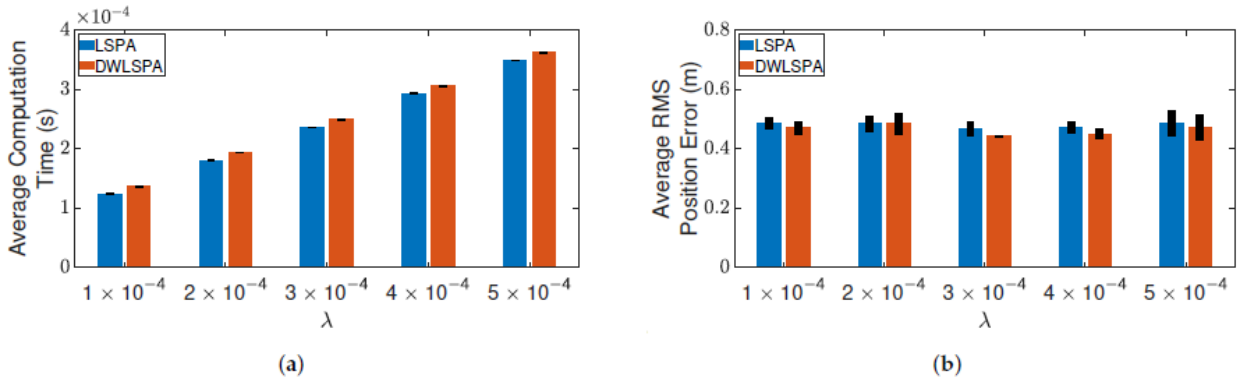



**Figure 42: Tracking three crossing targets using LSPA and DWLSPA. (a) Average computation time for obtaining association probabilities; (b) average RMS position error. Error bars indicate 95% confidence intervals.**

## 3.5 Conclusion

In this chapter, we formulated a distance-weighting PDA approach for LSPA and examined its effect for tracking multiple objects in cluttered environments. It has been previously shown that a modification of PDA according to a weighting scheme based on distances between predicted and true target positions improves the tracking accuracy of PDA. LSPA is known to be better than PDA and JPDA and, since PDA constitutes a crucial building block of LSPA, we expected the integration of DWPDA with LSPA to boost the overall performance even further. We studied the performance of LSPA against DWPDA, JPDA, and DWLSPA for a wide range of tracking scenarios involving multiple targets and varying clutter densities. Our results confirm that LSPA is superior to DWPDA in terms of tracking accuracy and dominates JPDA in terms of computation time. However, contrary to expectations, we found that the distance-weighting approach, when integrated with LSPA, does not enhance the performance of LSPA in terms of either tracking accuracy or computation time. The simulation scenarios in the experiment could be made more realistic with the addition of appearing and disappearing targets and time-

varying target velocities. These scenarios add extra layers of complexity to the DA without affecting the conclusions we draw in this paper. Regardless, we demonstrated the validity of LSPA having computational requirements suitable for real-time processing and accuracy of tracking multiple targets in cluttered environments.

# Dissertation Conclusion

The rapid growth of digital data has challenged many analyst's ability to store, process, and interpret Big Data. If the information is not processed in a timely manner, the information may be lost forever due to storage challenges. A Big Data Actionable Intelligence (BDAI) architecture was developed to overcome the challenge by transforming Big Data into a decision-making process. This architecture provides a framework for machine-learning algorithms to learn and analyze streaming data from heterogenous data sources and transform data into actionable information for decision makers. To overcome the challenges of detecting moving targets using Remote Sensing Big Data, a new real-time detection system was developed. This patented approach has shown to be significantly better than the current existing state-of-the-art detection methods to detect far away objects that cannot be easily seen. Finally, research was performed to gain major understanding on techniques that can support real-time tracking of large number quantity of targets. For future research, a decentralized fusion architecture will be investigated to further reduce latency of Big Data processing.

References


[1] Sandia Labs News Service (2019). “Wrangling Big Data”, *Albuquerque Journal*, November 4, 2019. https://www.abqjournal.com/1386752/wrangling-big-data-to-locate-actionable-info-a-lot-faster.html

[2] Reinsel, D., Gantz, J., & Rydning, J. (2018). *Data Age 2025 - The Digitization of the World From Edge to Core.* Framingham, MA: International Data Corporation (IDC). https://www.seagate.com/files/www-content/our-story/trends/files/idc-seagate-dataage-whitepaper.pdf

[3] Ma, P., and Sun, X. (2015), “Leveraging for Big Data Regression,” *Wiley Interdisciplinary Reviews: Computational Statistics*, 7, 70–76. https://doi.org/10.1002/wics.1324

[4] Qiu, J., Wu, Q., Ding, G. *et al.* “A survey of machine learning for big data processing”. *EURASIP J. Adv. Signal Process.* 2016, 67 (2016). https://doi.org/10.1186/s13634-016-0355-x

[5] Majumdar, J., Naraseeyappa, S. & Ankalaki, S. Analysis of agriculture data using data mining techniques: application of big data. *J Big Data* 4, 20 (2017). https://doi.org/10.1186/s40537-017-0077-4

[6] B. Chandramouli, J. Goldstein and S. Duan, "Temporal Analytics on Big Data for Web Advertising," *2012 IEEE 28th International Conference on Data Engineering*, Washington, DC, 2012, pp. 90-101. https://ieeexplore.ieee.org/document/6228075

[7] M. Mazhar Rathore, Awais Ahmad, Anand Paul, Seungmin Rho, “Urban planning and building smart cities based on the Internet of Things using Big Data analytics”, *Computer Networks*, Volume 101, 2016, Pages 63-80, ISSN 1389-1286. https://www.sciencedirect.com/science/article/pii/S1389128616000086

[8] D. Zhou *et al*., "Distributed Data Analytics Platform for Wide-Area Synchrophasor Measurement Systems," in *IEEE Transactions on Smart Grid*, vol. 7, no. 5, pp. 2397-2405, Sept. 2016. https://ieeexplore.ieee.org/iel7/5165411/5446437/07420696.pdfReinsel, D., Gantz, J., & Rydning, J. (2018). *Data Age 2025 - The Digitization of the World From Edge to Core.* Framingham, MA: International Data Corporation (IDC).

[9] National Spatial Data Infrastructure (NSDI), "Presidential Documents", Federal Register. Vol. 59, No. 71 Wednesday, April 13, 1993. https://www.archives.gov/files/federal-register/executive-orders/pdf/12906.pdf

[10] Waze. https://www.waze.com/

[11] Twitter Data Source. https://twitter.com/?lang=en

[12] Travel Midwest Data Source. https://www.travelmidwest.com

[13] City of Chicago Data Source. https://www.chicago.gov/city/en.html

[14] GDELT Data Source. https://www.gdeltproject.org/

[15] Mapquest Data Source. https://www.mapquest.com/

[16] Digital Globe Data Source. https://www.digitalglobe.com/

[17] E. Necula, "Dynamic Traffic Flow Prediction Based on GPS Data," 2014 IEEE 26th International Conference on Tools with Artificial Intelligence, Limassol, 2014, pp. 922-929. https://ieeexplore.ieee.org/document/6984576

[18] Y. Lv, Y. Chen, X. Zhang, Y. Duan and N. L. Li, "Social media based transportation research: the state of the work and the networking," in *IEEE/CAA Journal of Automatica Sinica*, vol. 4, no. 1, pp. 19-26, Jan. 2017. https://ieeexplore.ieee.org/document/7815548

[19] J. Barros, M. Araujo and R. J. F. Rossetti, "Short-term real-time traffic prediction methods: A survey," 2015 International Conference on Models and Technologies for Intelligent Transportation Systems (MT-ITS), Budapest, 2015, pp. 132-139. https://ieeexplore.ieee.org/abstract/document/7223248

[20] H. Hu, Y. Wen, T. Chua and X. Li, "Toward Scalable Systems for Big Data Analytics: A Technology Tutorial," in *IEEE Access*, vol. 2, pp. 652-687, 2014, doi: 10.1109/ACCESS.2014.2332453.

[21] S. Marchal, X. Jiang, R. State and T. Engel, "A Big Data Architecture for Large Scale Security Monitoring," *2014 IEEE International Congress on Big Data*, Anchorage, AK, 2014, pp. 56-63, doi: 10.1109/BigData.Congress.2014.18.

[22] Zhijiang Chen, Guobin Xu, Vivek Mahalingam, Linqiang Ge, James Nguyen, Wei Yu, Chao Lu, "A Cloud Computing Based Network Monitoring and Threat Detection System for Critical Infrastructures", Big Data Research, Volume 3, 2016, Pages 10-23, ISSN 2214-5796. https://doi.org/10.1016/j.bdr.2015.11.002.

[23] Pedro Casas, Alessandro D'Alconzo, Tanja Zseby, and Marco Mellia. 2016. "Big-DAMA: Big Data Analytics for Network Traffic Monitoring and Analysis." In Proceedings of the 2016 workshop on Fostering Latin-American Research in Data Communication Networks (LANCOMM '16). Association for Computing Machinery, New York, NY, USA, 1–3. DOI:https://doi.org/10.1145/2940116.2940117

[24] Julie Zhu, Bo Tang, Victor Li, "A five-layer architecture for big data processing and analytics", Int. J. Big Data Intelligence, Vol. 6, No. 1, 2019.

[25] Weiming Liu, Chen Zhang, Bin Yu, and Yitong Li. 2019. "A General Multi-Source Data Fusion Framework." In Proceedings of the 2019 11th International Conference on Machine Learning and Computing (ICMLC '19). Association for Computing Machinery, New York, NY, USA, 285–289. https://dl.acm.org/doi/abs/10.1145/3318299.3318394

[26] NIST Big Data Public Working Group (NBD-PWG), "NIST Special Publication 1500-1: NIST Big Data Interoperability Framework: Volume 1, Definitions", National Institute of Standards and Technology, California, September 2015. http://dx.doi.org/10.6028/NIST.SP.1500-1

[27] Gema Bello-Orgaz, Jason J. Jung, David Camacho, "Social big data: Recent achievements and new challenges", Information Fusion, Volume 28, 2016, Pages 45-59. https://www.sciencedirect.com/science/article/pii/S1566253515000780

[28] https://www.cloudera.com/downloads/hdp.html

[29] https://hadoop.apache.org/docs/r1.2.1/hdfs_design.html

[30] Microsoft Azure Stack. https://azure.microsoft.com/en-us/overview/azure-stack/

[31] Java Programming Language. https://www.java.com/en/

[32] Python Programming Language: https://www.python.org/

[33] Apache Storm. https://storm.apache.org/index.html

[34] Apache Kafka. https://kafka.apache.org/

[35] Hamid Nasiri, Saeed Nasehi, and Maziar Goudarzi, “Evaluation of distributed stream processing frameworks for IoT applications in Smart Cities”, Journal of Big Data, 6, Article number: 52 (2019). https://link.springer.com/article/10.1186/s40537-019-0215-2

[36] Aung, H. Yin Min and A. Htein Maw, "Performance Evaluation for Real-Time Messaging System in Big Data Pipeline Architecture," *2018 International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery (CyberC)*, Zhengzhou, China, 2018, pp. 198-1986, doi: 10.1109/CyberC.2018.00047.

[37] Apache Lucene. https://lucene.apache.org/solr/

[38] Joseph Redmon, Santosh Divvala, Ross Girshick, Ali Farhadi; “You Only Look Once: Unified, Real-Time Object Detection”, The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016, pp. 779-788. https://ieeexplore.ieee.org/document/7780460

[39] Snidaro, Lauro et al., “Context-Enhanced Information Fusion: Boosting Real-World Performance with Domain Knowledge”, Springer, 2016. https://link.springer.com/content/pdf/10.1007/978-3-319-28971-7.pdf

[40] Banana Dashboard. https://doc.lucidworks.com/lucidworks-hdpsearch/2.5/Guide-Banana.html

[41] Tian J. Ma, Rudy J. Garcia, Forest Danford, Laura Patrizi, Jennifer Galasso, and Jason Loyd; “Big Data Actionable Intelligence Architecture”, Springer Journal of Big Data, September 2020, DOI: 10.1186/s40537-020-00378-7 https://rdcu.be/cbdiv.

[42] Y. Wang, Z. Luo, and P.-M. Jodoin, “Interactive deep learning method for segmenting moving objects,” Pattern Recognition Letters, Sep. 2016. https://doi.org/10.1016/j.patrec.2016.09.014.

[43] P. Bideau, A. R. Chowdhury, R. R. Menon, and E. Learned-Miller, "The best of both worlds: combining CNNs and geometric constraints for hierarchical motion segmentation," IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2018.

[44] L.A. Lim, and H.Y. Keles, "Learning multi-scale features for foreground segmentation," Pattern Anal Applic, 2019. https://doi.org/10.1007/s10044-019-00845-9.

[45] Jayme Garcia Arnal Barbedo, “Impact of dataset size and variety on the effectiveness of deep learning and transfer learning for plant disease classification”, Computers and Electronics in Agriculture, Volume 153, 2018, Pages 46-53, ISSN 0168-1699, https://doi.org/10.1016/j.compag.2018.08.013.

[46] M. Vakalopoulou, K. Karantzalos, N. Komodakis and N. Paragios, "Building detection in very high resolution multispectral data with deep learning features," 2015 IEEE International Geoscience and Remote Sensing Symposium (IGARSS), 2015, pp. 1873-1876, doi: 10.1109/IGARSS.2015.7326158.

[47] Rudin, C. “Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead”. Nat Mach Intell 1, 206–215 (2019). https://doi.org/10.1038/s42256-019-0048-xD.

[48] Chen Sun, Abhinav Shrivastava, Saurabh Singh, Abhinav Gupta. “Revisiting Unreasonable Effectiveness of Data in Deep Learning Era.” The IEEE International Conference on Computer Vision (ICCV), 2017, pp. 843-852. https://doi.org/10.1109/ICCV.2017.97.

[49] Dani Deahl, “Volvo’s self-driving cars are having trouble recognizing kangaroos”, The Verge, July 3, 2017. https://www.theverge.com/2017/7/3/15916076/volvo-self-driving-cars-trouble-recognizing-kangaroos

[50] N. Akhtar and A. Mian, "Threat of Adversarial Attacks on Deep Learning in Computer Vision: A Survey," in IEEE Access, vol. 6, pp. 14410-14430, 2018, doi: 10.1109/ACCESS.2018.2807385.

[51] X. Yuan, P. He, Q. Zhu and X. Li, "Adversarial Examples: Attacks and Defenses for Deep Learning," in IEEE Transactions on Neural Networks and Learning Systems, vol. 30, no. 9, pp. 2805-2824, Sept. 2019, doi: 10.1109/TNNLS.2018.2886017.

[52] M. Piccardi, "Background subtraction techniques: a review," 2004 IEEE International Conference on Systems, Man and Cybernetics (IEEE Cat. No.04CH37583), 2004, pp. 3099-3104 vol.4, doi: 10.1109/ICSMC.2004.1400815.

[53] Jacques, J., Claudio, R.J., Soraia, R.M.: Background subtraction and shadow detection in grayscale video sequences. In 18th Brazilian Symposium on Computer Graphics and Image Processing, pp 189–196. IEEE (2005)

[54] Yang, J., Yang, W., Li, M.: An efficient moving object detection algorithm based on improved GMM and cropped frame technique. In: IEEE International Conference on Mechatronics and Automation, pp 658–663. IEEE (2012)

[55] Yin, J., Liu, L., Li, H., Liu, Q.: The infrared moving object detection and security detection related algorithms based on w4 and frame difference. Infrared Phys. Technol. 77, 302–315 (2016)

[56] Sengar, S.S., Mukhopadhyay, S. Moving object detection based on frame difference and W4. SIViP 11, 1357–1364 (2017). https://doi.org/10.1007/s11760-017-1093-8

[57] Z. Zivkovic, "Improved adaptive Gaussian mixture model for background subtraction," Proceedings of the 17th International Conference on Pattern Recognition, 2004. ICPR 2004., 2004, pp. 28-31 Vol.2, doi: 10.1109/ICPR.2004.1333992.

[58] Z. Zhou, X. Li, J. Wright, E. Candès and Y. Ma, "Stable Principal Component Pursuit," 2010 IEEE International Symposium on Information Theory, 2010, pp. 1518-1522, doi: 10.1109/ISIT.2010.5513535.

[59] N. Vaswani, T. Bouwmans, S. Javed and P. Narayanamurthy, "Robust Subspace Learning: Robust PCA, Robust Subspace Tracking, and Robust Subspace Recovery," in IEEE Signal

Processing Magazine, vol. 35, no. 4, pp. 32-55, July 2018, doi: 10.1109/MSP.2018.2826566.

[60] Reed, IS.; Gagliardi, R.M.; Shao, H. M., "Application of Three-Dimensional Filtering to Moving Target Detection," Aerospace and Electronic Systems, IEEE Transactions on, vol. AES-19, no.6, pp.898,905, Nov. 1983

[61] Reed, Irving S.; Gagliardi, R.M.; Stotts, L.B., "A recursive moving-target-indication algorithm for optical image sequences," Aerospace and Electronic Systems, IEEE Transactions on , vol.26, no.3, pp.434,440, May 1990

[62] Wei Yi; Morelande, M.R.; Lingjiang Kong; Jianyu Yang, "An Efficient Multi-Frame Track-Before-Detect Algorithm for Multi-Target Tracking," Selected Topics in Signal Processing, IEEE Journal of , vol.7, no.3, pp.421,434, June 2013

[63] D. Orlando, L. Venturino, M. Lops, and G. Ricci, “Track-before-detect strategies for STAP radars,” IEEE Trans. Signal Process., vol. 58, no.2, pp. 933–938, Feb. 2010

[64] Buzzi, S.; Lops, M.; Venturino, L. "Track-before-detect procedures for early detection of moving target from airborne radars", Aerospace and Electronic Systems, IEEE Transactions on, On page(s): 937 - 954 Volume: 41, Issue: 3, July 2005

[65] S. Buzzi, M. Lops, L. Venturino, and M. Ferri, “Track-before-detect procedures in a multi-target environment,” IEEE Trans. Aerosp. Electron. Syst., vol. 44, no. 3, pp. 1135–1150, Jul. 2008

[66] Grossi, E.; Lops, M.; Venturino, L. "A Novel Dynamic Programming Algorithm for Track-Before-Detect in Radar Systems", IEEE Transactions on Signal Processing, On page(s): 2608 - 2619 Volume: 61, Issue: 10, May15, 2013

[67] Grossi, E.; Lops, M.; Venturino, L. "A Heuristic Algorithm for Track-Before-Detect With Thresholded Observations in Radar Systems", Signal Processing Letters, IEEE, On page(s): 811 - 814 Volume: 20, Issue: 8, Aug. 2013

[68] Chan, Y.T.; Niezgoda, G.H.; Morton, S. P., "Passive sonar detection and localization by matched velocity filtering," Oceanic Engineering, IEEE Journal of, vol.20, no.3, pp.179,189, Jul 1995

[69] Bin Wu; Hao Yan "A Novel Track-before-Detect Algorithm for Small Dim Infrared Target", Multimedia and Signal Processing (CMSP), 2011 International Conference on, On page(s): 103 - 106 Volume: 1, 14-15 May 2011

[70] L. Úbeda-Medina, Á. F. García-Fernández and J. Grajal, "Adaptive Auxiliary Particle Filter for Track-Before-Detect With Multiple Targets," in IEEE Transactions on Aerospace and Electronic Systems, vol. 53, no. 5, pp. 2317-2330, Oct. 2017, doi: 10.1109/TAES.2017.2691958.

[71] J. Wang, W. Yi, T. Kirubarajan and L. Kong, "An Efficient Recursive Multiframe Track-Before-Detect Algorithm," in IEEE Transactions on Aerospace and Electronic Systems, vol. 54, no. 1, pp. 190-204, Feb. 2018, doi: 10.1109/TAES.2017.2741898.

[72] W. Yi, Z. Fang, W. Li, R. Hoseinnezhad and L. Kong, "Multi-Frame Track-Before-Detect Algorithm for Maneuvering Target Tracking," in IEEE Transactions on Vehicular Technology, vol. 69, no. 4, pp. 4104-4118, April 2020, doi: 10.1109/TVT.2020.2976095.

[73] Grossi, E.; Lops, M.; Venturino, L., "A Track-Before-Detect Algorithm With Thresholded Observations and Closely-Spaced Targets," IEEE Signal Processing Letters, vol.20, no.12, pp.1171,1174, Dec. 2013

[74] Manuel Guizar-Sicairos, Samuel T. Thurman, and James R. Fienup, "Efficient subpixel image registration algorithms," Opt. Lett. 33, 156-158 (2008)

[75] X. Rong Li and V. P. Jilkov, "Survey of maneuvering target tracking. Part I. Dynamic models," in IEEE Transactions on Aerospace and Electronic Systems, vol. 39, no. 4, pp. 1333-1364, Oct. 2003, doi: 10.1109/TAES.2003.1261132.

[76] Tian J. Ma, “Object Detection and Tracking System”, U.S. Patent No. 9,665,942, issued May 30, 2017

[77] Bar-Shalom, Y.; Tse, E. Tracking in a Cluttered Environment With Probabilistic Data Association. Automatica 1975, 11, 451–460. [CrossRef]

[78] Welch, G.; Bishop, G. An Introduction to the Kalman Filter; Technical Report; University of North Carolina at Chapel Hill: Chapel Hill, NC, USA, 1995.

[79] Kschischang, F.R.; Brendan, J.F.; Loeliger, H.-A. Factor Graphs and the Sum-Product Algorithm. IEEE Trans. Inf. Theory 2001, 47, 498–519. [CrossRef]

[80] Chen, X.; Li, Y.; Li, Y.; Yu, J.; Li, X. A Novel Probabilistic Data Association for Target Tracking in a Cluttered Environment. Sensors 2016, 16, 2180. [CrossRef] [PubMed]

[81] Williams, J.L.; Lau, R.A. Data Association by Loopy Belief Propagation. In Proceedings of the 13th International Conference on Information Fusion, Edinburgh, SA, Australia, 26–29 July 2010; pp. 1–8.

[82] Li, X.; Bar-Shalom, Y. Tracking in Clutter With Nearest Neighbor Filters: Analysis and Performance. IEEE Trans. Aerosp. Electron. Syst. 1996, 32, 995–1010.

[83] Fortmann, T.E.; Bar-Shalom, Y.; Scheffe, M. Sonar Tracking of Multiple Targets Using Joint Probabilistic Data Association. IEEE J. Ocean. Eng. 1983, 8, 173–184. [CrossRef]

[84] Rezatofighi, S.H.; Milan, A.; Zhang, Z.; Shi, Q.; Dick, A.; Reid, I. Joint Probabilistic Data Association Revisited. In Proceedings of the IEEE International Conference on Computer Vision, Santiago, Chile, 13–16 December, 2015; pp. 3047–3055.

[85] Blackman, S. Multiple Hypothesis Tracking for Multiple Target Tracking. IEEE Trans. Aerosp. Electron. Syst. 2004, 19, 5–18. [CrossRef]

[86] Roecker, J.A.; Phillis, G.L. Suboptimal Joint Probabilistic Data Association. IEEE Trans. Aerosp. Electron. Syst. 1993, 29, 510–517. [CrossRef]

[87] Reid, D.B. An Algorithm for Tracking Multiple Targets. IEEE Trans. Automat. Cont. 1979, 24, 843–854. [CrossRef]

[88] Bar-Shalom, Y.; Willett, P.; Tian, X. Tracking and Data Fusion: A Handbook of Algorithms; YBS Publishing: Storrs, CT, USA, 2011.

[89] Wiberg, N; Loeliger, H.-A.; Kotter, R. Codes and Iterative Decoding on General Graphs. Eur. Trans. Telecomm. 1996, 6, 513–525.

[90] Williams, J.L.; Lau, R.A. Convergence of Loopy Belief Propagation for Data Association. In Proceedings of the Sixth International Conference on Intelligent Sensors, Sensor

Networks and Information Processing, Brisbane, QLD, Australia, 7–10 December 2010; pp. 175–180.

[91] Williams, J.L.; Lau, R.A. Approximate Evaluation of Marginal Association Probabilities with Belief Propagation. IEEE Trans. Aerosp. Electron. Syst. 2014, 4, 2942–2959. [CrossRef]

[92] Meyer, F.; Braca, P.; Willett, P.; Hlawatsch, F. Scalable multitarget tracking using multiple sensors: A belief propagation approach. In Proceedings of the 18th International Conference on Information Fusion (Fusion), Washington, DC, USA, 6–9 July 2015; pp. 1778–1785.

[93] Meyer, F.; Braca, P.; Willett, P.; Hlawatsch, F. Tracking an unknown number of targets using multiple sensors: A belief propagation method. In Proceedings of the 19th International Conference on Information Fusion (FUSION), Heidelberg, Germany, 5–8 July 2016; pp. 719–726.

[94] Meyer, F.; Braca, P.; Willett, P.; Hlawatsch, F. A scalable algorithm for tracking an unknown number of targets using multiple sensors. IEEE Trans. Signal Process. 2017, 65, 3478–3493. [CrossRef]

[95] Meyer, F.; Kropfreiter, T.; Williams, J.L.; Lau, R.; Hlawatsch, F.; Braca, P.; Win, M.Z. Message passing algorithms for scalable multitarget tracking. Proc. IEEE 2018, 106, 221–259. [CrossRef]

[96] Gaglione, D.; Soldi, G.; Braca, P.; De Magistris, G.; Meyer, F.; Hlawatsch, F. Classification-aided multitarget tracking using the sum-product algorithm. IEEE Signal Process. Lett. 2020, 27, 1710–1714. [CrossRef]

[97] MATLAB. Version 9.9.0.1467703 (R2020b); MATLAB: Natick, MA, USA, 2020.

[98] Rahmathullah, A.S.; García-Fernández, Á.F.; Svensson, L. Generalized optimal sub-pattern assignment metric. In Proceedings of the 20th International Conference on Information Fusion (Fusion), Xi'an, China, 10–13 July 2017; pp. 1–8.

[99] Ma, T.J. Remote sensing detection enhancement. J Big Data 8, 127 (2021). https://doi.org/10.1186/s40537-021-00517-8

[100] Damale, P.U.; Chong, E.K.P.; Ma, T.J. Performance Study of Distance-Weighting Approach with Loopy Sum-Product Algorithm for Multi-Object Tracking in Clutter. Sensors 2021, 21, 2544. https://doi.org/10.3390/s21072544