AFRL-RH-WP-TR-2015-0011

TENTACLE: Multi-Camera Immersive Surveillance System Phase II

Randy Milbert, David Hemphill, Justin Benjamin, and Andreas Robinson

Primordial, Inc.
1021 Bandana Boulevard East
Saint Paul MN 55108

April 2015
FINAL REPORT

THIS IS A SMALL BUSINESS INNOVATION RESEARCH (SBIR) PHASE II REPORT.

AIR FORCE RESEARCH LABORATORY
711 HUMAN PERFORMANCE WING
HUMAN EFFECTIVENESS DIRECTORATE
WRIGHT-PATTERSON AIR FORCE BASE OH 45433
AIR FORCE MATERIEL COMMAND
UNITED STATES AIR FORCE

NOTICE AND SIGNATURE PAGE

AFRL-RH-WP-TR-2015-0011 HAS BEEN REVIEWED AND IS APPROVED FOR PUBLICATION IN ACCORDANCE WITH ASSIGNED DISTRIBUTION STATEMENT.

//signed//
DARREL G. HOPPER
Program Manager
Battlespace Visualization Branch

//signed//
JEFFREY L. CRAIG
Chief, Battlespace Visualization Branch
Warfighter Interface Division

//signed//
WILLIAM E. RUSSELL
Chief, Warfighter Interface Division
Human Effectiveness Directorate
711 Human Performance Wing

REPORT DOCUMENTATION PAGE			*Form Approved* *OMB No. 0704-0188*

1. REPORT DATE ***(DD-MM-YY)*** 16-04-15	**2. REPORT TYPE** Final	**3. DATES COVERED** ***(From - To)*** 25 June 2012 – 31 March 2015

4. TITLE AND SUBTITLE TENTACLE: Multi-Camera Immersive Surveillance System Phase II	**5a. CONTRACT NUMBER** FA8650-12-C-6302
	5b. GRANT NUMBER
	5c. PROGRAM ELEMENT NUMBER 65502F
6. AUTHOR(S) Randy Milbert, David Hemphill, Justin Benjamin, and Andreas Robinson	**5d. PROJECT NUMBER** 3005
	5e. TASK NUMBER CV
	5f. WORK UNIT NUMBER H0D9 (Historic: 3005CV26)
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Primordial, Inc. 9955 59th Avenue North, Plymouth, MN 55442	**8. PERFORMING ORGANIZATION REPORT NUMBER** N/A
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) Air Force Material Command Air Force Research Laboratory 711 Human Performance Wing, Human Effectiveness Directorate Warfighter Interface Division, Battlespace Visualization Branch Wright-Patterson Air Force Base OH 45433-7022	**10. SPONSORING/MONITORING AGENCY ACRONYM(S)** USAF AFMC 711 HPW/RHCV
	11. SPONSORING/MONITORING AGENCY REPORT NUMBER(S) AFRL-RH-WP-TR-2015-0011

12. DISTRIBUTION/AVAILABILITY STATEMENT

Distribution A: Approved for public release; distribution is unlimited. 88ABW Cleared 05/21/2015; 88ABW-2015-2528.

13. SUPPLEMENTARY NOTES

This is a Small Business Innovative Research (SBIR) Phase II report developed under a SBIR contract for topic AF103-032 Multi-camera real-time Feature Recognition, Extraction, and Tagging Automation (McFRETA). Contains Patentable Information (patent being pursued but not approved).

14. ABSTRACT

Tentacle is a multi-camera immersive surveillance system that facilitates quickly understanding the status of the battlefield by processing input from a variety of sensors in real time and providing meaningful results to an operator. Tentacle processes this data using operator-specified rule-based alerts to focus incoming data on events meaningful to the user.

15. SUBJECT TERMS realtime video surveillance, people and vehicle tracking, multi-camera multi-sensor fusion algorithms, automated identification and tagging, animated graphical interface

16. SECURITY CLASSIFICATION OF:			**17. LIMITATION OF ABSTRACT:**	**18. NUMBER OF PAGES**	**19a. NAME OF RESPONSIBLE PERSON** (Monitor)
a. REPORT	**b. ABSTRACT**	**c. THIS PAGE**			Darrel G. Hopper
Unclassified	Unclassified	Unclassified	SAR	166	**19b. TELEPHONE NUMBER** *(Include Area Code)*

Standard Form 298 (Rev. 8-98)
Prescribed by ANSI Std. Z39-18

PRIMORDIAL

PRIMORDIAL.COM

MEMORANDUM FOR RECORD

FROM: Primordial, Inc., 9955 59th Avenue North, Plymouth, MN 55442

SUBJECT: Agreement and Approval for Public Release of SBIR Data Rights (FA8650-12-C-6302)

1. Primordial Inc. waives its SBIR Data Rights for the attached technical report entitled "TENTACLE: Multi-Camera Immersive Surveillance System Phase II" for SBIR Phase II contract FA8650-12-C-6302. The Government is granted an unlimited nonexclusive license to use, modify, reproduce, release, perform, display or disclose this chart set and the data contained therein.

2. The attached has been reviewed and is approved for public release, distribution unlimited. The following request is published in the interest of scientific and technical information exchange and does not constitute Government approval or disapproval of the ideas or findings.

Randy L. Milbert — April 17, 2015

Randy L. Milbert, Director, Software — Date
Primordial, Inc.

HOPPER.DARREL.G.1231147264 Digitally signed by HOPPER.DARREL.G.1231147264
DN: c=US, o=U.S. Government, ou=DoD, ou=PKI, ou=USAF, cn=HOPPER.DARREL.G.1231147264
Date: 2015.04.17 12:04:15 -04'00'

Darrel G. Hopper — Date
Principal Electronics Engineer
711 HPW/RHCV

jeffrey.craig.9@us.af.mil Digitally signed by jeffrey.craig.9@us.af.mil
DN: cn=jeffrey.craig.9@us.af.mil
Date: 2015.04.17 14:38:36 -04'00'

(Branch Chief Title/Signature) — Date

(Wing STINFO Title/PA Submitter/Signature) — Date

9955 59th Avenue North • Plymouth, MN • 55442
Phone: (763) 519-1778 • Fax: (651) 644-1294

TABLE OF CONTENTS

Section **Page**

LIST OF FIGURES

LIST OF TABLES

FOREWORD

The PE65502F $749,570.00 SBIR Phase II contract FA8650-12-C-6302, Air Force Research Laboratory (AFRL) Battlespace Visualization Branch (RHCV) Workunit H0D9 (3005CV26), was awarded to Primordial, Inc. on 25 Jun 2012 and ended 31 Mar 2015.

The Phase II effort reported herein was a follow-on to the Phase I contract FA8650-11-M-6193 (AFRL WU 3005CV09) awarded to Primordial, Inc. on 7 March 2011 with an end date of 5 December 2011. The Phase I Final Report has been published to DTIC; the citation is:

> Justin W. Benjamin, Benjamin L. Post, Kyle K. Estes, Randy L Milbert,
> *TENTACLE: Multi-Camera Immersive Surveillance System,*"
> AFRL-RH-WP-TR-2012-0005, Wright-Patterson AFB: AFRL Human Effectiveness Directorate, 79 pp (December 2011). Distribution A.

These Primordial two efforts were awarded under the SBIR Topic "AF103-032 Multi-camera real-time Feature Recognition, Extraction, and Tagging Automation (McFRETA)" program.

The objective of the McFRETA program is to develop an open and scalable framework/tool to perform automated feature recognition of multiple streaming sources and to extract metadata and make available for both ongoing operations and forensics.

Sub-objectives are:

(a) identify algorithms for feature recognition and extraction suitable for real time application;

(b) identify suitable metadata tags that allow for human and machine devices search criteria;

(c) devise a framework that would function in orchestration and event processing frameworks;

(d) design a prototype system.

The ultimate objective is automated identification, tagging, and tracking of humans and vehicles from multiple real time video feeds. A framework and demonstration of how existing and new algorithms can be incorporated and tested is also sought, and of how an operator can develop queries and rules that assist assessment and execution. Scalability from tactical to regional areas of interest is required.

The tool sought under the McFRETA program comprises definition of an open framework for integration of real-time feature recognition and extraction algorithms, generation of a stream of standardized metadata associated with the content source, and design and demonstration of an open, scalable system that supports queries and event/alert notification based on rule sets. Due to the heterogeneous nature of the content capture and storage systems as well as the operations (or forensics) systems, the integrating framework must be open and user friendly so as to enable queries in a broad manner.

1.0 SUMMARY

The following is a summary of the work completed during phase II of the contract.

During the first reporting period (June and July of 2012), the following tasks were completed. The team members conducted a phase II kickoff meeting with AFRL (Air Force Research Laboratory), as well as individual kickoff telecons for each of the three partners: IntuVision, Carnegie Mellon University (CMU) and All Hazards Management. Additionally, IntuVision developed a whitepaper to address lighting changes and color model problems in video. Internally, Primordial transitioned leadership role of the project to Andreas Robinson.

During the second reporting period (October 2012), the following tasks were completed. Primordial conducted a demonstration of Tentacle at NASA (National Aeronautics and Space Administration) WorldWind meeting. Primordial also finalized a subcontract with CMU and defined the statement of work for their tasks. Primordial researched and obtained an initial sampling of UAV (Unmanned Aerial Vehicle) video datasets from several sources such as VIVID (Video Verification of Identity), UCF (University of Central Florida), TUM (Technical University Munich) and YouTube. Primordial requested UAV video from higher-quality military sources and provided IntuVision with UAV video samples to improve tracking. Primordial updated Tentacle to output marked up video samples post-tracking and updated Tentacle Server to use a database system to store tracked entity metadata. Primordial updated Tentacle Server to store marked up video clips for tracked entities and associate with stored metadata records. Primordial updated Tentacle to use latest IntuVision SDK (Software Development Kit), which required some code modification. Primordial worked with IntuVision to update camera types to higher resolution and modify XML (Extensible Markup Language) settings to improve tracking and updated Tentacle code to support video streaming across the network rather than solely metadata as before. IntuVision updated the color model in their SDK to LAB (Lightness and A and B color opponent dimension color space model) and added GPU (Graphical Processing Unit) processing support. Primordial researched and purchased a high-resolution test camera (AXIS Q1755) to support better tracking/extraction of metadata. Primordial performed analysis of frame rate problems during tracking. Primordial also performed several fixes: fixed graphical issues with entity and map display in the client GUI (Graphical User Interface), issues with 3D (Three-dimensional) projection during tracking and issues with video processing in Tentacle mote application.

During reporting period 3 (November and December 2012), the following tasks were completed. Primordial researched and developed a tool for extracting embedded KLV (Key-Length Value format) metadata from UAV video. They also explored development of in-house UAV-video visual tracking approach with several different tracking algorithm options. Primordial also addressed issues with video processing and graphical arrangement of the client GUI application. Primordial updated post-processing filter code. Primordial developed an initial pass of geolocation based on UAV metadata. Primordial evaluated previously extracted KLV metadata for accuracy. Primordial began work to develop application to generate small sections of KLV metadata for video clips by manually assigning tie points to known real-world locations. Primordial completed an initial implementation of archive search functionality. Primordial provided IntuVision with examples of videos where tracking was suboptimal for further review.

Primordial contacted researchers involved in the VIRAT (Video and Image Retrieval and Analysis Tool) project to discuss issues with KLV metadata accuracy. IntuVision updated their SDK to support ingestion of pre-generated bounding boxes representing tracked entities. They also updated their SDK to include updated color histograms to improve tracking. IntuVision provided Primordial suggestions for camera settings to improve tracking performance. IntuVision also provided a whitepaper on improvements for tracking.

During reporting period 4 (January and February of 2013), the following tasks were completed. Primordial finished developing a manual KLV metadata creation tool. Primordial received and integrated an updated IntuVision SDK into Tentacle. Primordial worked on geolocation and tracking issues inherent in the system and worked with IntuVision to resolve tracking problems. Primordial manually generated a set of KLV-style metadata for test video clips. Primordial set up a dedicated Linux server to support CMU SDK testing. CMU provided a first draft SDK and worked with Primordial to get it up and running with required dependencies. Primordial performed initial testing of CMU SDK. Primordial updated Tentacle to process HD (High Definition) quality video frames. Primordial updated Tentacle to stream video properly across the network. Primordial performed bug fixes related to the core Tentacle server application and client GUI. Primordial created demonstration videos of the Tentacle system to show progress. IntuVision provided whitepaper on tracking improvements including elimination of ghost objects and improved vehicle classification. The team conducted 6 month telecom with AFRL and interested parties to discuss status of project.

In reporting period 5 (March and April of 2013), the following tasks were completed. Primordial worked on networking code and issues related to transmission of data. Primordial fixed issues with accurate avatar display in client. Primordial updated client alert classifications to only fire on relevant real-world scenarios. IntuVision continued work on testing advanced color histograms on Primordial-provided test videos. Primordial researched existing spatial query systems similar to what was desired from Tentacle. Primordial created and integrated a SQL (Structured Query Language) database system for storing tracked events and retrofitted the Tentacle server to use it. Primordial updated the Tentacle server to use rudimentary spatial queries (proximity) when searching archived events. Primordial updated Tentacle client to display KML (Keyhole Markup Language) shapes and to use KML inputs for spatial queries. Primordial performed some experimentation with CMU SDK to become familiar with its functionality and use. IntuVision assisted Primordial in overcoming issues with tracked entity passing between the IntuVision SDK and the Tentacle tracking application. CMU provided an updated SDK to Primordial. IntuVision investigated gender and ethnicity extraction algorithms. IntuVision investigated use of Local Binary Pattern features for entity correlation.

During reporting period 6 (May and June, 2013) the following tasks were completed. Primordial completed initial version of proximity-based spatial queries. Primordial began work on implementing complex query types (hierarchical/boolean queries, loitering). Primordial modified network code for transmission of queries to use XML format for robustness/flexibility. Primordial began working on prototype of enter/exit spatial query. Primordial made modifications to client GUI to support complex query types. Primordial worked on demo video creation. CMU provided updated version of SDK, Primordial updated and tested. IntuVision investigated available face recognition and detection systems for use in Tentacle. Primordial continued work on complex queries. Primordial continued to update networking code for queries, including serialization of queries for transmission. Primordial created demo videos to showcase query system. IntuVision integrated Luxand FaceSDK for face detection/recognition into their SDK. IntuVision updated vehicle classification models. Primordial worked with partners to develop a tentative Gantt chart for project completion expectations.

During reporting period 7 (July and August, 2013), the team completed the following tasks. Primordial continued work on updating the client GUI, networking code, and spatial queries. CMU provided new version of SDK, Primordial conducted testing of it. IntuVision continued integration/testing of face recognition SDK into IntuVision SDK. Primordial continued creating demo videos to display spatial query system. Primordial finalized XML version of network query transmission. Primordial finalized work on implementing and testing first-pass count and enter/exit spatial queries. Primordial began work on containment spatial query. Primordial began work on cluster detection algorithm. Primordial continued work on hierarchical queries. CMU began implementing auto-target detection in their SDK. IntuVision continued working on facial recognition integration and testing. IntuVision began experimenting with UAV object recognition with existing BIM (Biologically Inspired Model) models. Primordial implemented internal tools for producing large quantities of targets for testing system load.

During reporting period 8 (September and October 2013), the team completed the following tasks. Primordial began designing code architecture for live spatial queries. Primordial assessed third-party libraries for performing real-time complex event processing. Primordial began testing query setup with large quantities of flight data to simulate large system load and assessed performance. IntuVision worked on UAV object classification using BIM models. CMU continued work on auto-entity detection in UAV video. Primordial worked on improving spatial query speed by using quad-tree structures for data storage. Primordial replaced its internal distance function for spatial objects with a much more efficient one to improve speed of processing. Primordial updated spatial queries to include altitude (quasi-3D processing). Primordial updated Tentacle to use Nesper engine for event processing. IntuVision added support for facial gender recognition to their SDK. CMU continued work on auto-entity detection in UAV video.

During reporting period 9 (November and December, 2013) the following tasks were completed. Primordial acquired by Polaris, some delay due to these changes. Primordial worked on demos for telecon. The team conducted a telecon with AFRL on project status. Primordial worked on optimizing queries. Primordial continued working on implementing live alerts. IntuVision experimented with improvements to metadata extraction from UAV imagery based on BIM models. CMU continued work on auto-entity detection in UAV video. Primordial continued work on live queries. IntuVision produced a new SDK version containing their recent work, Primordial integrated it into Tentacle detection application. CMU began work on integrating geo-location support into the SDK.

During reporting period 10 (January and February 2014), the team completed the following tasks. Primordial worked on testing IntuVision SDK update and getting facial recognition plugin up and running. CMU released new SDK to Primordial. Primordial obtained and tested CMU SDK. Primordial modified CMU test application to output target bounding box shapes for

IntuVision ingestion. Primordial worked with IntuVision on getting IntuVision SDK working with bounding box ingestion. Primordial spent significant time cataloguing and evaluating the UAV videos they received from AFRL. Primordial wrote scripts to convert AFRL video formats/telemetry into something Tentacle could ingest. Primordial conducted conversions of various test videos from the AFRL UAV video sets. IntuVision continued work on UAV object classification. Primordial provided IntuVision with target bounding boxes for select test videos to assist in testing metadata extraction. Primordial continued work on queries using Nesper. Primordial transfers team lead role to David Hemphill. IntuVision provides new SDK for testing.

During reporting period 11 (March and April, 2014), the following tasks were completed. The team conducted a telecon with AFRL to discuss project status. Primordial worked to redesign Tentacle system to treat live and archived queries the same when doing under-the-hood processing. Primordial created a simulator application to send large quantity test data to the server to examine performance under load. Primordial fixed bugs related to networking. Primordial selected a set of videos to use for the final demonstration. Primordial researched and assessed several IPC (Inter-Process Communication) libraries using test applications to facilitate communication between CMU's test application and Tentacle detection application. Primordial made code changes to support ingestion of CMU SDK output. Primordial did internal assessment to verify work status of all Tentacle applications to support project prioritization. Primordial began implementing Tentacle SDK but decided to halt progress to focus on more core elements of the system. Primordial packaged and tested Tentacle client application on multiple remote machines to simulate expected use. Primordial set up a VM (Virtual Machine) environment for CMU SDK and test application to facilitate final demonstration. Primordial developed internal replacement for Nesper processing engine to facilitate customization and to reduce reliance on third party libraries. Primordial worked on project cleanup and bug fixes to move toward a final version of Tentacle. Primordial continued to work on end-to-end testing of the system and bug fixes. Primordial tested CMU SDK to determine state of auto-entity targeting. Primordial continued to work on integration of CMU SDK/test app and the Tentacle system. IntuVision continued working on metadata extraction from UAV video.

During reporting period 12 (May and June, 2014), the following tasks were completed. The team conducted a telecon with AFRL about the project's status and discussing steps for finalizing the project. The no-cost extension for the project was approved. Primordial internally investigated using drones to create metadata, but were unable to come to a successful resolution. Primordial wrote sample applications to test several IPC libraries (Redis.IO and RabbitMQ) to assist in communicating between CMU SDK and Tentacle system. Primordial modified CMU SDK test app to output alpha-channel images for ingestion into IntuVision SDK. Primordial modified Tentacle detection application to ingest the output from newly updated IntuVision SDK using alpha-channel images. Primordial wrote scripts to assist in generating demo telemetry for demo videos as the telemetry was insufficient for 3D georectification. IntuVision worked on updating SDK to ingest CMU alpha channel target mask images. Primordial worked on significant bug fixes, client upgrades. Primordial worked on verifying the validity of subscribed query alerts. Primordial worked on cleaning up and removing outdated code to support final version of Tentacle. IntuVision continued work on supporting target mask ingestion.

After the last reporting period until the end of the project, the following tasks were completed. In July of 2014, the team conducted another telecon with AFRL about project completion plans, final demo timeline. Primordial worked on more bug fixing/optimization of alert system. Primordial worked on polishing GUI interface to be more user friendly. Primordial conducted internal review of Tentacle project and summarization to support creation of the final report. During August of 2014, Primordial began work on final report. Primordial continued evaluating Tentacle system for bugs, and worked on optimization. Primordial began preparing resources for final demonstration. Primordial ceases new improvements on Tentacle to prepare for final demonstration. In October of 2014, Primordial distributed the draft final report. Primordial also gave the final demonstration at AFRL, and delivered final version of Tentacle system and user's guide. In February 2015, Primordial and partners worked on revising the final report based on comments from AFRL. In March 2015, the team delivered the final report to AFRL.

2.0 INTRODUCTION

2.1 Objectives of the McFRETA Program

The objective of the Multi-camera real-time Feature Recognition, Extraction & Tagging Automation (McFRETA) program is to (a) identify algorithms for feature recognition and extraction suitable for real time application,(b) identify suitable metadata tags that allow for human and machine devices search criteria, (c) devise a framework that would function in orchestration and event processing frameworks, and (d) design a prototype system. The ultimate objective is automated identification, tagging, and tracking of humans and vehicles from multiple real time video feeds. A framework and demonstration of how existing and new algorithms can be incorporated and tested is also sought, and of how an operator can develop queries and rules that assist assessment and execution. Scalability from tactical to regional areas of interest is required. The tool sought comprises definition of an open framework for integration of real-time feature recognition and extraction algorithms, generation of a stream of standardized metadata associated with the content source, and design and demonstration of an open, scalable system that supports queries and event/alert notification based on rule sets. Due to the heterogeneous nature of the content capture and storage systems as well as the operations (or forensics) systems, the integrating framework must be open and user-friendly so as to enable queries in a broad manner (1).

2.2 Primordial Team Approach to Tool Sought under McFRETA Program

The technological capacity to gather raw sensor data has far exceeded our capacity to exploit and understand it. There are neither enough people available nor intelligent computer algorithms developed to process all incoming sensor data in real-time. To address this problem, Team Primordial (Primordial plus partners Panoptic, IntuVision, and Carnegie Mellon University) proposes Tentacle, a multi-camera immersive surveillance system. Simply put, Tentacle is software that facilitates quickly understanding what is happening now and in the past. For its eyes, Tentacle uses the ever-growing number of battlefield sensors. Tentacle fuses sensor feeds into a 3D (three-dimensional) virtual world comprised of imagery, terrain, and avatars representing people and vehicles. Tentacle works by integrating real-time feature extraction and tagging algorithms for detecting, tracking, and identifying entities with rule-based database query and dynamic alert tools for de-cluttering and highlighting relevant events. In phase I, Primordial demonstrated Tentacle's ability to display a 3D world mimicking reality while supporting queries and alerts. In phase II, Primordial will update Tentacle by improving its static tracking, entity correlation, metadata extraction, and behavior analysis algorithms. We will also add support for tracking entities from moving platforms (e.g. UAVs) and performing spatial queries.

2.3 Primordial Team Members

2.3.1 Background on Primordial, Polaris, and Their Pre-McFRETA Tentacle Software.

Primordial is a geographic information system (GIS) software company established in 2002 and based in Minnesota. Primordial's flagship products are Ground Guidance off-road route planning software and Ooze crowdsourcing GIS platform. Primordial has secured more than $14 million in contracts from the United States Army, United States Air Force (USAF), Missile Defense Agency (MDA), United States Marine Corps (USMC), Defense Advanced Research Projects Agency (DARPA), Rockwell Collins, General Dynamics, Lockheed Martin, Raytheon, Oshkosh, Nokia, Polaris, and others. In 2012, the United States Army acquired a government-wide license for Ground Guidance. The Army also funded Primordial to integrate and field Ground Guidance with FalconView, Nett Warrior, Tactical Ground Reporting System (TIGR), Joint Battle Command-Platform (JBC-P). Primordial has seven issued and two pending patents.

During the 1980s and 1990s, the US military's rising demand for Polaris off-road vehicles led to Polaris being the first ATV OEM to produce militarized vehicles for SOCOM and the US Army. The extensive success of Polaris vehicles in-theater led to the official founding of Polaris Defense in 2005 as a way for Polaris to better serve the needs of the nation's government and armed forces. Between 2005 and 2008, Polaris Defense steadily developed a range of vehicles to serve the US and allied forces including the MVRS, MV700 and RZR-SW to meet the increasing demand for light off-road mobility platforms. In 2011-2012, Polaris Defense launched the next generation of ATV and LT-ATVs to meet the demanding needs of the US and international special operations, expeditionary and light infantry forces. Polaris was awarded several contracts from the US and other allied governments for the MRZR (LT-ATV) and MV850 (ATV) vehicles. Polaris' military vehicles have been designed, engineered and assembled in the American Midwest. The people of Polaris bring passion and excellence to everything we do, taking great pride in supplying vehicles to our troops. Polaris Defense continues to work closely with the United States and Allied forces, with Defense products operational in over 100 countries across 7 continents. Backed by the infrastructure and resources of a recognized global leader in the off-road vehicle industry, Polaris Defense employs some of the best designers, engineers and manufacturers to continue to provide breakthrough technology for military and government use. In November 2013, Polaris acquired Primordial Inc.

2.3.2 Background on Panoptic Systems and Its U.S. Distributor, All Hazards Mgmt and their C-Thru software.

Panoptic has designed and begun developing an innovative, patent-pending 3D surveillance display—C-Thru—with the aim of increasing security event detection rates, decreasing response times, and reducing the number of operators required to survey an area. Instead of presenting an operator with dozens of continuously cycling, spatially disjointed camera feeds, C-Thru presents a synthesized 3D environment with translucent structures and avatars representing entities such as people and vehicles detected within the environment (2). **Figure 1** provides screenshots from an interactive 3D demonstrator.

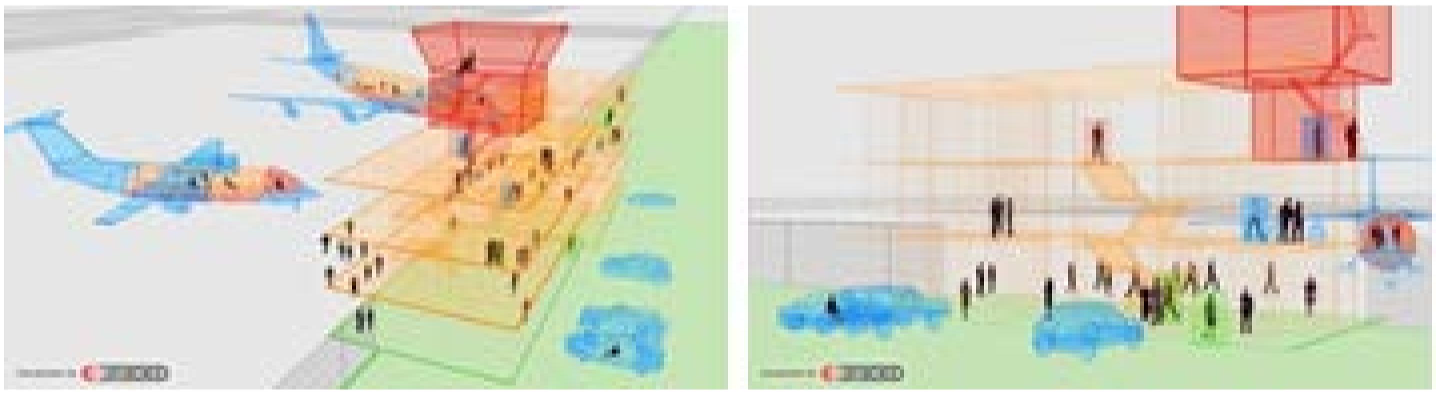


Figure 1. Panoptic C-Thru

All Hazards Management (AHM) delivers innovative Emergency Management Planning, Implementation Support and Tools to government and corporate organizations. AHM is a subsidiary of IDEA Inc. (Integrated Design Engineering Analysis), an industry leader providing strategic and tactical programs and processes for Emergency Management.

2.3.3 Background on IntuVision and Their Panoptes & Video Recall Software.
IntuVision was founded in January 2006 to meet the unique needs of intelligence community for high performance video content extraction solutions. With extensive experience in research, development and deployment of video technologies, Dr. Guler recognized an unmet need for robust yet easily customizable video analytics for both government and commercial domains. She assembled a unique team of creative scientists and engineers who share this vision and created a new paradigm, designing video analysis algorithms to emulate the human visual cognition system, and a new generation of intelligent video was born.
Since its inception IntuVision has been very successful in solving the most challenging video analytics problems and taking the advanced research concepts into working systems for end-users in both government and commercial domains. As a result of both government and industry funded R&D efforts, IntuVision has deployed high performance products for Security, Video Surveillance and Video Investigation applications and the sustainable business strategy earned the company a unique position in the intelligent video arena. IntuVision's real-time video event detection and video forensic investigation technologies have received top ranks in government and organizational evaluations including the first place in Video Event Detection Evaluation run by National Institute for Standards and Technology (NIST TRECVid08).

The IntuVision team has in-depth knowledge and expertise in computer vision complemented by strong software and system development skills to successfully create high performance yet efficient and intuitive products and solutions. The IntuVision team works with small and large clients to develop customized security and surveillance solutions to meet the unique needs of each deployment.

IntuVision develops two widely-fielded, state-of-the-art video surveillance applications:
Panoptes is a real-time video surveillance application that supports cycling through hundreds of video feeds from diverse sensors, performing multi-spectral object tracking, classifying entities

(e.g. as animal, person, or vehicle), detecting dozens of configurable events (e.g. crossing a tripwire or abandoning an object), tracking entities across viewpoints and time (e.g. using soft biometrics and motion analysis), recognizing faces, automatically assigning metadata (e.g. gender and hair color) to tracked entities, and performing real-time queries (e.g. for Asian men wearing a red shirt who recently entered the building). For details, see the 31-page user manual (3).
Video Recall is a forensic video surveillance application that supports analyzing video archives, training entity classifiers, performing queries, generating reports, and sharing results with other analysts. Video Recall is available as a standalone application or i2 Analyst's Notebook plug-in. For details, see the 34-page user manual (4).

2.3.4 Background on Carnegie Mellon University and Their Video Processing Software.
CMU's Robotics Institute is a world leader in the development of autonomous vehicles and has developed over 100 platforms since its inception in 1979. With 400+ faculty, staff and students, it is the largest Robotics research organization in the world. The Robotics Institute has automated both ground and air vehicles for use in commercial, space and military applications. Notable successes include winning the DARPA Urban Challenge, software autonomy to guide the NASA robots (spirit and opportunity) on Mars and significant developments under military programs such as Crusher, Gladiator and Dragon Runner. CMU's past projects iRobot and CMU demonstrated an integrated UAV and UGV (Unmanned Ground Vehicle) system for surveillance missions involving ground target search, tracking, pursuit, and geolocation. Demonstrated with Packbot UGVs and Raven UAVs, the system improved situational awareness and reduced workload. Additionally, CMU and iRobot are extending the demonstrated collaborative UAV/UGV system capability to support larger teams of robots operating in challenging urban environments. CMU is developing a scalable and extensible architecture, improving search/pursuit/tracking capabilities, and addressing target occlusion in urban areas.

2.4 Proposed Prototype Panoptic C-Thru 3D Surveillance Display Based on Tentacle

Specifically, Panoptic Systems will provide a C-Thru software module for use in conjunction with NASA WorldWind. This module will be capable of displaying high-fidelity indoor and outdoor models including buildings and avatars when a Tentacle user zooms in on an area of interest such as a compound or square. Philippe Van Nedervelde, Panoptic's CTO (Chief Technology Officer), and Dr. H. John Caulfield, director of Panoptic's scientific board of advisors, will lead this effort.

2.5 Summary of McFRETA Phase I Accomplishments by the Primordial Team

In Phase I, the Tentacle team developed an in-house tracking system using COTS (Commercial Off The Shelf) IP (Internet Protocol) cameras. The system used a background subtraction mechanism to track entities in the video feed. The system also had the ability to reduce the video stream to simplified images based on Canny edge-detection. The team developed a 3D-projection algorithm to project the locations of entities in the video stream to their location in 3D-space and displayed them on a 3D map in a NASA WorldWind based client application. The team integrated IntuVision's SDK into the system to provide behavioral detection functionality and classification of entities and color extraction. The system also supported multiple cameras and coordination between them using assigned entity IDs (Identification numbers). The team developed a client-server model for the system where a Tentacle mote (sensor) feeds data into the server which then broadcasts information to connected clients.

3.0 METHODS, ASSUMPTIONS, AND PROCEDURES

3.1 Improve Static Camera Tracking

In Phase II, one of the main tasks of the Tentacle project was to improve the quality of the static camera tracking demonstrated in Phase I. **Figure 2** and **Figure 3** show the state of static camera tracking at the end of Phase I.

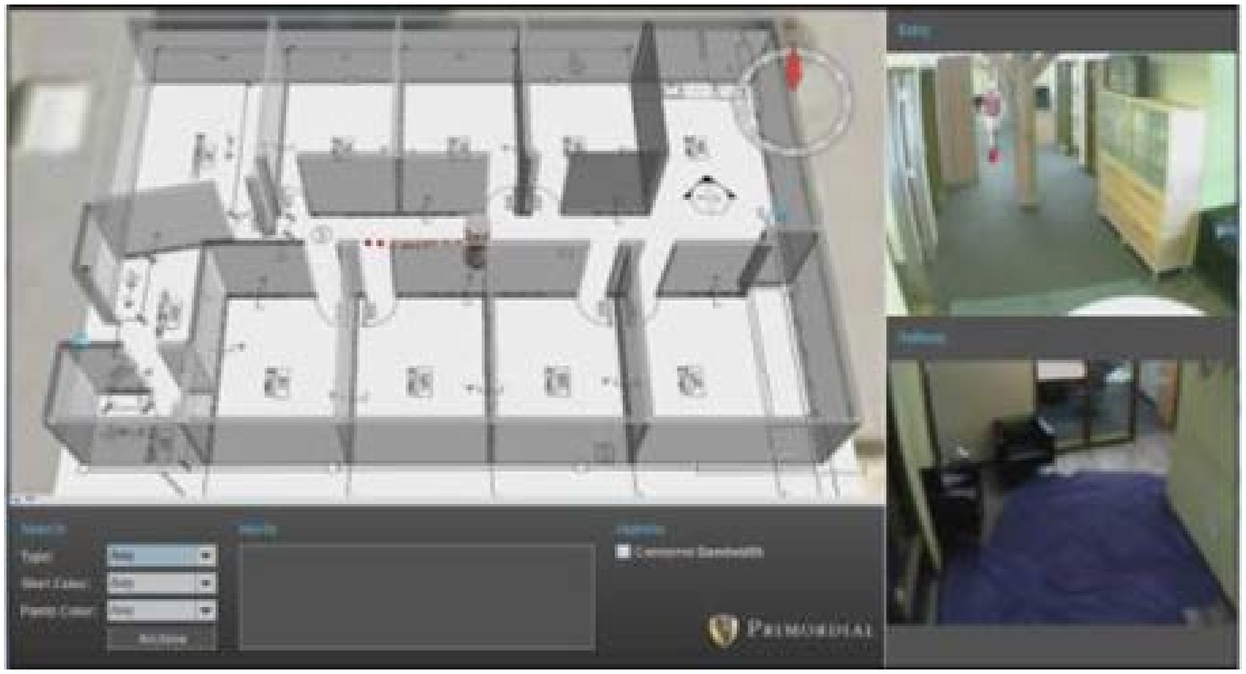


Figure 2. Phase I Demonstration Of Person Tracking Using Fixed Cameras

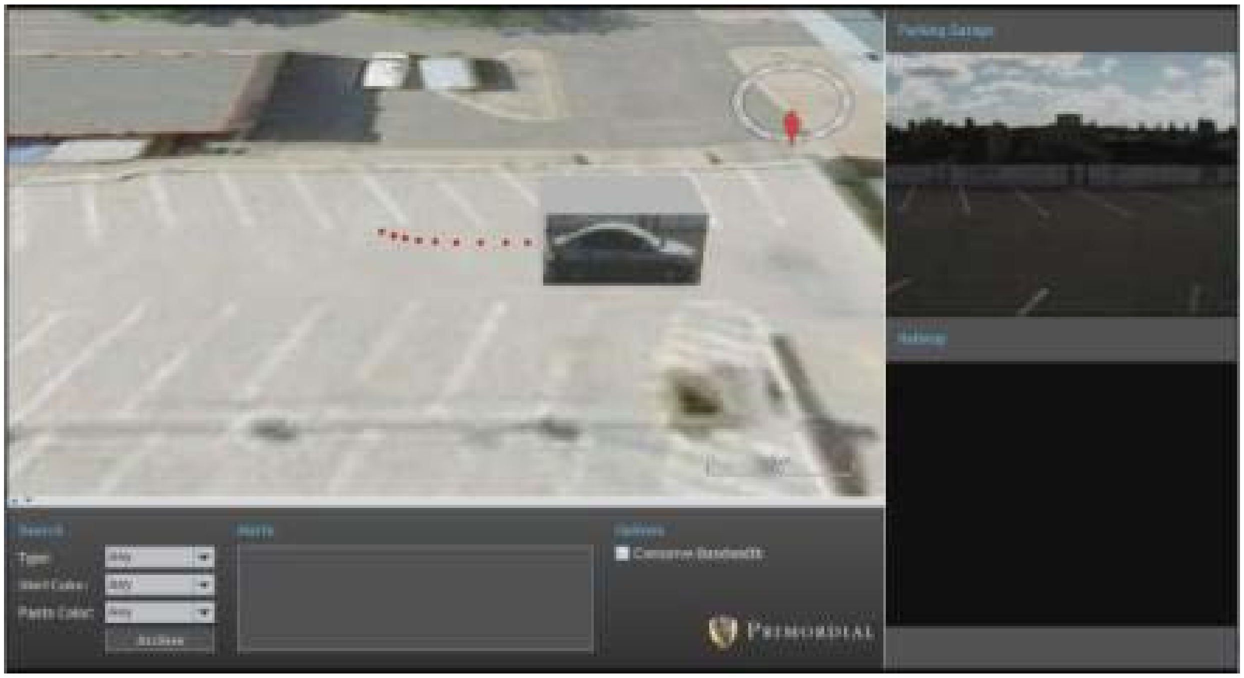


Figure 3. Phase I Demonstration Of Vehicle Tracking Using Fixed Cameras

At the start of Phase II, IntuVision began to work on general tracking improvements. One of the major issues encountered in Phase I was the effects of scene illumination changes during tracking of moving objects. IntuVision proposed the use of a new Lab background model based on the CIEL*a*b* color space which would improve the tracking of objects under various scene illumination changes that may be caused by cloud cover, one of the main problematic scenarios encountered in Phase I.

The default scene background model IntuVision's Panoptes system used relied on the RGB (Red Green Blue) color space to establish the scene background. This approach, while mostly preferred due to the input video formats, has an inherent limitation related to handling of illumination changes as the RGB channels are correlated. These correlations are significantly affected by illumination. As the scene gets brighter or darker, the RGB values increase or decrease across the board. This is illustrated in **Figure 4** below depicting the average grayscale value of a set of frames of a test sequence. The change in illumination over time is due to changes in cloud cover and is quite significant. Thus, for example, as clouds pass by over a scene, parts of the scene may darken and these illumination changes may be large enough to get detected as false objects.

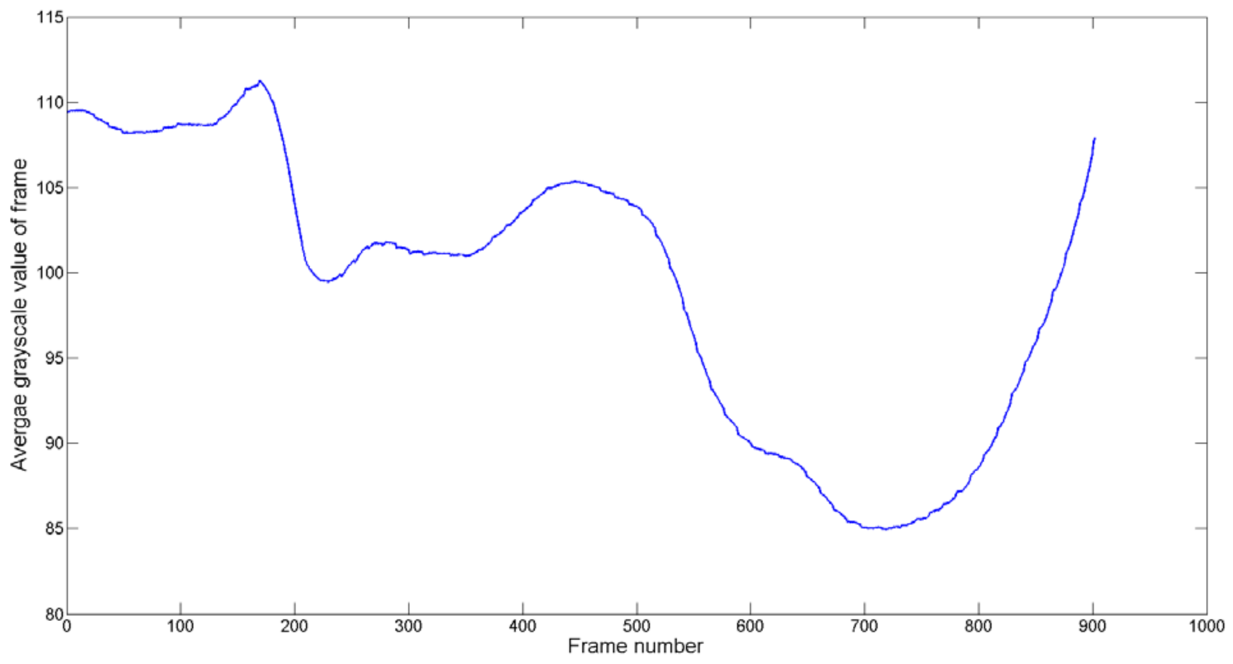


Figure 4. Average Grayscale Value of Test Sequence Over Time

Similarly, when larger objects (large relative to the scene dimensions) move in the scene, they may affect the lightness of parts of the scene. This combined with the illumination changes may result in increased size of the detected object as shown in the left image of **Figure 5**.

One possible solution to this is the use of the CIEL*a*b* color space. This color space approximates the human vision system with the a* and b* channels capturing color information based on the opponent color axes, while L* captures a lightness value thereby de-correlating the illumination and color information. This makes it possible to detect changes in illumination and control the response of the background model.

Figure 5 shows the performance of the traditional RGB background model (left) and that of the CIEL*a*b* based background model (right) for the same test sequence. As can be seen in the images, the CIEL*a*b* model results in much tighter bounding box and is hardly affected by the passing cloud cover.

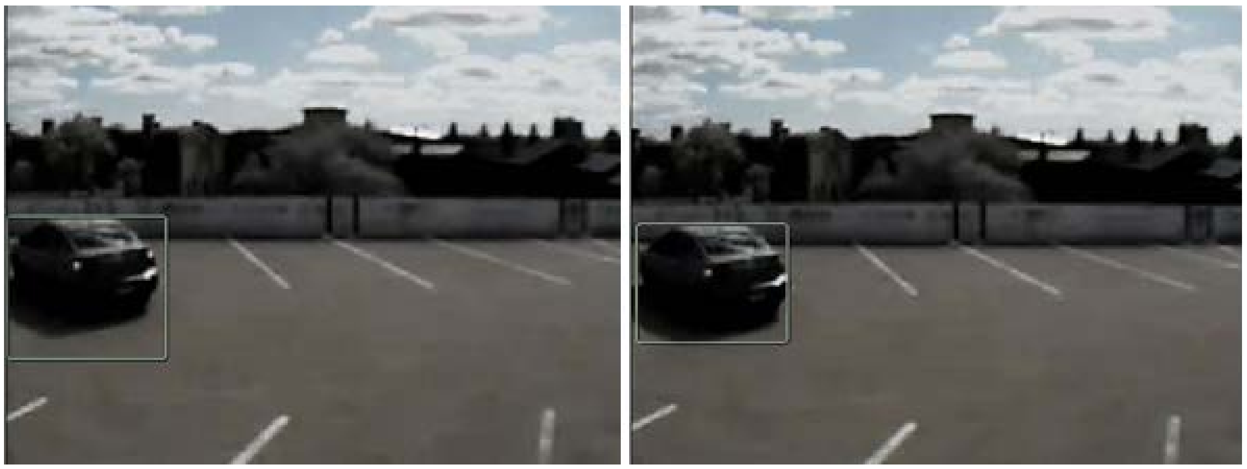


Figure 5. Same Vehicle Tracked Using the RGB Background Model (Left) and Ciel*A*B* Background Model (Right) Results in a Tighter Object Bounding Box Reducing the Effects of Changes in Scene Illumination

To facilitate improving the performance of the tracking system, Primordial provided IntuVision with a set of 15 baseline videos depicting tracking and meta-data errors Primordial had been encountering with the IntuVision SDK. For the purposes of generating these samples the Tentacle detection application code was updated to store output videos depicting the entity bounding boxes as well color-coded meta-data information. After reviewing these baseline videos, IntuVision provided feedback regarding potential modifications to the camera XML configuration file to improve results; they also recommended switching the camera resolution from QVGA (Quarter-Video Graphics Array Resolution, 320x240 pixel) video to VGA (Video Graphics Array Resolution, 640x480 pixels), which our Phase I cameras already supported at a lower frame rate.

Primordial performed initial experimentation based on these suggested adjustments, and preliminary results with newly generated videos appeared to have significantly reduced errors. Based on an initial assessment, the primary drivers of improved tracking results appeared to have been increasing the resolution of the camera feed, which reduced false negatives in entity detection and improved meta-data extraction accuracy, as well as bug fixes to the static object detection in the IntuVision SDK, which had reduced false positives in detecting static objects such as dropped-bags.
Because of these findings and further recommendations from IntuVision, Primordial investigated alternate surveillance camera systems and placed an order for a high-definition (HD) video camera to facilitate testing. In particular, the surveillance camera purchased was an Axis Q1755 model; IntuVision had indicated that their SDK would likely offer much improved performance with this type of high resolution camera.

During testing of the newly modified system, Primordial discovered errors in the 3D projection behavior; it appeared that these were primarily related to regressions that occurred when updating to use VGA video. These errors were then rectified.

Primordial provided a second round of 16 additional baseline videos to IntuVision, along with current tracking results obtained from their SDK to assist in continuing improvements to the static tracking system. These videos were collected using the same camera as the original tests, but at a higher resolution. IntuVision reviewed each of the video clips and Primordial's observations. The color response was still inferior to what would be produced by the Axis camera which IntuVision recommended to Primordial for purchase. As a result, the advanced color histograms had only marginally improved the metadata color representation for those videos in contrast to what would be produced using HD data. IntuVision provided comments and the following suggestions. Adjustment of the minimum object size setting was needed per-camera in order to filter out small reflections and false detections as relevant object sizes vary depending on the camera view. Adjustment of the sensitivity setting was needed per-camera basis in order to achieve better detection rate, as scene colors and contrast may vary depending on the camera view. Turning off Static Object Detection when not doing left object event detection and setting the invalid-window tracking tag to 10-15 seconds was necessary to increase processing efficiency. Performing tests in Release Mode instead of Debug Mode was essential, as testing the performance in Debug Mode can/will degrade detection/tracking accuracy.
In the Primordial baseline video results there were a few examples of objects not tracked at all or delayed in being detected. When reprocessed at IntuVision using the same settings, this was not encountered. **Figure 6** gives an example of one of these problematic samples.

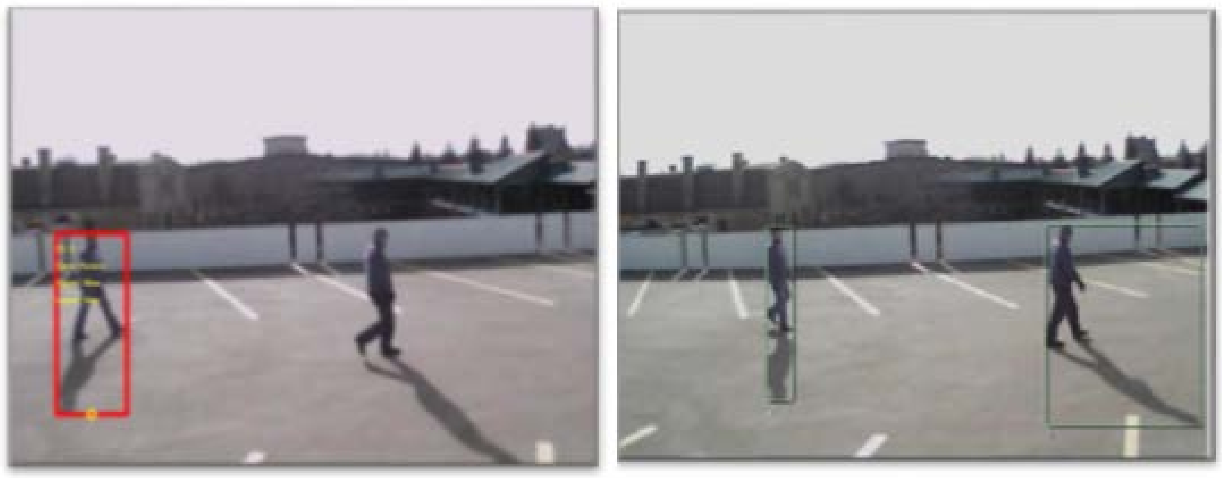


Figure 6. Example of Missed Detection by Primordial [Using IntuVision SDK] (left) And Detection of the Same Object by IntuVision Processing (right) Using Same Settings

IntuVision suggested that this may have something to do with the shadows in the image and that the detected shadows might be somewhat improved by using the Lab/CIEL color background model or object saliency filtering.

Primordial made several modifications to the Tentacle system at this point to address various performance issues. They addressed multi-threading issues in the video analysis code, which were intermittently causing frames to be analyzed out of order. Additionally, they updated the video recording code to automatically save at the measured average frame rate from streaming video, rather than the frame rate value from the camera settings. This resolved some issues in processing related to frame synchronization.

IntuVision continued work on implementing the CIEL*a*b* background model into their Panoptes SDK. The a*channel encoded the color information along the green – magenta axis and the b*channel encoded the color information along the blue – yellow axis. The L* channel captured the luminance and hence it is de-correlated from the color channels. In this implementation the user could choose between the grayscale background model, a traditional RGB background model (**Figure 7**) or the CIEL*a*b* background model with options for CPU (Central Processing Unit) and GPU usage. Since the Lab/CIEL color background model is computationally more expensive, IntuVision also ported the implementation on to the GPU to have both variants available. For a video with 640x480 frame resolution the CPU CIEL*a*b* performs at 11 FPS (Frames Per Second) on a quad core Xeon processor running at 2.67 GHz. For the same video the GPU variant of CIEL*a*b* background model processing speed is 44 FPS on the same processor with an Nvidia GeForce 310 Graphics card with 16 cores. **Figure 8** shows improvements in tracking with the updated CIEL model.

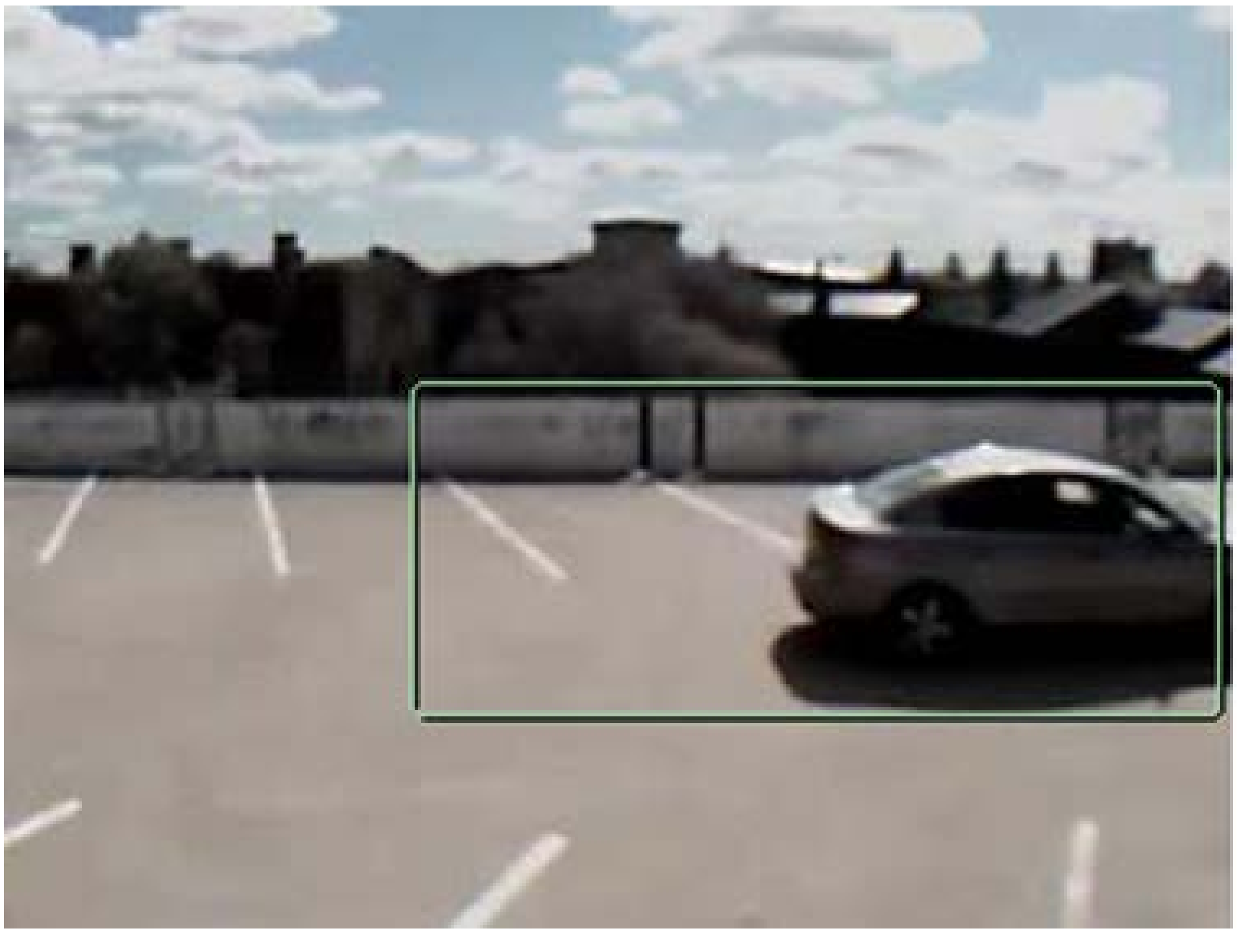


Figure 7. Vehicle Tracked Using the RGB Background Model

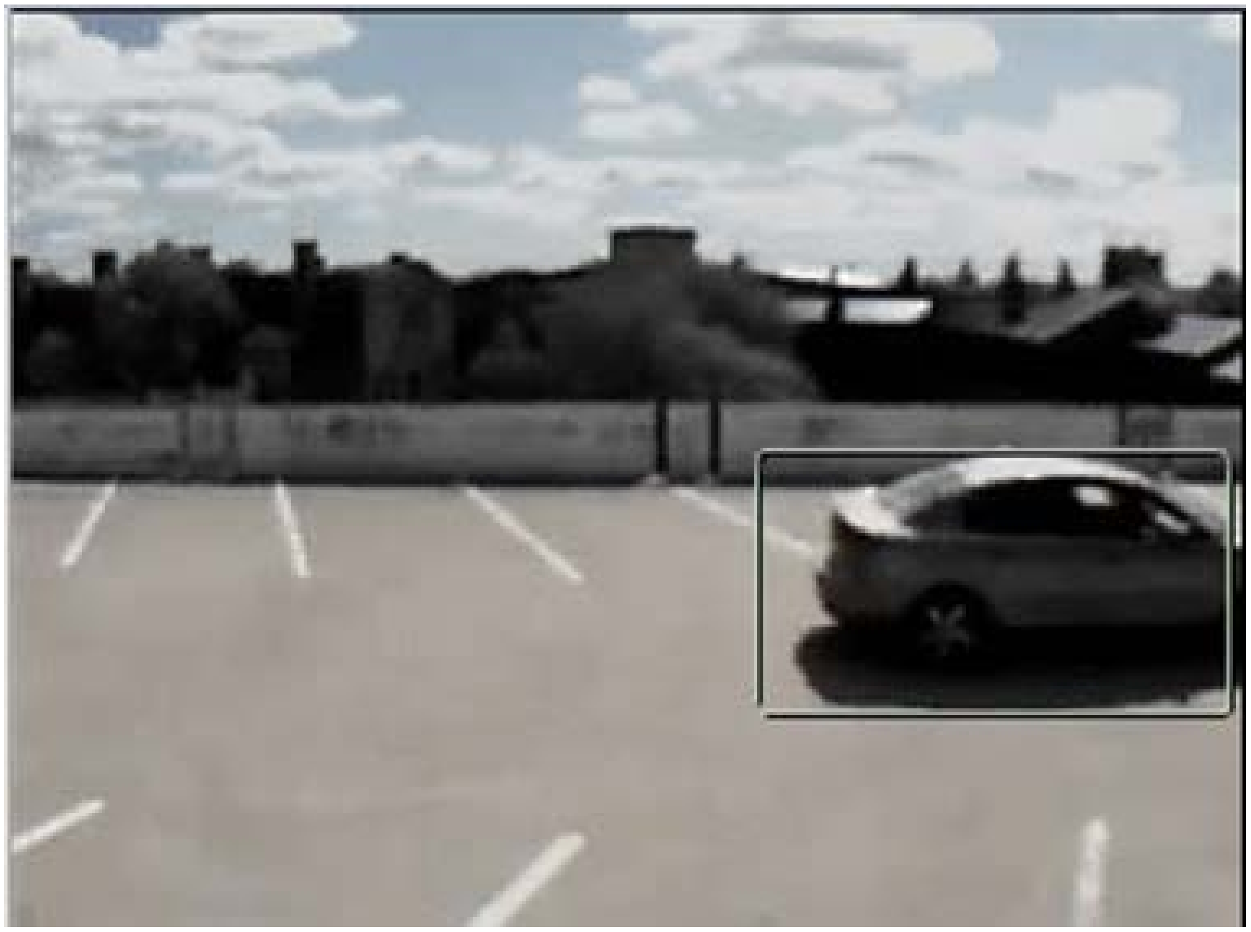


Figure 8. Vehicle Tracked Using Ciel*A*B* Background Model

Another round of video clips collected by the Primordial team were then tested by IntuVision to find optimal settings for detection and tracking of people and vehicles as well as addressing detection of static objects. IntuVision again provided recommended settings and processing options to Primordial to assist in improving the tracking performance. Specifically, the noise filtering options were improved to handle a wide variety of scenes with “auto” settings.

Primordial continued to work on improving its Tentacle post-processing filters, including filtering out “ghost” entities which don’t persist beyond a minimum threshold time period. IntuVision provided guidance with these upgrades. When tracking, it is important to reliably establish the scene background both for object tracking and for detecting high level activities. In the real world it is not possible to view the scene background without any moving (foreground) objects. Moving objects that were in the scene when the processing starts would sometimes leave ‘ghost’ object detections (stationary objects located where the removed object used to be) as illustrated in **Figure 9**.

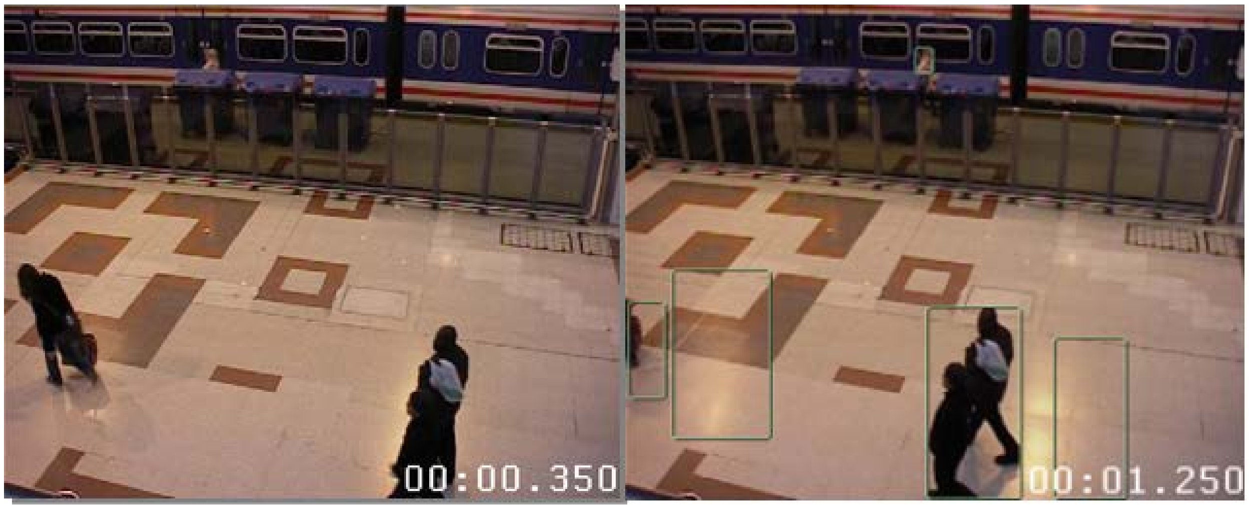


Figure 9. The Video Starts with People in the Scene (Left), as the People Move, 'Ghost' Objects Are Left Where They Were Standing (Right)

IntuVision updated their SDK to improve the tracking algorithm, both in terms of how well it removed 'ghost' objects, and in making sure it did not remove actual stationary objects (see **Figure** 10).

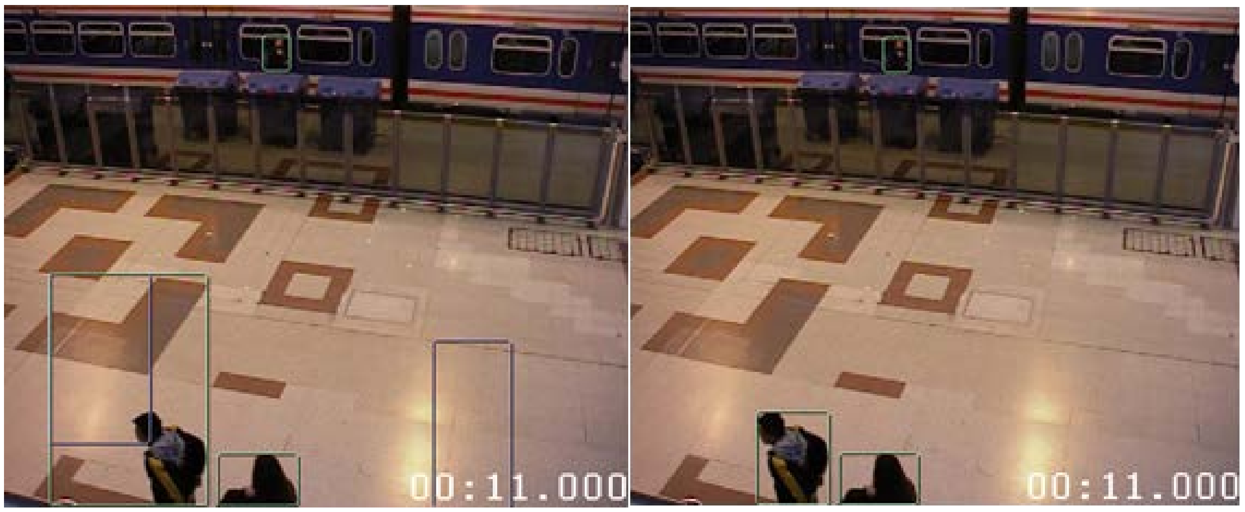


Figure 10. Factors Such as Lighting Changes May Cause These 'Ghost' Objects to Linger and Detected as Idling (Left), the Updated Object Removal Algorithm Reliably Removed These Spurious Detections (Right)

After IntuVision's changes, Primordial updated its code integration with IntuVision's SDK to help address a long-standing issue of intermittent delays in tracking detection, which did not occur in the IntuVision example app. Specifically there was a concurrency bug in the callback method being used by the IntuVision SDK which was overwriting memory with new frames before old frames could be processed. This caused numerous strange bugs as well as the aforementioned delay in tracking. The code was modified to prevent this race condition and it significantly reduced processing delay and eliminated the transient errors that had been occurring.

In order to better scope the project, IntuVision limited its SDK client alerts to significant events only, specifically “trip-wire” and “dropped-bag”, as opposed to frequent occurrences such as motion detection to help improve the overall tracking performance.
Primordial received the Axis Q1755 camera which had been ordered previously and which had been recommended by IntuVision. They updated the Tentacle detection application code to ingest the much larger HD (1920x1080) video frames. This step was intended to help increase the accuracy of the tracking system, as larger objects on the screen provide more data for the tracking algorithm to work with. At this stage, Primordial provided another round of baseline test videos to IntuVision to help guide adjustments to their SDK.

Once the daylight illumination, ghost object and tracking setting issues had been resolved, IntuVision focused their efforts for status camera tracking on reducing glare reduction (specifically vehicle headlights at night). In the figures below, the headlights of a vehicle are seen prior to the vehicle entering the camera view. The vehicle headlight illuminates the scene and the head light beam gets captured as a standalone object or as a large extension of the vehicle in traditional video analytics solutions. The vehicle may never be seen by the camera but the headlight beam of a passing car may get detected triggering false alarms. As the headlight beam usually lights up a large area, it can also cause multiple objects to get merged thereby reducing the quality of detections and tracking. IntuVision’s SDK provided an intelligent headlight glare removal option that had potential to mitigate such issues. The two sets of images below demonstrate the headlight glare removal in action.

Figure 11 shows a camera view that has been illuminated by the headlight of an approaching vehicle. The scene pixels illuminated by the headlight beam get a high probability of belonging to the foreground based on the learnt background model. The headlight glare removal component correctly identifies the illuminated pixels as belonging to the headlight and removes them from the final processed image. (Top left) A vehicle headlight beam illuminates the camera view. (Top right) As the headlight beam is strong, a large number of pixels get a high probability of belonging to the foreground based on the learnt background model. (Bottom left) The detected headlight beam pixels. (Bottom right) The final processed image devoid of any headlight beam false positives.

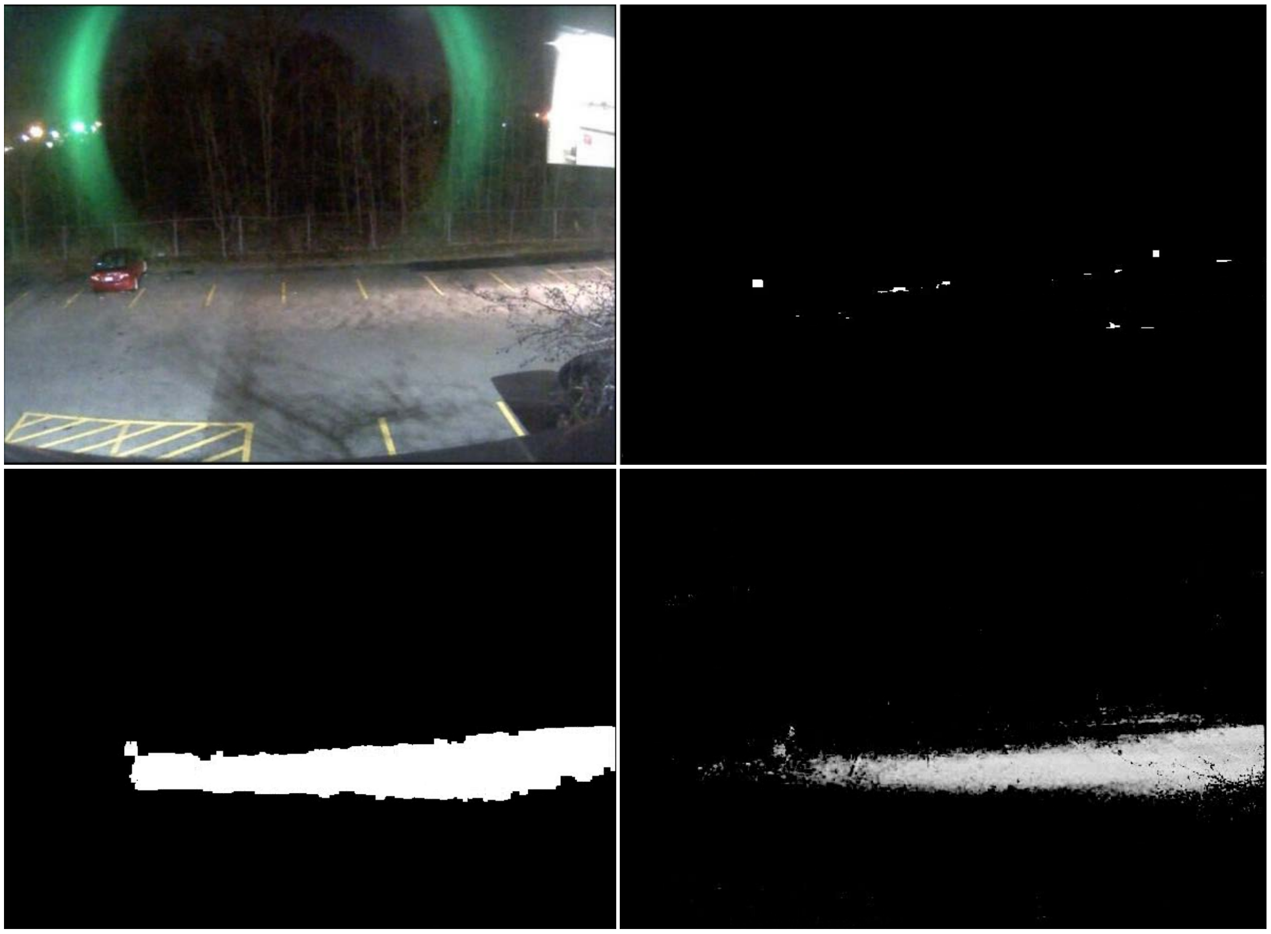


Figure 11. Headlights

Later the vehicle enters the camera view (**Figure 12**). As can be seen in the images the vehicle gets correctly detected as a foreground object while the pixels illuminated by the headlight get removed from the final processed image. (Top left) A vehicle, with the headlight turned on, enters the camera view. Note that only the vehicle is detected (with a bounding box) as a foreground object. (Top right) As the pixels illuminated by the headlight beam and the vehicle pixels get a high probability of belonging to the foreground based on the learnt background model. (Bottom left) The detected headlight beam pixels. (Bottom right) The final processed image devoid of any headlight beam false positives, the vehicle gets correctly detected. These changes resulted in better vehicle detection without headlights causing false positive reports.

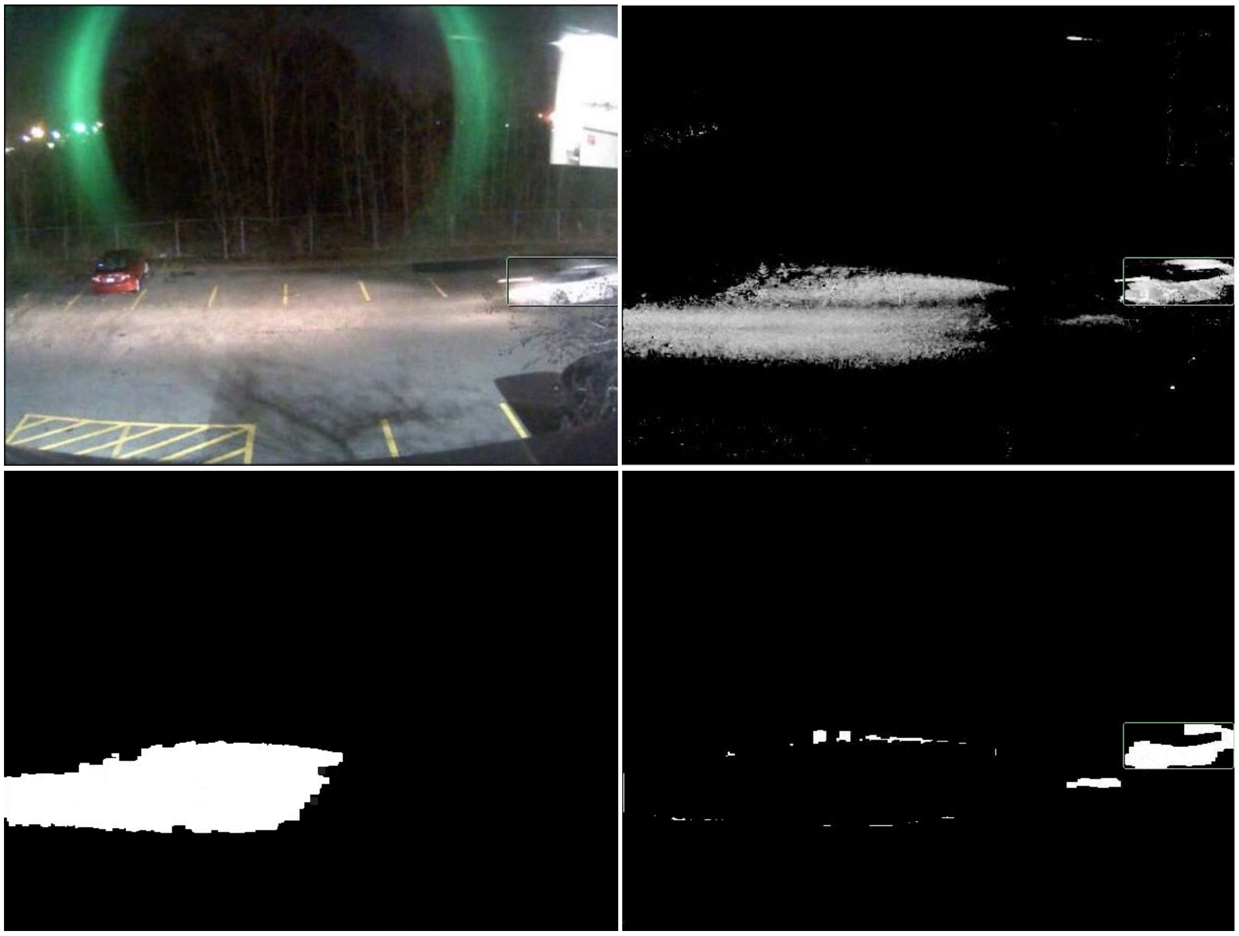
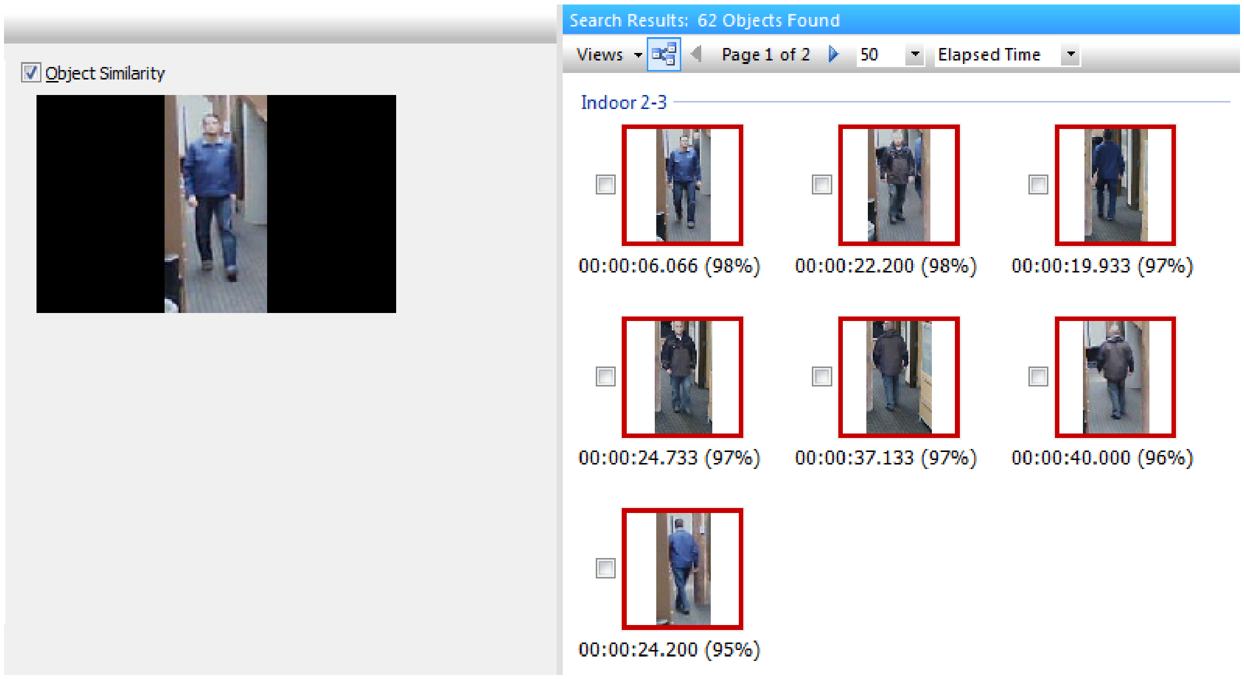


Figure 12. Headlight Removal

3.2 Improve Entity Correlation

During Phase II, the Tentacle team worked to improve entity correlation in the Tentacle system to provide more consistent representation of detected entities internally.
In an effort to improve on the Phase I system, IntuVision examined a technique for extracting features from video called Local Binary Pattern (LBP).

LBP captures the distribution of pixel intensities around each pixel in an image. This feature is made rotationally invariant via mapping to a 60 bin histogram. LBP does a very good job of capturing and comparing the texture similarity. IntuVision had used LBP in the past for scene classification and object similarity modules, though never incorporated it into their SDK. IntuVision tested the initial baseline videos that Primordial had provided at the beginning of phase II with the existing LBP module to determine the feature's usefulness in this project. IntuVision found that for this application, although LBP provides descriptive properties, it did not provide sufficiently discriminatory features. In most test cases, LBP was able to correlate different instances (videos) of the same person within a single camera view. Unfortunately, these features were not able to correlate well between views, or when the person was in a different pose. Additionally, there was very little separation in similarity confidence between different people in the same pose. **Figure 13** and **Figure 14** show examples of this testing. The best match is the same person from another video shot with the same camera. Other people at the same location and pose also show similar confidences.



00:00:06.066 (98%)
00:00:22.200 (98%)
00:00:19.933 (97%)
00:00:24.733 (97%)
00:00:37.133 (97%)
00:00:40.000 (96%)
00:00:24.200 (95%)

Figure 13. Local Binary Pattern

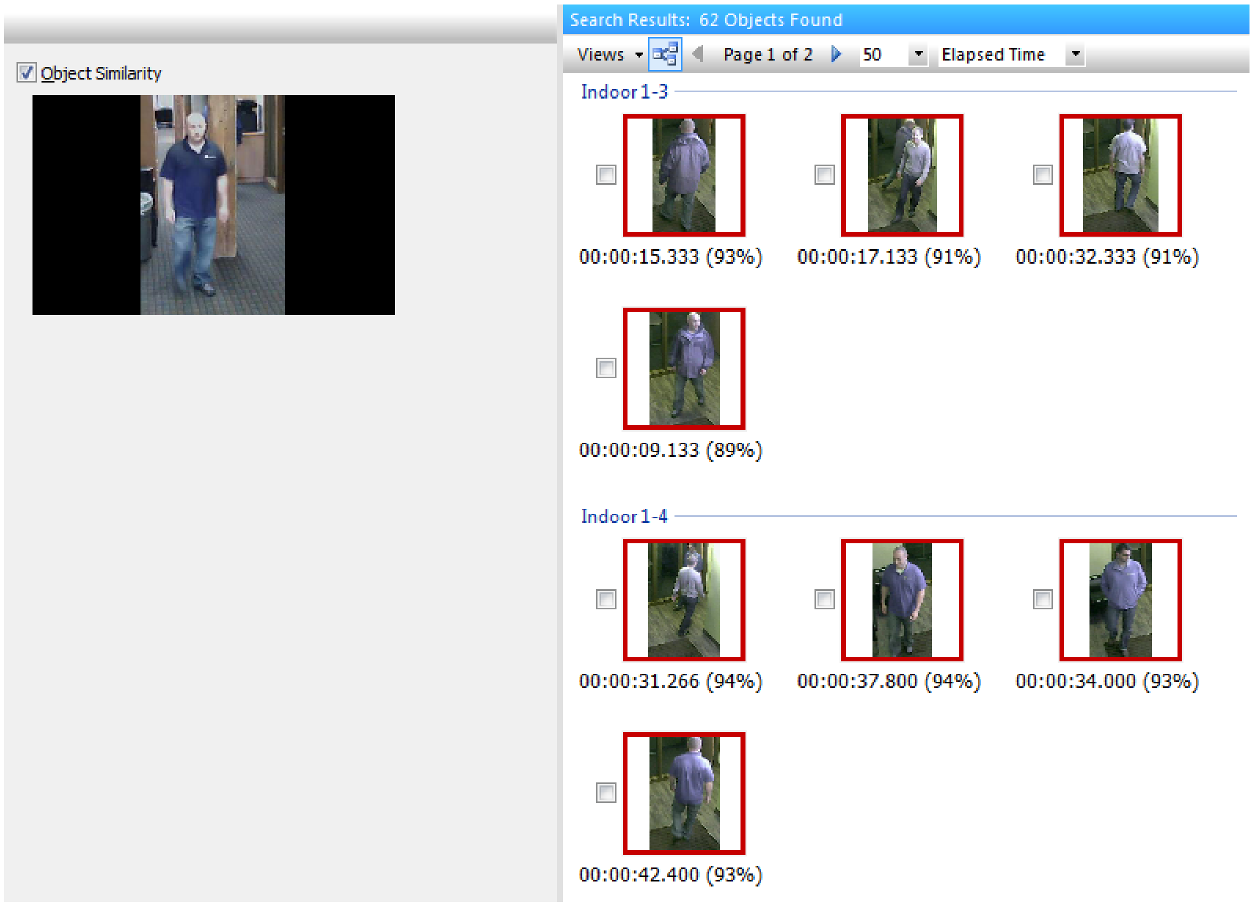


Figure 14. Local Binary Pattern

LBP doesn't do as well at matching objects between different camera views. Due to these factors, the team decided that LBP did not perform sufficiently to be a useful feature for the cross-camera correlation required by this project.

Primordial continued to work on its Phase I version of entity correlation by updating the Tentacle client GUI to only display avatars for entities in World Wind after they have been tracked over multiple frames, to avoid false positive avatar depictions. This behaved as a rudimentary filter for briefly detected objects which tended to often be erroneous.

After the initial foray into LBP, IntuVision continued to analyze the baseline videos provided by Primordial at the beginning of Phase II. The baseline videos from Primordial did not contain broad color diversity (blue/grey shirts and pants) and the colors were all very bland and washed out. Additionally, the camera resolutions were less than optimal for video analysis. Under these conditions, however, the original histograms performed reasonably well. IntuVision worked on further improving the color accuracy by use of advanced color histograms to better handle these circumstances. Advanced color histograms use 512 color bins (8 bins per color channel), rather than the standard 64. Hence, they provide a more detailed and accurate representation of an object's colors. One example of the improvements that followed from these changes is that the advanced histograms can more accurately represent the color of blue jeans, as seen in the following example in **Figure 15**. The advanced histogram better captures the color of the man's blue jeans.

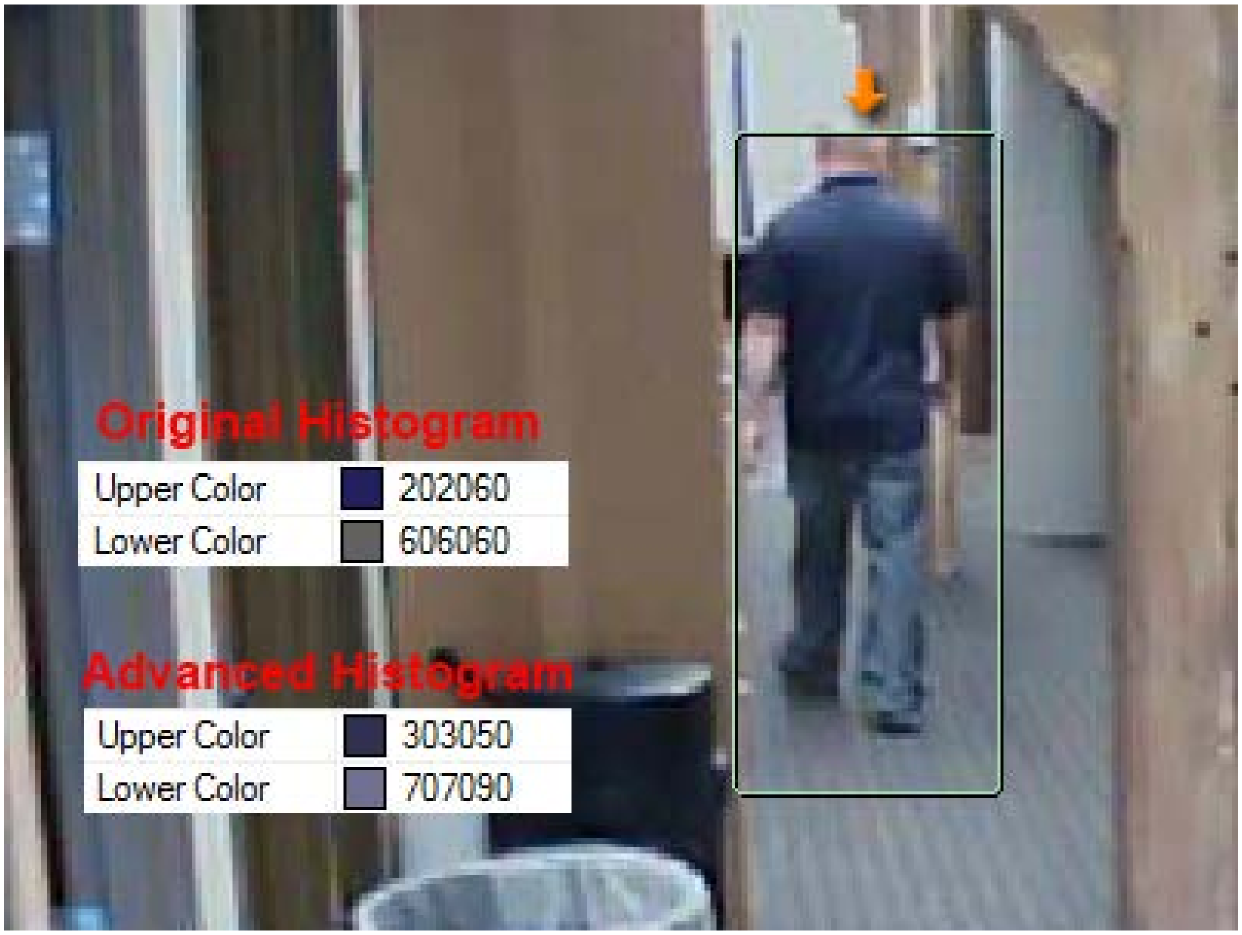


Figure 15. Advanced Histograms

IntuVision was not able to find any examples in the baseline videos to showcase the secondary color feature (which can be used to capture the colors of a backpack or second shirt). IntuVision also worked on improving vehicle classification to better identify various vehicle types to assist in identifying objects across camera views. Often, different camera views will observe an object in different poses and the object pose variability decreases the classification performance.

Traditional object classification model creation uses all possible positive data for generating the model. This strategy works well when then there is limited variability in the object poses. If different object pose images are added to the training dataset to accommodate different camera views, the intra-class variance increases. This is usually associated with a drop in the classification performance as the classifier tries to learn the different appearances of the object in a single model. Because of this, some object categories deserve the generation of multiple models to account for different poses of the same object in different views. Classifying object A with pose 1 or with pose 2 does not affect the final result of classifying the object as of type A, as the different object poses need not be used for testing against other pose models. IntuVision decided to investigate this approach for use in this project. **Figure 16** shows a good example of object pose variability with the front and back views of an auto-rickshaw, besides a glimpse of the side of the vehicle; the two views of the vehicle are drastically different. (Top) Front view of an auto-rickshaw. (Bottom) Rear view of an auto-rickshaw.

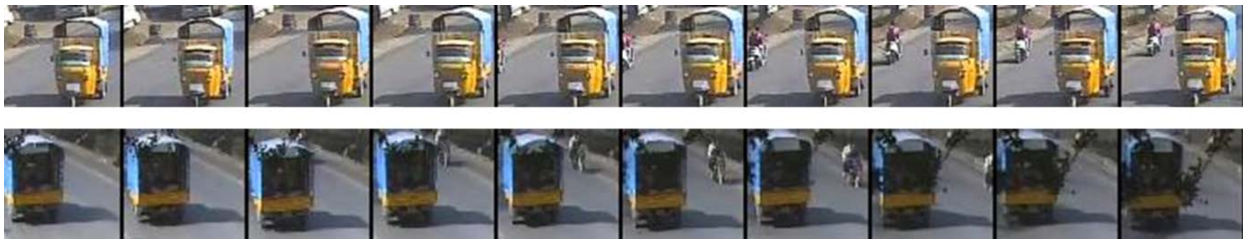


Figure 16. Different Viewpoints of the Same Object Contributes to Intra Class Variance

In the auto-rickshaw example, the front views are used only in the training and testing of the front view model and are not used to test the rear view model. **Figure 17** provides a comparison of the performance of a classifier trained to classify auto-rickshaws on all positive data at once for a single model and that of the separate models. Single model for all the views and separate models for each view. MD% is missed detection rate and FA% is false alarm rate. As can be seen from the figure creating separate models improves the classification performance. For the model training and testing, the negative examples belonged to the object categories of people, road background, bus, cars, bikes and trucks.

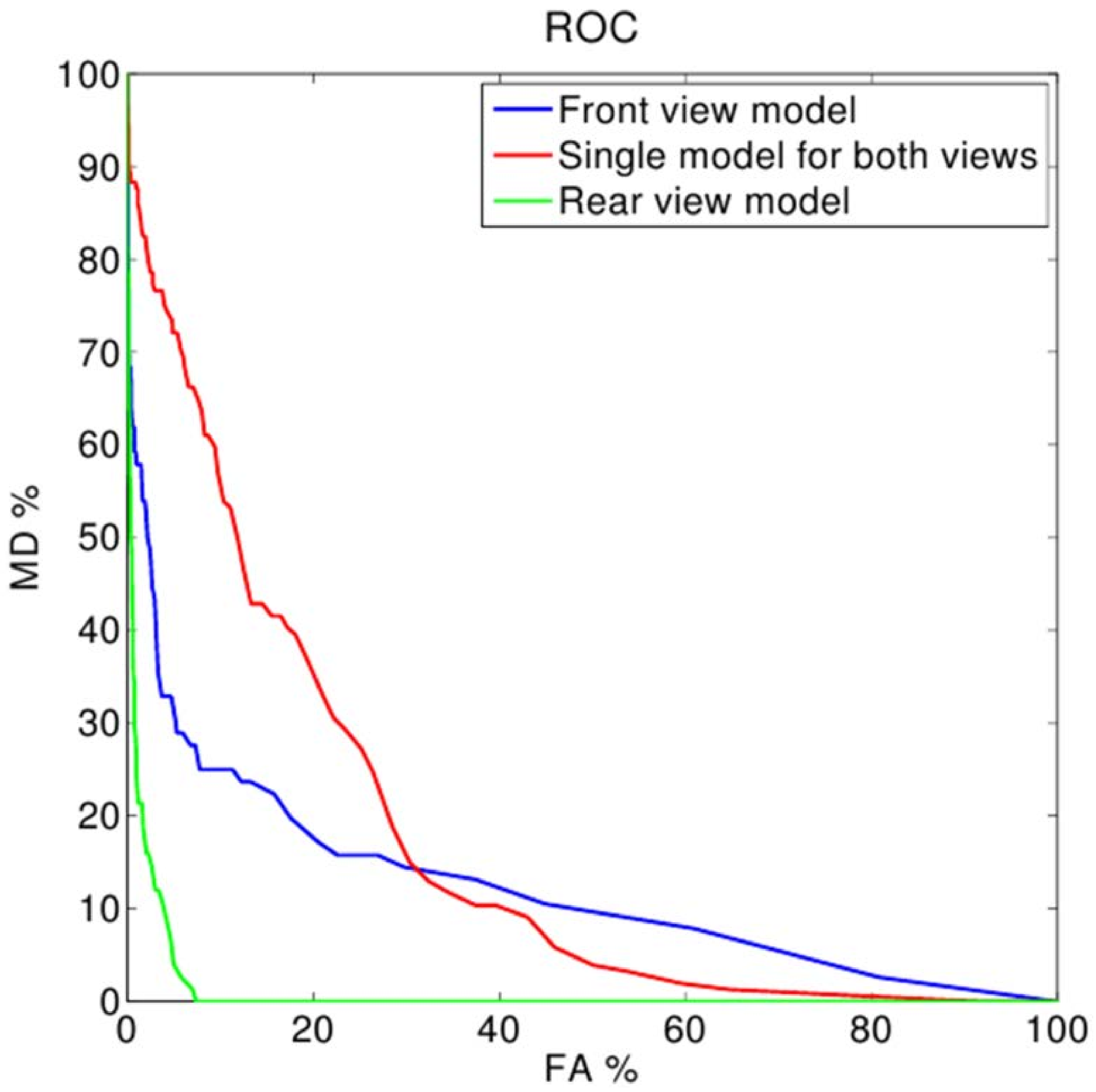
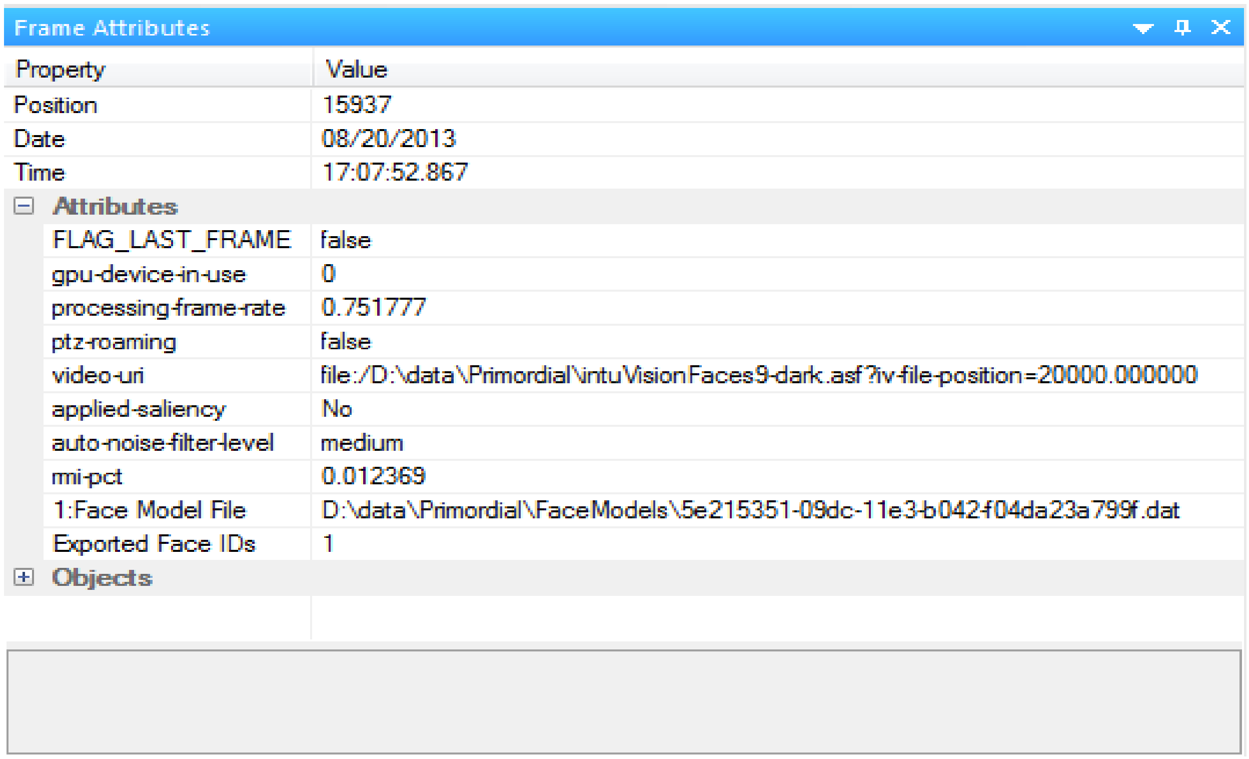


Figure 17. Performance of Different Approaches for Model Creation

In addition to improving the classification for multiple views, IntuVision also worked on cross-camera face matching using the help of the Luxand Face SDK. The Luxand Face SDK facilitates facial detection and matching by creating and comparing templates based on a single face image. These templates are rather large, each occupying 16kb of memory. The IntuVision SDK constructed a more general face model by recording the five most common (but diverse) templates. The resultant model was too large (80kb) to transfer directly out of the SDK via the normal callback methods as was done with text-based classification and color information for other tracked objects. Because of this, IntuVision developed a new scheme for handling cameras where face matching models are extracted. In this scheme, a directory could be supplied to the

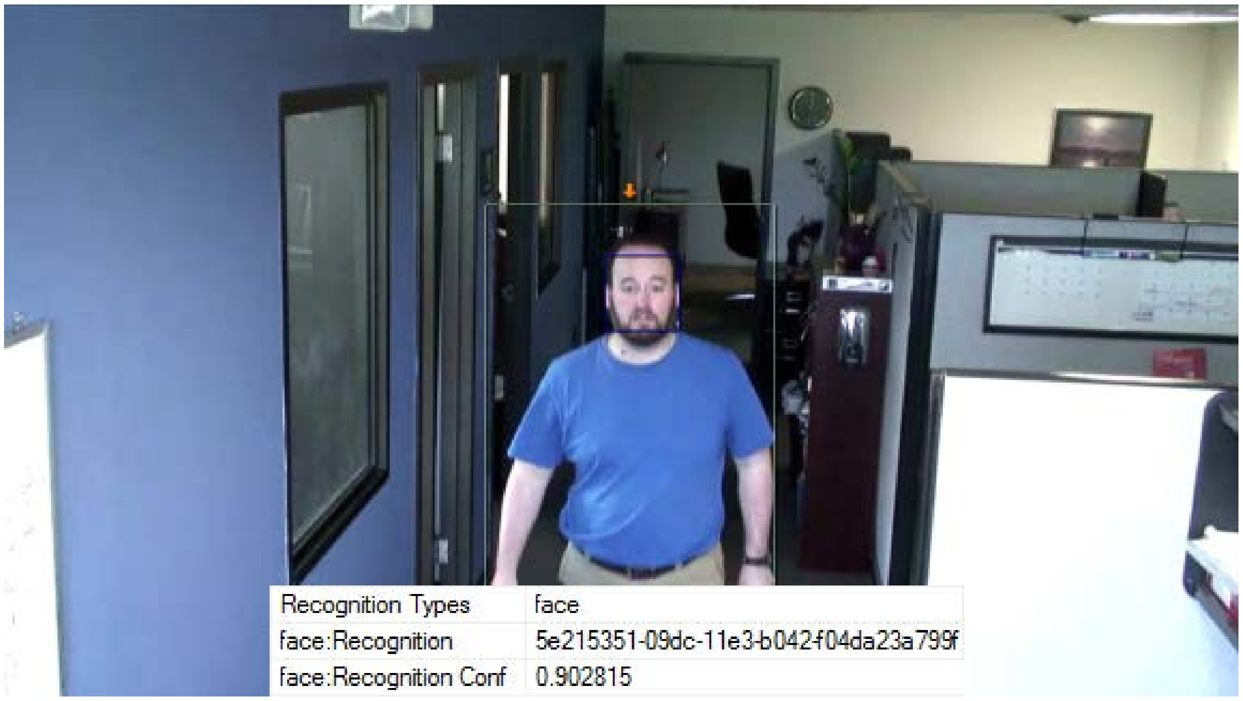
SDK via an xml settings file. Models would be deposited in this directory once generated and the filename would then be passed to the Tentacle system as a frame attribute as illustrated in **Figure 18**. Models exported on this frame are listed under "Exported Face IDs". Each model's file path is listed as *<ID>:Face Model File*. When a cross-camera object is suspected, Tentacle would pass the filename into the second camera (again via the frame attributes).



Property	Value
Position	15937
Date	08/20/2013
Time	17:07:52.867
⊟ Attributes	
FLAG_LAST_FRAME	false
gpu-device-in-use	0
processing-frame-rate	0.751777
ptz-roaming	false
video-uri	file:/D:\data\Primordial\intuVisionFaces9-dark.asf?iv-file-position=20000.000000
applied-saliency	No
auto-noise-filter-level	medium
rmi-pct	0.012369
1:Face Model File	D:\data\Primordial\FaceModels\5e215351-09dc-11e3-b042-f04da23a799f.dat
Exported Face IDs	1
⊞ Objects	

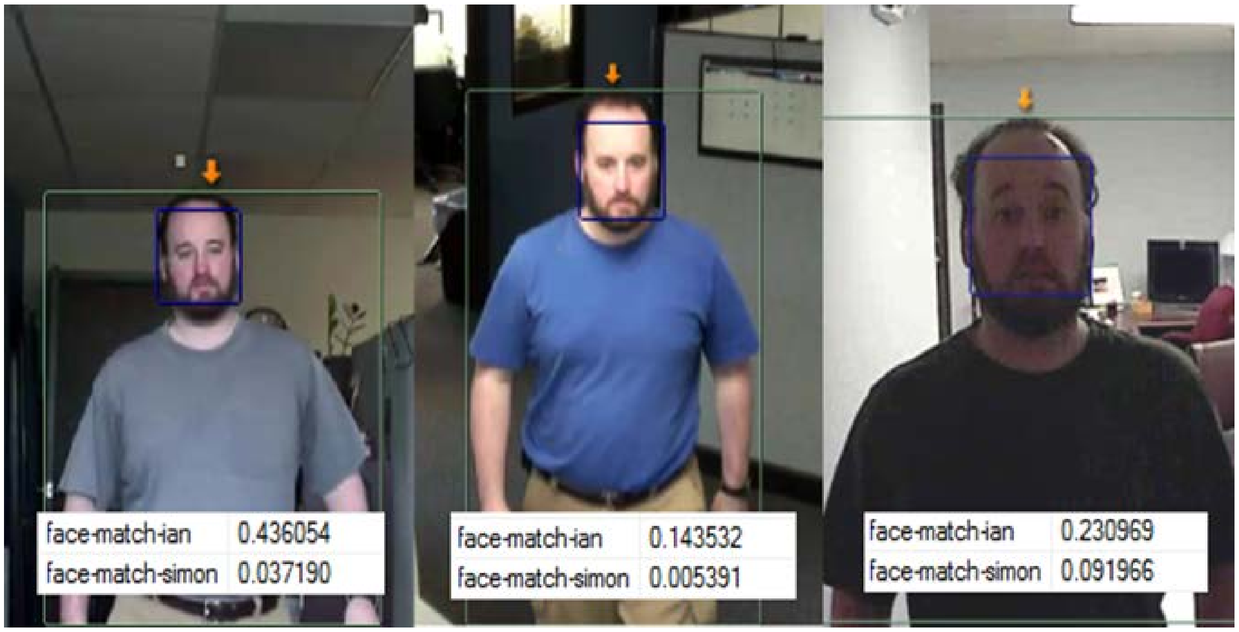
Figure 18. Frame Attributes From IntuVision SDK for Face Matching

The second camera would then match detected faces against the models provided in this manner, and report any positive matches (**Figure 19**, **Figure 20**). **Figure 19** was matched against the previously exported model at a 90% confidence.



Recognition Types	face
face:Recognition	5e215351-09dc-11e3-b042-f04da23a799f
face:Recognition Conf	0.902815

Figure 19. Matched Faces are Marked in the Object's Attributes



face-match-ian	0.436054
face-match-simon	0.037190

face-match-ian	0.143532
face-match-simon	0.005391

face-match-ian	0.230969
face-match-simon	0.091966

Figure 20. Face Matching

Finding a unique ID is challenging in this case, due to large variation in angles/resolutions, but results were promising for improving entity correlation.

3.3 Improve Entity Geolocation

After the beginning of Phase II, the focus had changed to moving target detection which is a distinctly different problem than static camera detection. Because the initial entity geolocation in Phase I with static cameras performed quite well, only incremental changes were made in this task to the existing static camera system, with most of the effort being focused on UAV-tracked entity geolocation.

To improve the smoothing of entity positions, during the work on entity correlation, Primordial updated the client GUI application to only display entities which were tracked over several frames; this improved the output and reduced erroneous visual reporting of entities on the map. Additionally, when making modifications to the camera resolutions for video input, Primordial discovered errors in the 3D projection behavior due to the change in resolution. They spent significant time modifying the Phase I algorithms to be more robust to changing camera resolution values which made the system much more flexible.

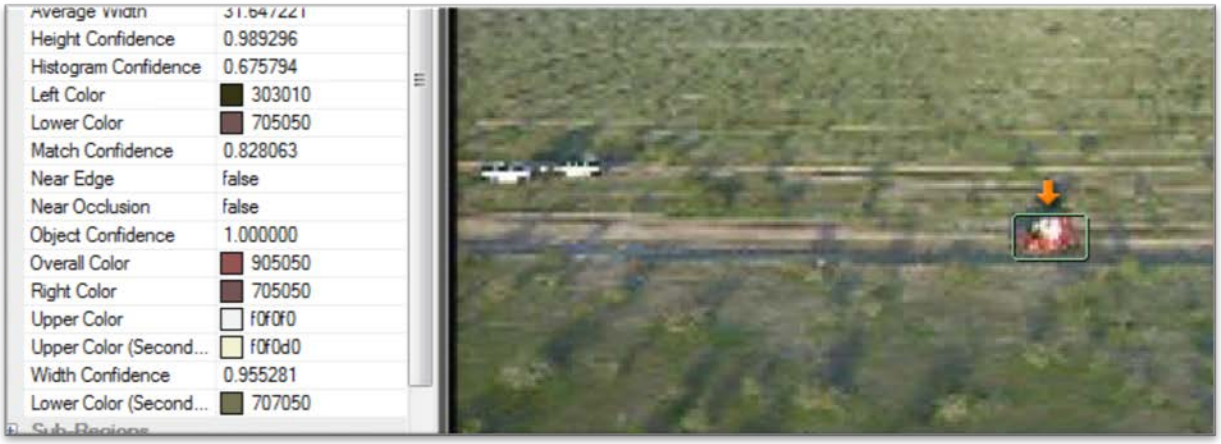
Most of the effort in Phase II was to support geolocation of entities from moving platform video inputs, such as UAV-recorded video clips, through the use of its integrated telemetry data. Once the frame's corner positions were determined from the telemetry, a projection algorithm essentially the same as the one from Phase I was used to project the target pixel locations to the ground plane. More information on this process is included in the moving camera tracking section of this document.

3.4 Implement Moving Camera Tracking

One major focus of Phase II was the addition of the ability to process and analyze UAV video feeds. This task constituted a very large portion of the work done in Phase II for the project.
In order to update the Tentacle system to support UAV video, it was crucial to have access to sample UAV tracking videos for initial testing and prototyping. Not only was it essential to test the tracking algorithms on actual UAVs to be sure they would perform properly, but also because UAV video is either paired with or contains embedded within it metadata related to the geographic location of the area on the video. This information is needed in order to georectify tracked target locations. With these two needs in mind, Primordial performed a survey of existing datasets, and was initially able to obtain UAV videos from the following sources: DARPA Video Verification of Identity (VIVID) Program: UAV tracking videos collected at Fort Picket; University of Central Florida (UCF) dataset: Quad-rotor UAV tracking videos for image processing research; Technical University Munich (TUM) dataset: UAV tracking videos for image processing research and YouTube: Police helicopter chase-videos, as well as low quality UAV videos from in theatre.

Primordial consulted with IntuVision to develop an initial format for the UAV metadata API (Application Programming Interface) for Tentacle. Primordial made a request to IntuVision to modify their SDK to ingest pre-detected bounding boxes. This would allow objects detected by external algorithms to be tracked, as well as to allow the extraction of classification and object color metadata as depicted in

Figure 21. IntuVision updated their SDK to support this and provided a supplemental document that described integration steps to Primordial once this was completed.



Average Width	31.647221
Height Confidence	0.989296
Histogram Confidence	0.675794
Left Color	■ 303010
Lower Color	■ 705050
Match Confidence	0.828063
Near Edge	false
Near Occlusion	false
Object Confidence	1.000000
Overall Color	■ 905050
Right Color	■ 705050
Upper Color	□ f0f0f0
Upper Color (Second...	□ f0f0d0
Width Confidence	0.955281
Lower Color (Second...	■ 707050
Sub-Regions	

Figure 21. UAV Tracking Via Object Bounding Box Ingestion

During this period, Primordial began modifications to the core Tentacle system to support entity detection from UAV video. In order to accurately place avatars for targets extracted from UAV video into the World Wind view, it was necessary to geo-locate the tracked entities. To achieve this, Primordial updated Tentacle detection application to support extracting telemetry metadata from UAV video.

Initially, the only videos with available telemetry data was the DARPA VIRAT dataset. The telemetry in the VIRAT videos that were being used stores information about the vehicle and camera positions and pose in each frame, using the standardized KLV format. Primordial developed a tool to extract KLV metadata from the video files using the Python programming language and updated Tentacle to support loading the metadata from a separate file and then associating it correctly with each ingested frame. However, initial assessments indicated that the errors in the VIRAT telemetry were surprisingly high. During the initial tests with one of the videos these errors were often greater than 1km when projecting a tracked entity onto the ground. After a manual investigation of several of the data files, it was found that the telemetry in some cases was even reporting frame positions in the wrong direction and in positions that were as far as an entire U.S. state away from the real-world location. Primordial contacted researchers at Kitware who were involved in the original VIRAT project, and they indicated that the telemetry in the videos was in fact unusually low quality because an auxiliary sensor was attached to the UAV and used to collect the data rather than the actual UAV sensor which would normally be used. This suggested to Primordial that one high priority going forward should be to obtain UAV sample videos with actual telemetry from the built in sensors included as metadata. Without this, it would be difficult if not impossible to accurately perform 3D projection of entities tracked in the video frames.

Given the low quality of the telemetry that Primordial had access to at the time, they began implementing a fallback approach to geo-rectify UAV videos manually with nearby aerial imagery. Initially, geolocation of tracked targets was performed using a linear interpolation of pixel locations based on the frame corner latitudes/longitudes captured from the metadata **Figure 22**. However, this first-pass approach was relatively inaccurate, and so the code was updated to instead use a homography mapping approach. In the updated version, the frame-corner locations were used as data-points to solve for the parameters of a homography mapping. The mapping then used manually selected tie points linking the camera view to the aerial imagery view; the corner and center locations of the video frames were then be mapped to latitude/longitude values and stored to an XML metadata file. The frame corner and center locations were a natural choice for extraction, since that information corresponds to the telemetry metadata available in the DARPA VIRAT UAV videos. Once the video frames were geo-rectified, it was straight-forward to project the tracked avatars onto the World Wind aerial imagery (**Figure 23**) using the created mapping. Also, particularly in cases where the video has relatively wide field of view such as WAMI (Wide Area Motion Imagery) video, there was potential in the future to automate the process of feature matching e.g. using scale-invariant feature transform (SIFT) features, but this was deemed to be too complex for a first pass version.

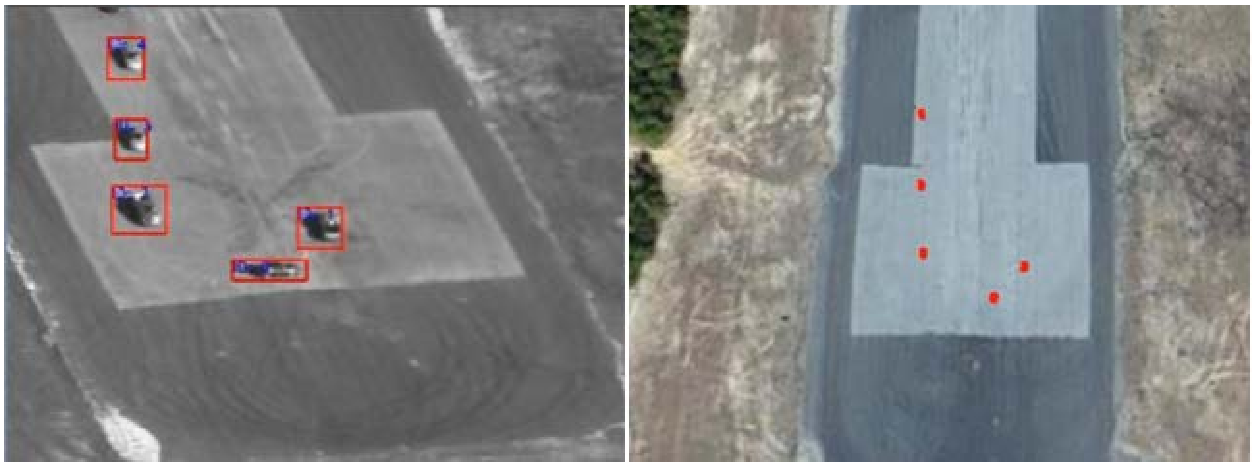


Figure 22. First Pass Tracking Using Linear Interpolation of Points

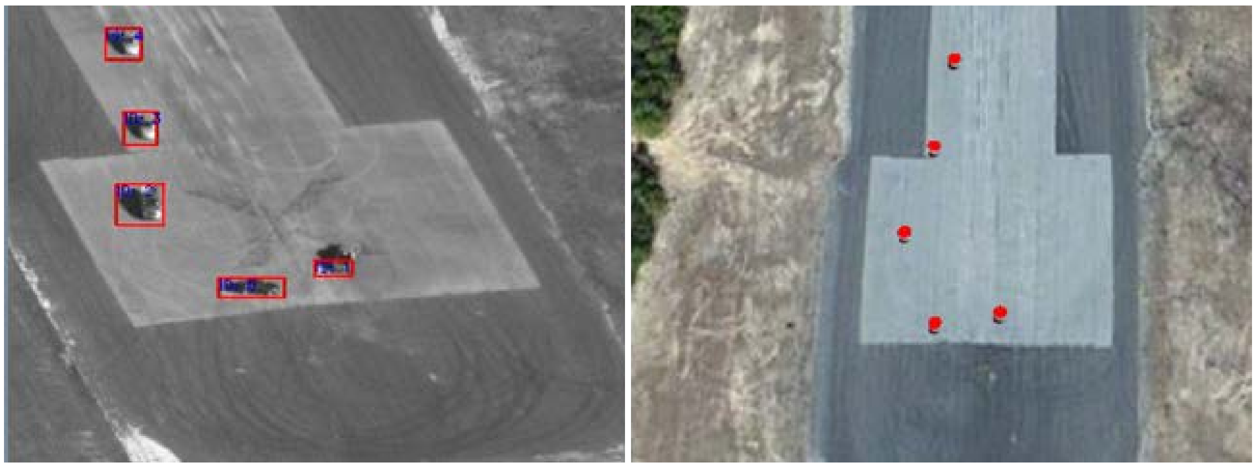


Figure 23. Tracking After Implementing Homography Mapping Approach

After the geolocation problem was resolved, Primordial began work on preliminary approaches to UAV moving video tracking. For static surveillance cameras, moving entities are comparatively straight-forward to extract, e.g. by performing background subtraction relative to the stationary view. However, for moving cameras mounted to UAVs, both the foreground and background are moving at once, making the foreground extraction a more challenging process. To start, Primordial began adding support for two types of approaches for moving-camera-based tracking: motion-based and appearance-based.

Primordial completed a preliminary implementation of the primary motion-based tracking approaches described in (5). The basic concept of this approach is to use automatically tracked image features (e.g. Shi-Tomasi features) to determine the homography mapping necessary to rectify neighboring image frames in the video. Once the homography is determined, the nearby frames are warped into alignment and a difference image is computed reflecting the changes that have occurred between the frames. The blobs extracted from these difference images reflect an estimate of the foreground entity locations. However, because this is a noisy foreground estimate, a sliding window of image differencing is performed, with each frame differenced with multiple nearby image frames; each difference image is then summed and the summation values are adjusted logarithmically. A threshold is applied to this summed difference image which then provides estimates of detected foreground blob locations.

Once the blob estimates are extracted, the entities are tracked over time based on spatial proximity, though the plan was to eventually update it to use appearance attributes as well. The tracked entities are automatically assigned IDs, and the tracking information is used to improve the quality of the entity detection. In particular, an entity is only considered detected after it has been successfully tracked over multiple frames. Also, once an entity has been successfully tracked over time, a less strict standard is applied for determining whether the entity has been detected in subsequent frames. See **Figure 24** for a screenshot of the tracking results on a DARPA VIVID UAV video. The approach involves warping nearby frames into alignment and performing image differencing. The IDs are automatically assigned in the order that the entities are first detected.

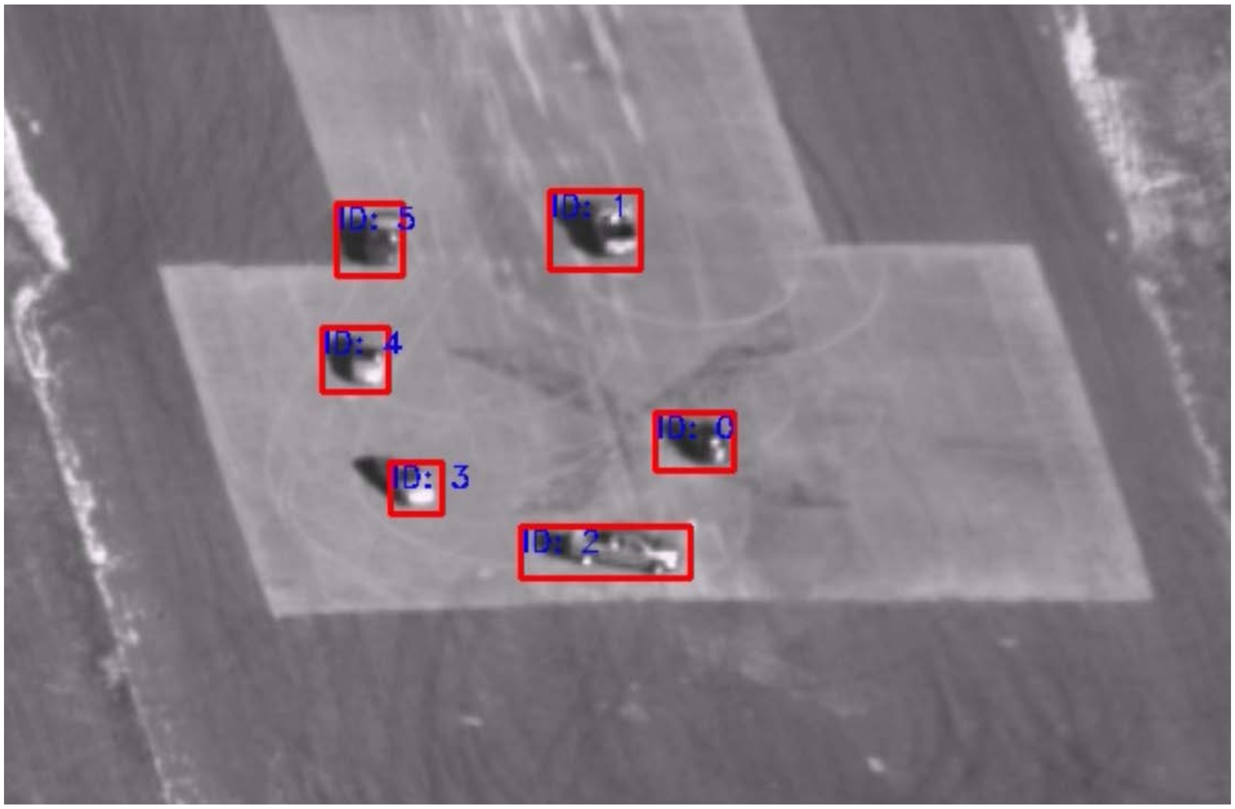


Figure 24. A Screenshot of Tentacle's Motion-Based UAV Tracking Results, Applied to UAV Footage from the DARPA VIVID Dataset

In addition to the motion-based approach described above (where entities which are moving relative to the background are extracted), Primordial also implemented multiple well-known techniques for tracking entities over time based on appearance models, including mean-shift, cam-shift, template tracking, and edge-based template tracking. One downside of these approaches compared to the motion-based approach is that they require manual initialization of the entity's location in order to extract an appearance model and begin tracking. One potential solution to this was to seed the appearance-based approaches based on the motion-based blobs. The mean-shift approach extracts a histogram of the entity's hue (or other feature) values and then iteratively searches in the vicinity of the entity for the best match in the next frame. The cam-shift algorithm builds on this, allowing for changing entity size/orientation over time. The template-based approaches, simply perform pixel-by-pixel comparisons with the entity image bounding box, to find the closest match location in each successive frame; the template is updated each frame, in order to account for changes in the entity's appearance over time. In addition, to RGB image-patch-based templates, Primordial also implemented a template-tracking approach using edge images, with edges extracted using Sobel operators. Of these appearance-based approaches, the RGB image template seemed to offer the most robust results, though Primordial planned to test it more thoroughly. At this point, Primordial planned to augment the motion-based tracker above with the appearance-based techniques described in this section.

After an initial evaluation, Primordial identified several issues with the current state of the UAV tracking. It was deemed necessary to improve robustness across range of test videos, combine appearance and motion based information, and optimize for real-time performance as well as to improve entity correlation when entities were detected.
Primordial created a test video to demonstrate the state of UAV tracking at this point (5).

In addition to Primordial's in-house UAV tracking algorithm implementations, CMU completed an initial version of their Linux-based UAV tracking SDK and provided a software release to Primordial. This release primarily provided a wrapper around algorithms that CMU had already developed under previous research contracts. CMU also provided a summary of their proposed API for the tracking SDK, which would allow Primordial to incorporate CMU's UAV video tracking capabilities directly into the Tentacle system. Primordial was able to setup a Linux workstation and experiment with this SDK using some of the DARPA UAV video sets. The CMU tracker supported template and color based approaches as well as simultaneous tracking of multiple entities in a UAV video as well as automatic video stabilization. The CMU visual tracking SDK had been extended to provide an interface for accessing target masks corresponding to the estimate of foreground and background pixels in each active target track sub-window. Each tracking algorithm used a different internal representation and combination of target attributes. As such, the characteristics of the mask produced depended on the tracker algorithm in use. Target masks were provided to Tentacle project partners to support investigation of track metadata classification. **Figure 25** shows CMU's initial SDK using manual targeting of entities. Yellow crosses indicate targets manually identified by the user which are then tracked.

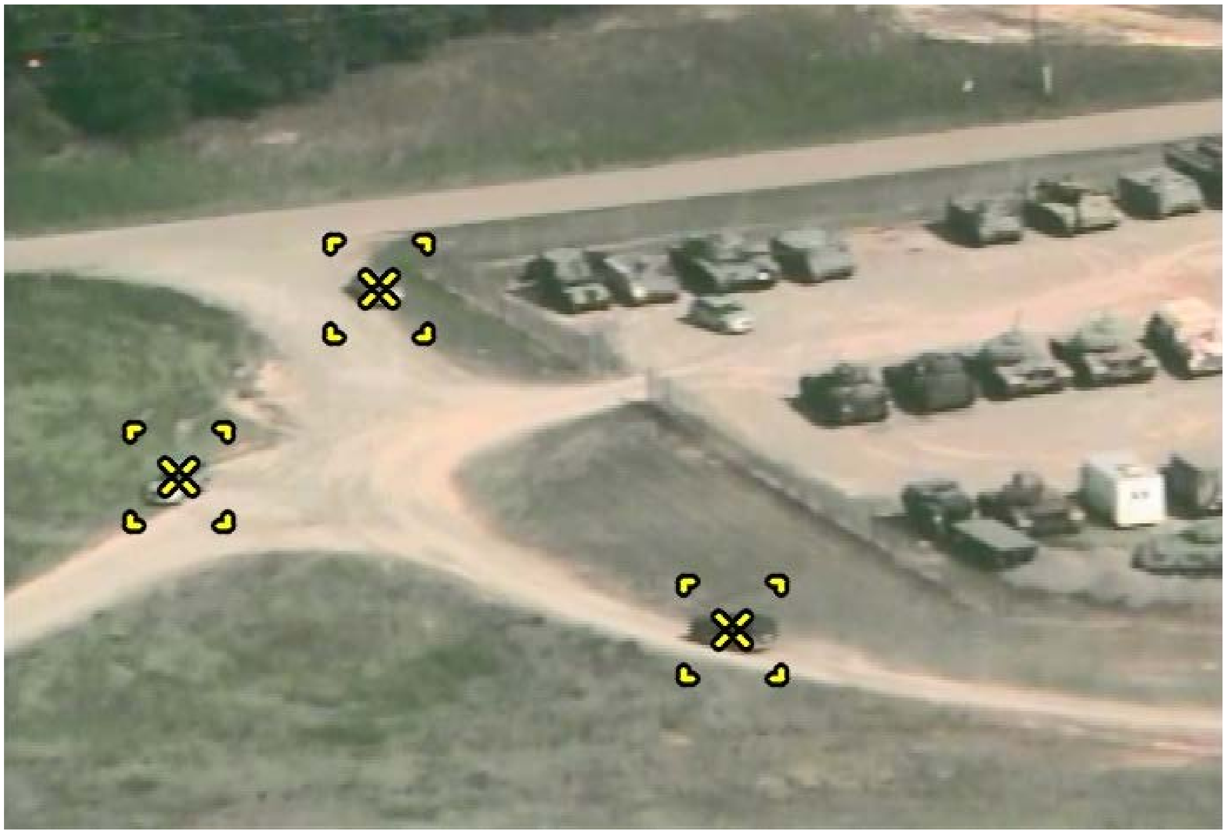


Figure 25. Screenshot of the CMU UAV-based tracking system applied to a VIRAT UAV video

At this stage, Primordial created another demo video to display the state of the Tentacle system's capabilities for AFRL (6).

CMU began updating their UAV tracking SDK to support auto-entity detection (see **Figure 26**). **Figure 26** is an example video frame from the VIRAT dataset showing the target masks produced by two tracker types: Color Adaptive Mean-Shift and the Gradient Patch tracker tracking a moving SUV. These foreground masks are provided to facilitate accurate metadata extraction. The goal of this feature was to allow for more accurate initialization of the UAV tracker, since currently the Tentacle UAV tracking was often not accurate if the user did not manually select the initial ground target locations. The approach being used for this was known as the motion history image (MHI) technique (**Figure 27**), in which adjacent frames are warped into alignment and a history of moving pixel locations (based on frame differencing) are merged into temporally-coded images representing the motion history (**Figure 28**). **Figure 27** represents auto-detection of moving entities with a moving camera (in a UAV video stream), using the motion history image (MHI) technique. The moving pixel locations are found by warping adjacent frames into alignment and then performing frame differencing. The colors represent different time slices in the motion history. Computing the MHI is an initial step in the full auto-detection implementation. The moving entities can then be extracted as the ends of chronologically ascending trails in the motion history (**Figure 29**). This method of detection provided candidates for tracker initialization in the SDK where previously the operator would need to manually tag objects to track.

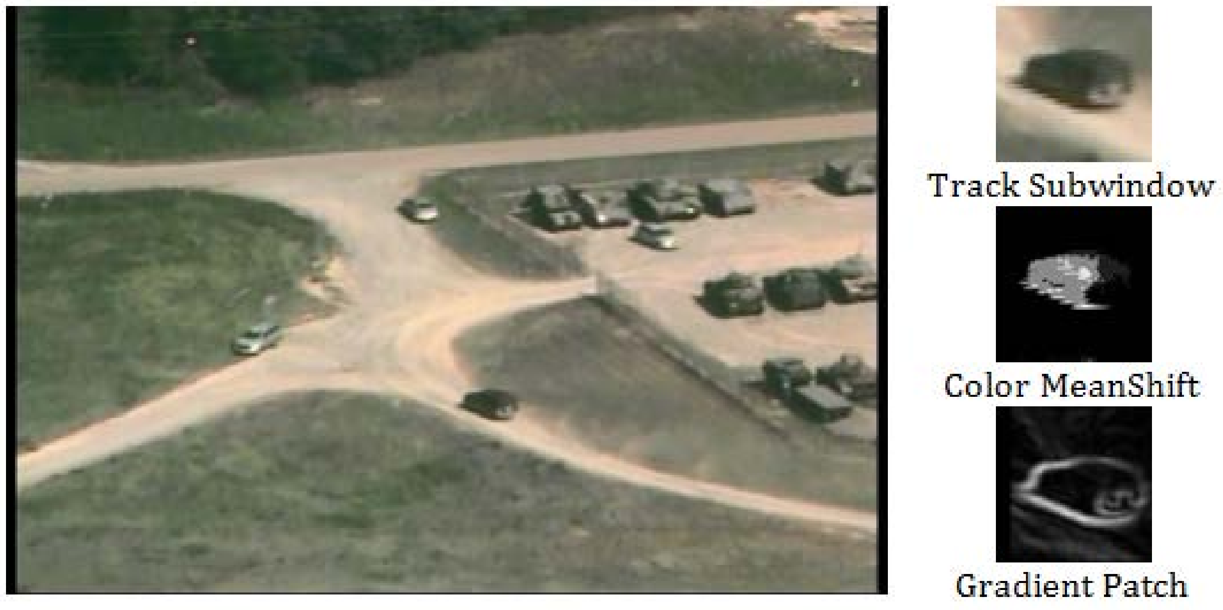


Figure 26. UAV Target Masks

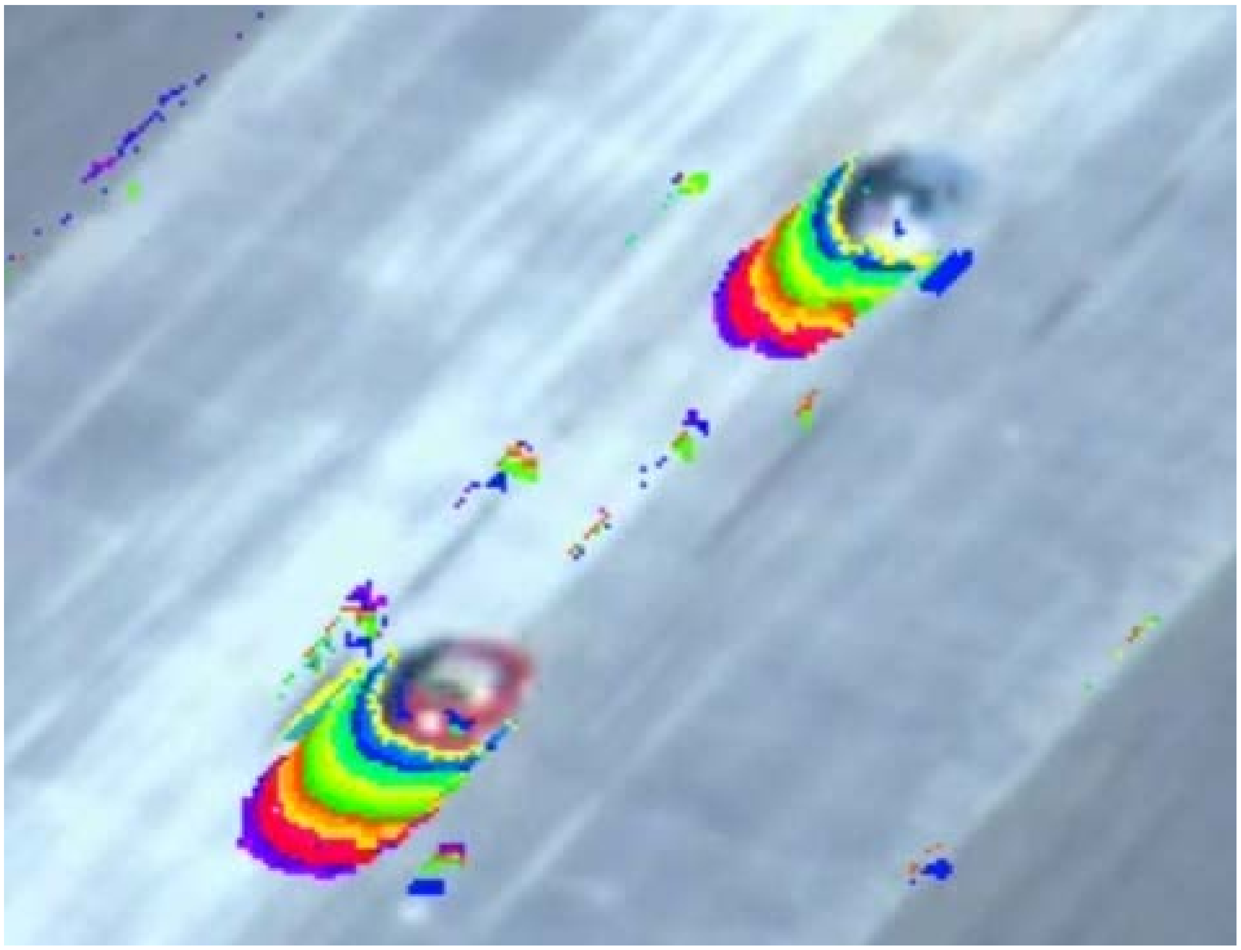


Figure 27. Motion History Image

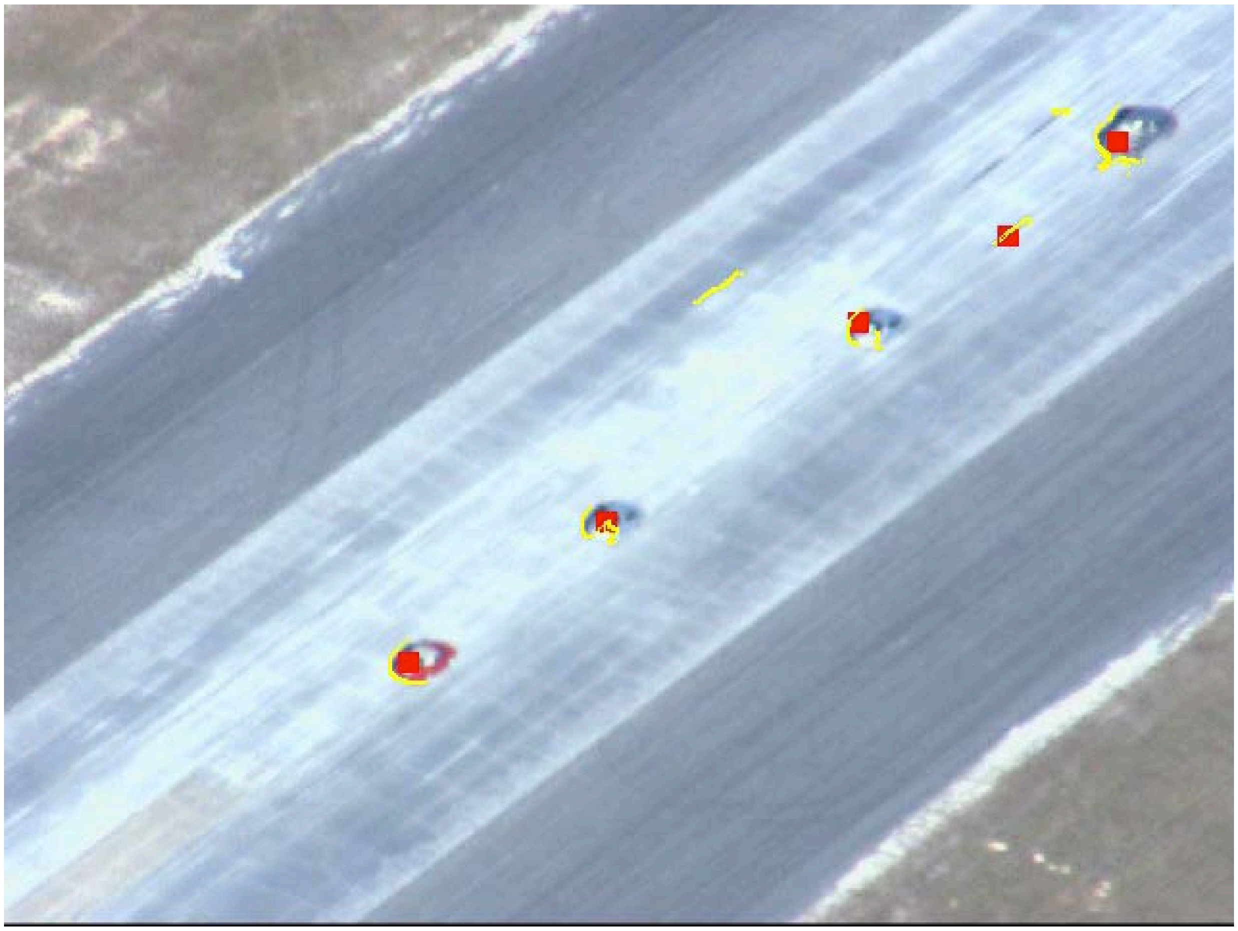


Figure 28. Steady Motion (yellow line), Target Candidate (center)

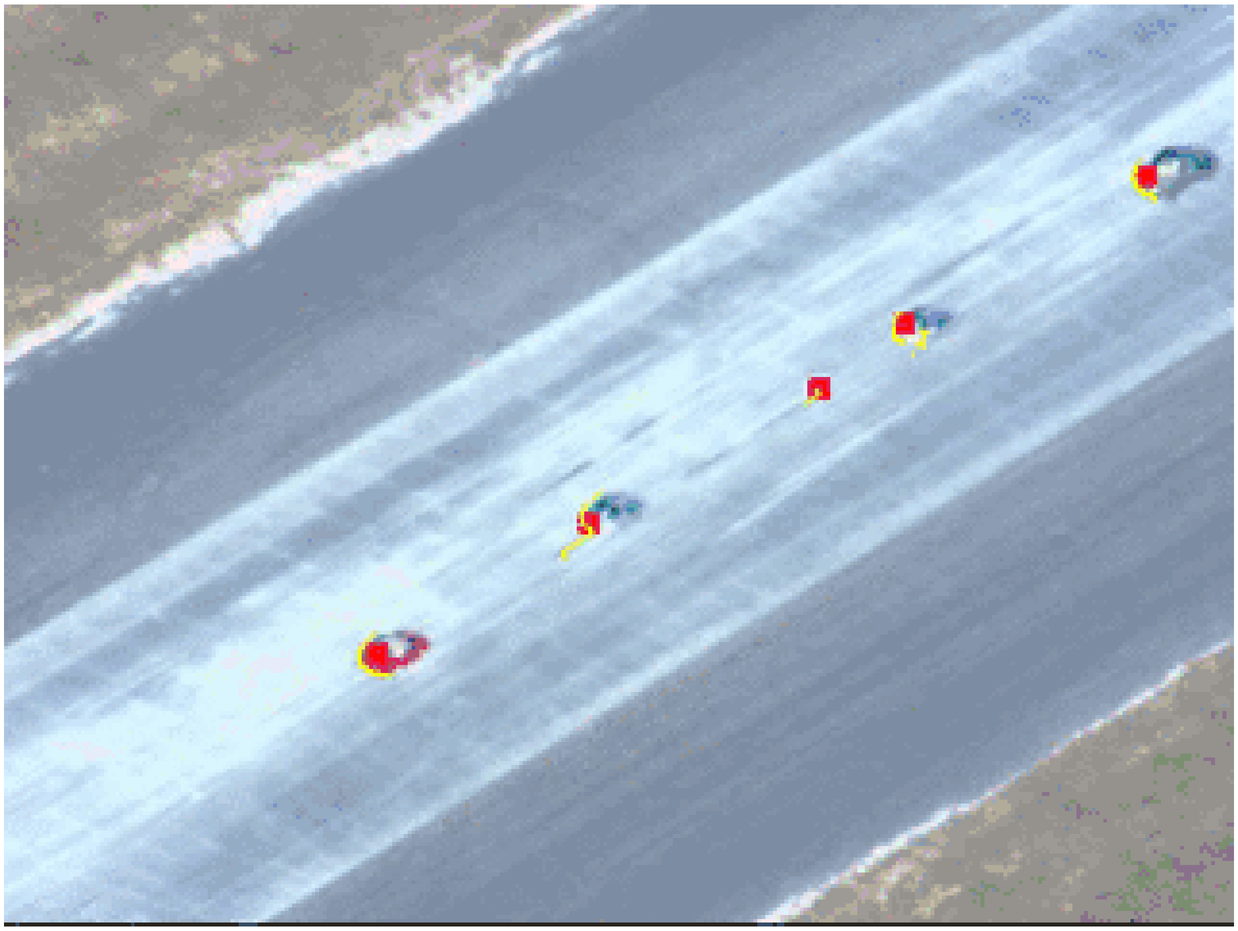


Figure 29. Target Tracking

IntuVision began investigating the use of Biologically Inspired Models (BIM) based classification for differentiating vehicles and people in UAV videos. The advantage of BIM classification for the UAV domain is that unlike IntuVision's standard SVM (Support Vector Machine) classification method, precise object masks (which pixels within a bounding actually belong to the object) are not required.

BIM-based classification has been found to be a best fit for video surveillance data since the BIM features are tolerant for the partial occlusions and changes in illumination. The model is also sometimes called HMAX (Hierarchical Model and X). The BIM tries to build a system that emulates object recognition in human cortex and has experimentally shown to perform better than some current state of art features extractors for several object detection/classification tasks. BIM is the bases of Scale Invariant Feature Transform (SIFT) features and Histogram of Oriented Gaussians (HOG) model for object recognition. The Biologically Inspired Model features extracted from detected object images, each feature is obtained by combining the response of local edge-detectors that are slightly position- and scale-tolerant over neighboring positions and multiple orientations (like complex cells in the human primary visual cortex). These features are more flexible than template-based object models as they allow for small distortions of the input; and they are more selective than histogram-based descriptors as they

preserve local feature geometry. BIM's higher performance is partly due to these properties. Because of this, BIM features exhibit a better trade-off between invariance and selectivity than template-based or histogram based features extracted from object images.
The main steps of the BIM feature extraction is described in Appendix B, a graphical representation of the BIM model training is depicted in **Figure 30**.

IntuVision's BIM based person/vehicle classification used the computed C2 features for training and testing a Support Vector Machine (SVM) classifier using 200 patches as BIM features.

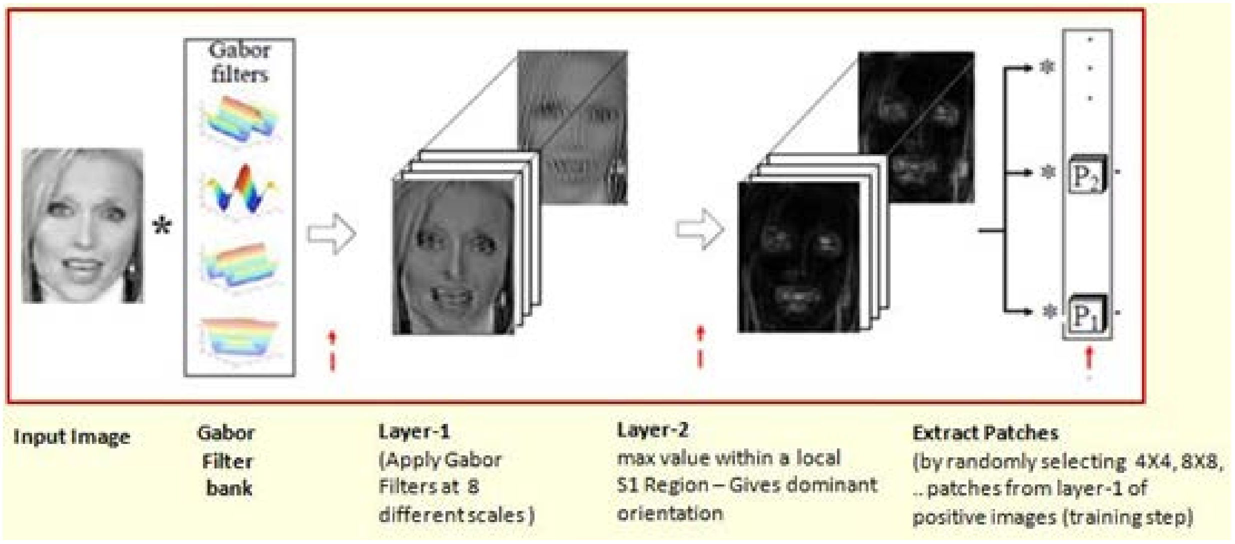

Gabor filters
*
P2
P1
Input Image
Gabor Filter bank
Layer-1 (Apply Gabor Filters at 8 different scales)
Layer-2 max value within a local S1 Region – Gives dominant orientation
Extract Patches (by randomly selecting 4X4, 8X8, .. patches from layer-1 of positive images (training step)

Figure 30. BIM Model Layers for an Example Image

The first experiment using BIM was to simply test IntuVision's existing person and vehicle models against the available example UAV videos frrm the VIVID and CRCV (Center for Research in Computer Vision at University of Central Florida) data sets. The existing models were trained with Person and vehicle samples that are taken from regular surveillance camera views (no more than 10-15 meter-high camera views) (**Figure 31**, **Figure 32**). Because of the difference in resolution and view angle, the existing models were not appropriate for UAV imagery.

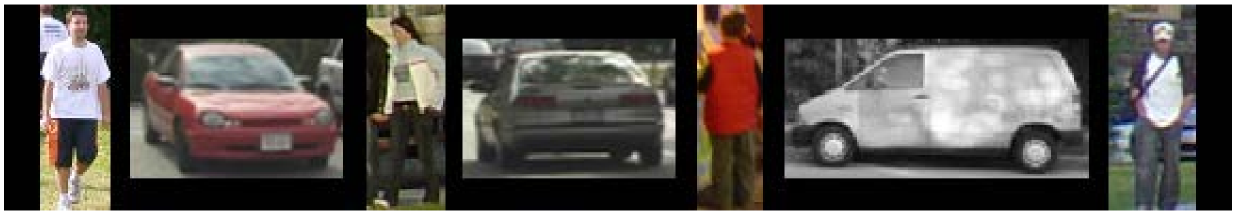


Figure 31. Training Examples From Existing Person And Vehicle Models

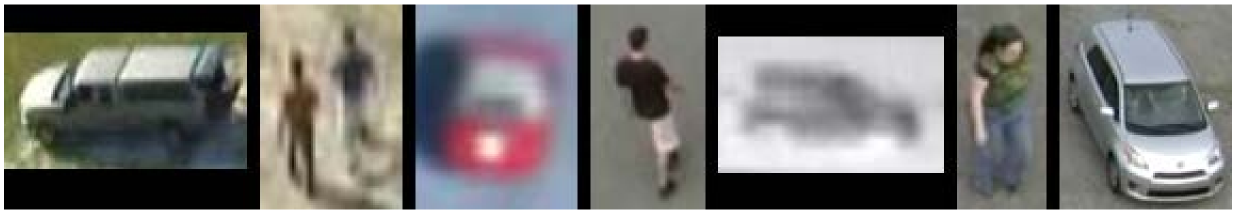


Figure 32. Sample People and Vehicles from VIVID and CRCVG Datasets

These models typically require few hundred training samples to train a robust BIM model. To gather sufficient training samples, IntuVision turned to the Overhead Imagery Research Data Set (OIRDS) for building new vehicle models (OIRDS only contained vehicle data and annotations at that time). One downside of this dataset is that it did not include any examples for training a person classification model.

The OIRDS dataset included both UAV (from VIVID) and satellite imagery from USGS (US Geological Survey) (**Figure 33**). Since the model was to be tested against the VIVID dataset, IntuVision relied exclusively on the USGS data in building the new vehicle models.

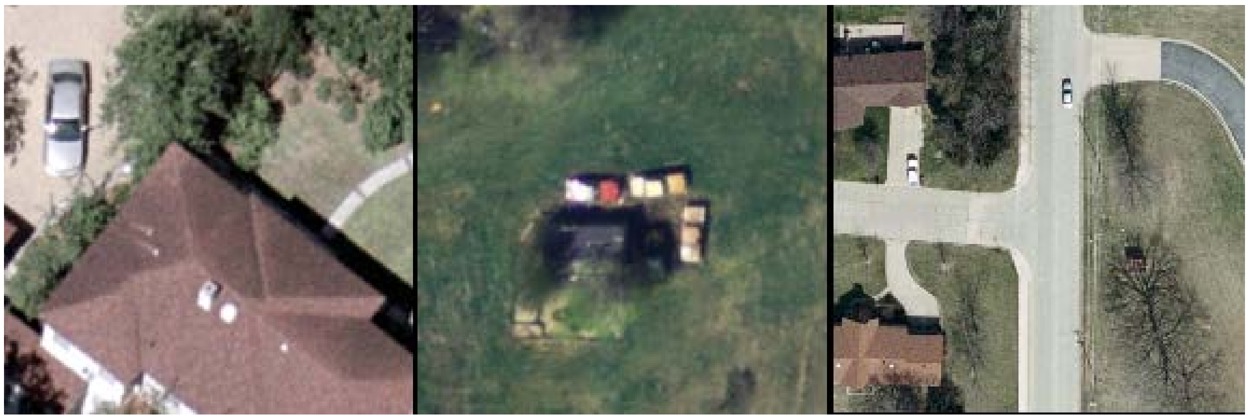


Figure 33. Sample images from OIRDS Image Dataset

The newly trained vehicle model worked well on the VIVID dataset (**Figure 34**), especially where the view angle is similar to that of the satellite imagery used for training (directly overhead). Data collections from left to right: Eglin Air Base, Fort Pickett and CMU Red Team ground vehicle. The “RedTeam” subset (shot at a much lower angle) performed less well with this model, but the vehicles were still classified correctly in many frames.

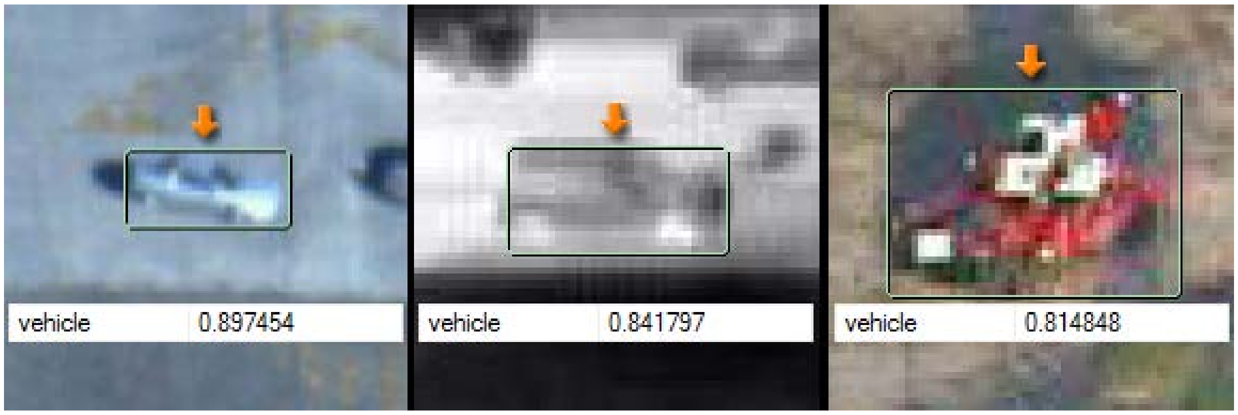


Figure 34. Sample Vehicle Classification for the VIVID UAV Video Dataset

The CRCV dataset (shot from a low-flying balloon platform) showed problems with the new vehicle model too, owing largely to the difference in object resolution and view angle (which stands part way between the overhead view and the standard head-on viewpoint)(**Figure 35**).

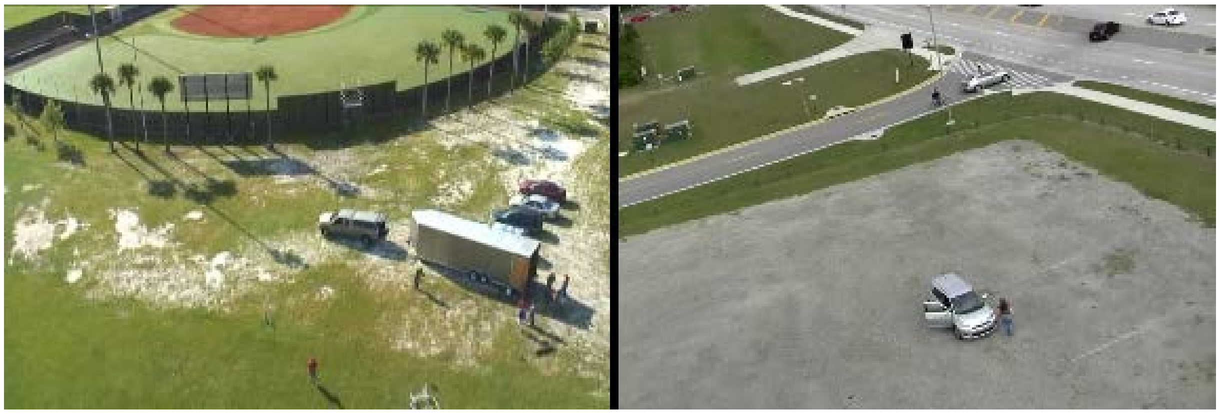


Figure 35. Example Screenshots from the CRCV Dataset Videos

To address the shortcomings, IntuVision extracted person and vehicle training examples from the VIRAT video data set. The VIRAT Ground Dataset has a similar view angle to the CRCV data set, and includes scenes containing both vehicles and pedestrians (**Figure 36**).

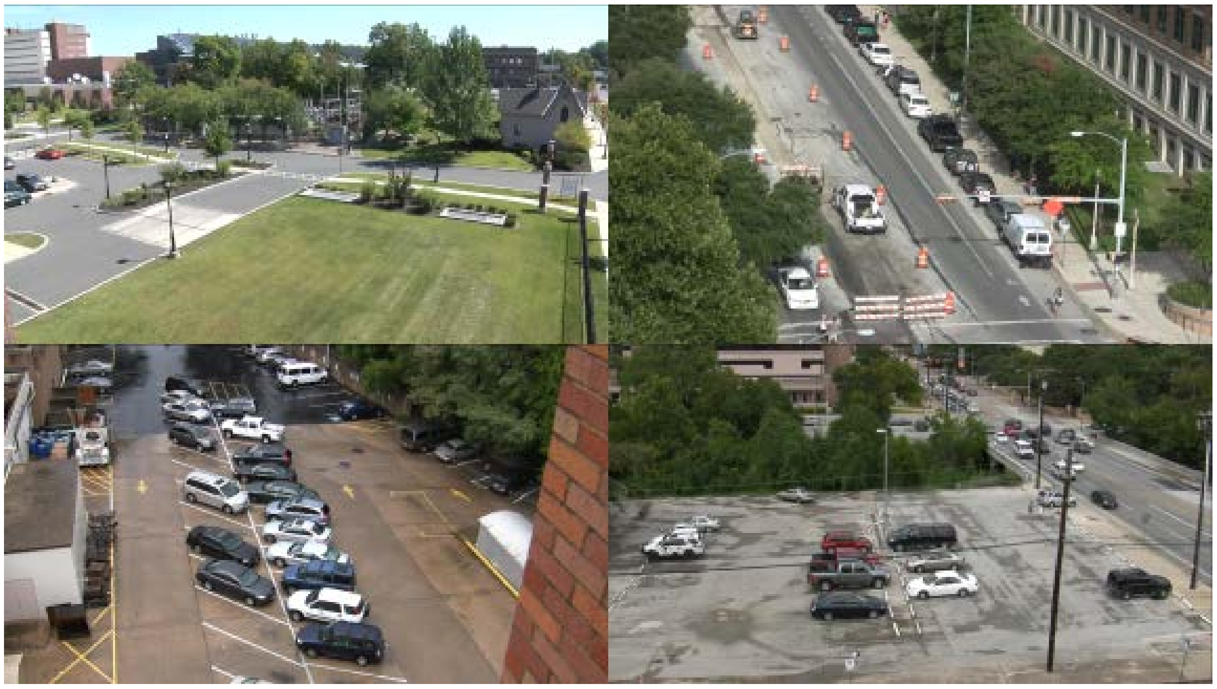
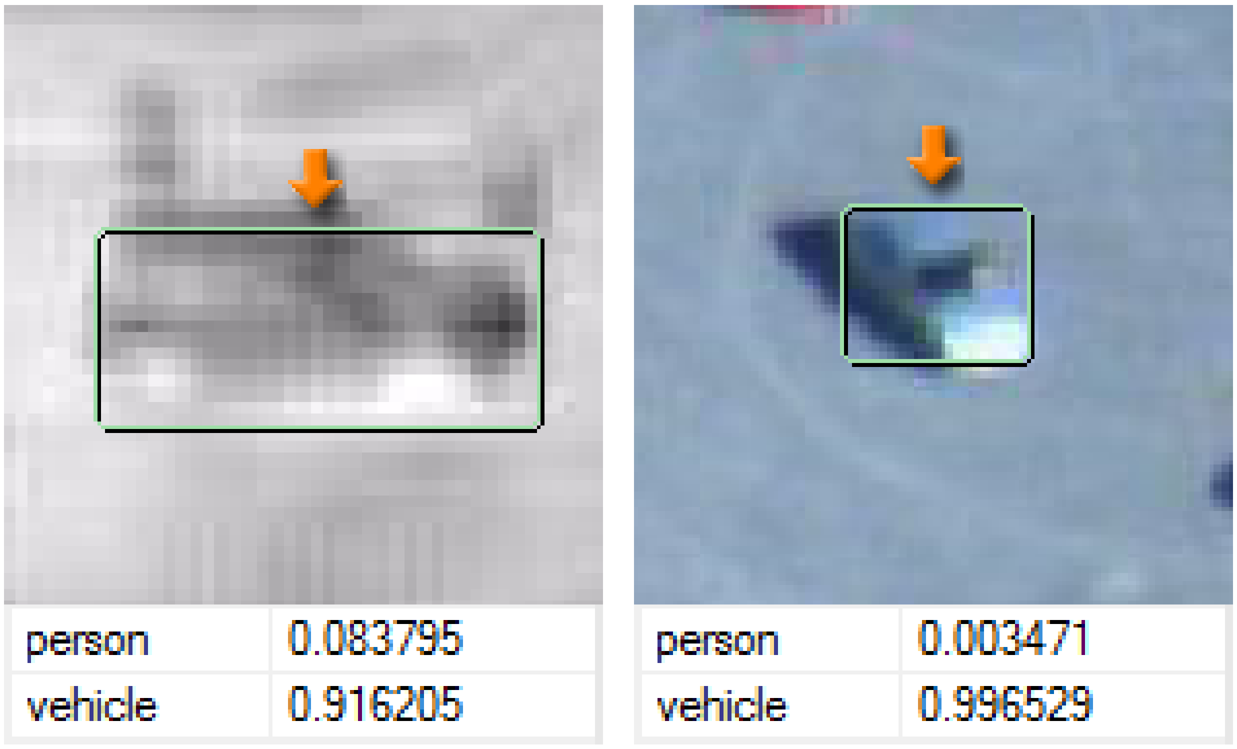


Figure 36. Low Angle Overhead Video from the VIRAT Dataset Contain Both Vehicle and Pedestrian Traffic

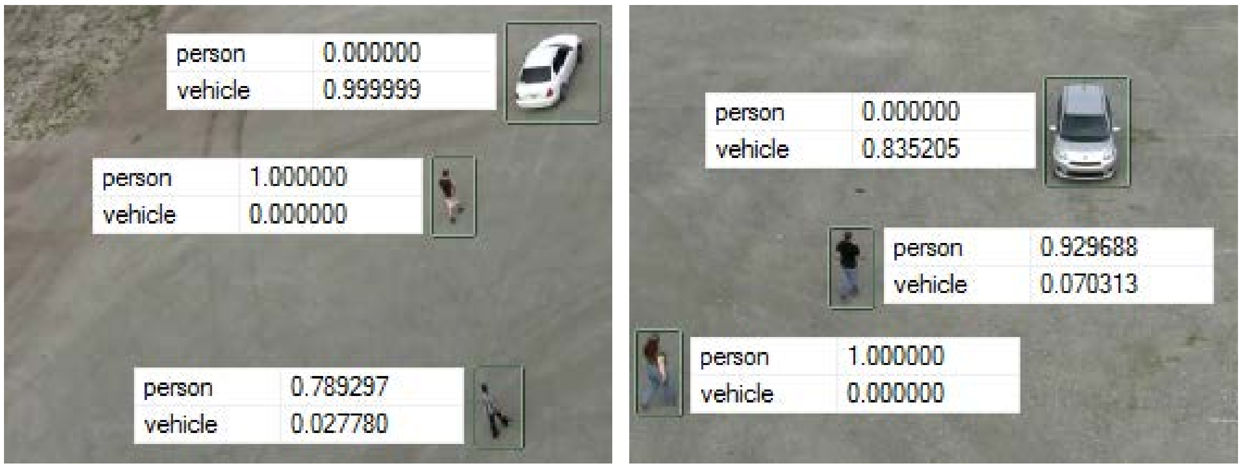
The models IntuVision trained using both VIRAT and ORDIS data worked well in both the high overhead VIVID and the lower angle CRCV videos as illustrated in the following examples (**Figure 37**, **Figure 38**).



person	0.083795
vehicle	0.916205

person	0.003471
vehicle	0.996529

Figure 37. Object Classification in VIVID Videos (Fort Pickett and Eglin Air Base)



person	0.000000
vehicle	0.999999

person	1.000000
vehicle	0.000000

person	0.789297
vehicle	0.027780

person	0.000000
vehicle	0.835205

person	0.929688
vehicle	0.070313

person	1.000000
vehicle	0.000000

Figure 38. Object (Person and Vehicle) Classification in CRCV Videos

Starting from the vehicle and person classification models previously trained, IntuVision performed classification ROC (Receiver Operating Characteristic) tests on the available data to establish a benchmark.

To compare with the previous models, IntuVision created training and test sets for person and vehicle models from multiple datasets, e.g. the CRCV Action dataset, VIRAT dataset and

OIRDS dataset. **Figure 39** and **Figure 40** provide the ROC curves for the BIM vehicle and person models respectively.

In **Figure 39**, ROC curves for four vehicle classification models (including the original model from the last period) are shown. The shaded area on the graph indicates the preferred area of operation for the classification models with a true detection rate of over 70% and false acceptance rate below 30%. The dark blue curve corresponds to the original vehicle BIM model. The black cross indicates the default operating point of the model.

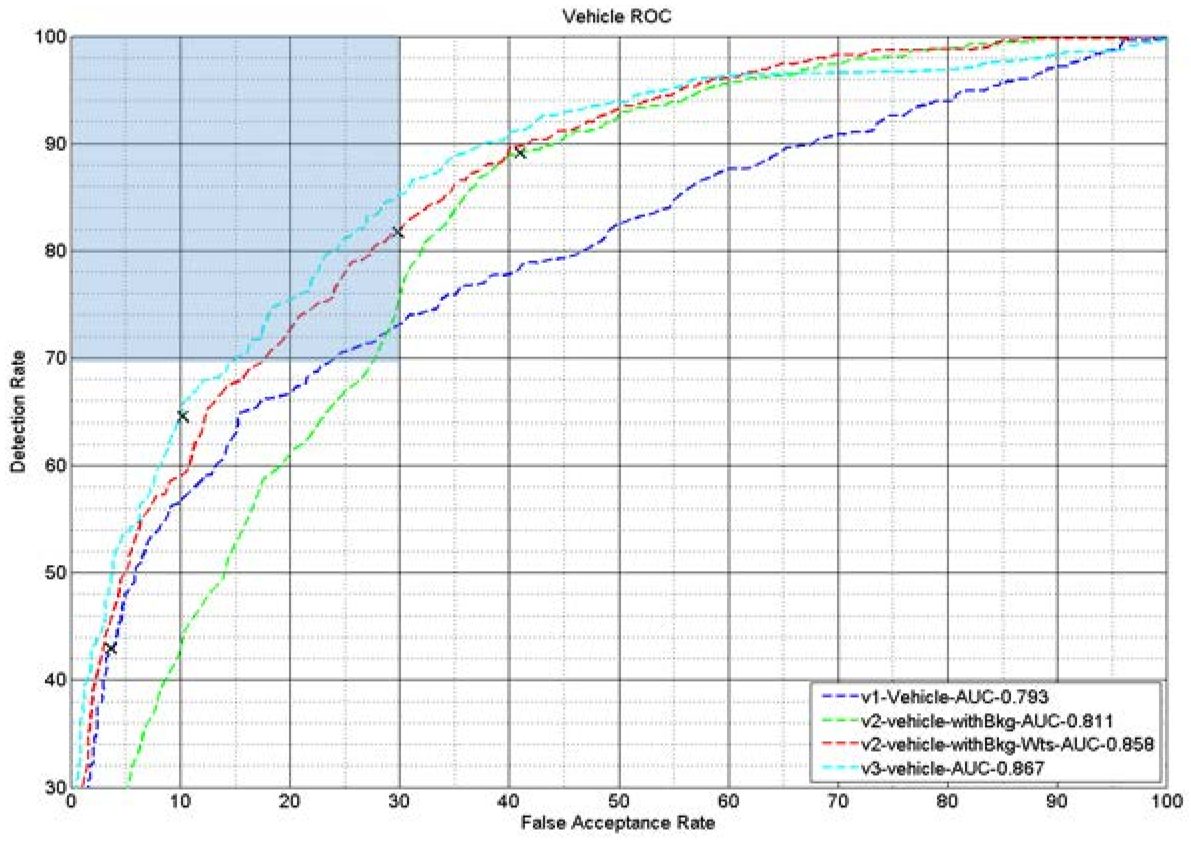


Figure 39. ROC for Vehicle BIM Model

The original vehicle BIM model is shown in dark blue. This model has an area under curve (AUC) value of 0.793 with the default operating point, score threshold of 0, at a detection rate of 42.92% and false acceptance rate of 3.65%. The original model used training samples from the VIRAT dataset only. IntuVision trained additional models including data from the other datasets as described below.

The second model with performance shown with the green ROC curve (in **Figure 39**) was trained with VIRAT, CRCV and OIRDS data. It has an AUC value of 0.811 with the default operating point at detection rate of 89.1% and false acceptance rate of 40.99%.

The third model corresponding to the red ROC curve with an AUC 0.859 uses the same data as green ROC curve model; they differ in the training penalty values. For this model the default operating point at detection rate of 81.69% and false acceptance rate of 29.83%. The fourth model with the best AUC value of 0.867, cyan curve, has the default operating point at detection rate of 64.49% and false acceptance rate of 10.22%. This model used VIRAT and CRCV data for training.

All the newer models had a better AUC value than the original model. If the default operating point was chosen, the third model with the red ROC curve would be better choice though the fourth model with the cyan ROC curve provided a better tradeoff between true detection and false acceptance rate overall.

Figure 40 provides the performance details for the five person BIM models. The original person BIM model, represented with a dark blue ROC curve, had a high AUC value of 0.905 with a high true detection rate at the default operating point. This model was trained on the VIRAT dataset. The dark blue curve corresponds to the original person BIM model. The black cross indicates the default operating point of the model.

IntuVision retrained the original model with different penalty values, the magenta ROC curve, resulting in the AUC value increasing to 0.914 while the default operating point had a higher true detection rate of 85.65% with a lower false acceptance rate of 20.34%.

The other two models, represented with cyan and red ROC curves, showed significant improvements in the low false acceptance region. These models were trained with CRCV data and had a default operating point at a much lower false acceptance rate. The final model, shown as the green ROC curve, was trained with the VIRAT and OIRDS datasets. It had a low false acceptance rate default operating point but it had the lowest AUC of all the models.
Overall, the newer models provided better classification with higher AUC with some models having a much better performance in the low false acceptance region.

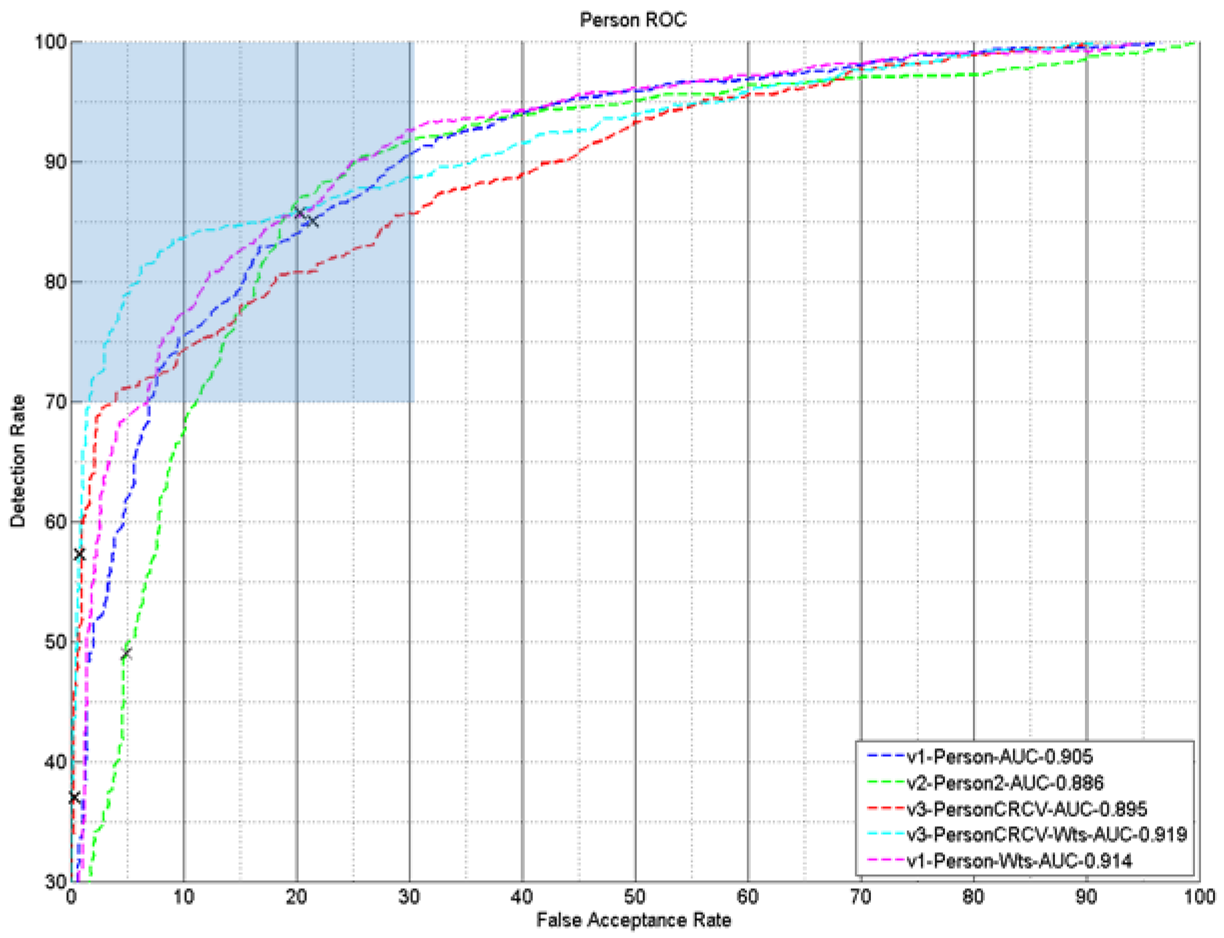


Figure 40. ROC for the Person BIM Model

IntuVision determined that the best practice for the final system may involve choosing the classification models that provide the best operating region for the type of data being processed.

The training & testing data for the additional models was collated from multiple datasets. Samples were chosen so that they cover different object poses and distances to object. All the trained models were tested on the test set in **Table 1** in the "Test Data" section. The OIRDS data set was partitioned into 2 non overlapping sets, some of the models used background patches from part 1 for training while part 2 was used to test all the models.

Table 1. BIM Testing and Training Data Results on Vehicles and People

Training data

<u>Vehicle BIM Model</u>

Vehicle model – AUC 0.793, blue ROC curve

Data Type	Data	Number of samples
Positive	VIRAT Vehicles	192
Negative	VIRAT People	197

Vehicle models – AUC 0.811 & 0.858, green & red ROC curves

Data Type	Data	Number of samples
Positive	VIRAT Vehicles	384
	VIRAT multi-pose vehicle	366
Negative	VIRAT people	394
	CRCV-Actions3 people	36
	OIRDS-Background	803

Vehicle Model – AUC 0.867, cyan ROC curve

Data Type	Data	Number of samples
Positive	VIRAT Vehicles	384
	VIRAT multi-pose vehicle	366
Negative	VIRAT people	394
	CRCV-Actions3 people	36

<u>Person BIM Model</u>

Person Models – AUC 0.905 & 0.914, blue & magenta ROC curves

Data Type	Data	Number of samples
Positive	VIRAT People	197
Negative	VIRAT Vehicles	192

Person Model – AUC 0.886, green ROC curve

Data Type	Data	Number of samples
Positive	VIRAT People	197
	Daimler People	150
Negative	VIRAT Vehicles	192
	OIRDS-Background	75
	Daimler Non People	75

Person Models – AUC 0.895 & 0.919, red & cyan ROC curve

Data Type	Data	Number of samples
Positive	CRCV Actions3 People	36
Negative	VIRAT Vehicles	384
	VIRAT multi-pose vehicle	366

Testing data

Data Type	Data	Number of samples
Vehicle	CRCV-Actions1	14
	CRCV-Actions2	26
	CRCV-AerialTape	178
	OIRDS-P2	672
Person	CRCV-Actions1	696
	CRCV-Actions2	226
	CRCV-AerialTape	144

IntuVision put in a request and sent a hard disk to receive an additional set of UAV test data (CSUAV - Columbus Surrogate Unmanned Aerial Vehicle) from SDMS (Sensor Data Management System).

IntuVision provided Primordial with an updated version of their SDK including the UAV video classification updates. However, these modifications had mixed success in Primordial's internal testing. Some vehicles and people were correctly detected, others were misclassified. Primordial discussed this with IntuVision and they confirmed the SDK was being used correctly but that the detection is still a work in progress. IntuVision expected them to improve in upcoming versions of the SDK. To assist IntuVision in making adjustments, Primordial implemented code to output bounding boxes and target mask image files in PNG (Portable Network Graphics) format for every frame in the video sets which were then sent to IntuVision for examination.

CMU also provided Primordial with an updated version of their UAV tracking SDK. Primordial then upgraded its Linux testing server to include the new SDK version and performed some initial first-pass tests, then gave feedback to CMU on the results.

Per their request, Primordial received a set of UAV-recorded video samples via hard drive from AFRL. To make better determination of which data sets would show representative targeting scenarios, Primordial did a thorough inventory and description of the test set videos. The summaries of the sets are in Appendix A.

Primordial noted that there were a lot of similar vehicles in the videos and that there was potential to combine all of the VIVID videos into a single set for easier testing if desired and feasible given available time.

Primordial decided to process the data files from three of the datasets that were received from the Air Force: CSUAV (Columbus Surrogate Unmanned Aerial Vehicle), CLIF06 and CLIF07 (Columbus Large Image Format) as they seemed to be the most promising for testing. Many of the AF datasets had a proprietary video format where the frames and metadata were embedded together in binary files or had multiple frames within a single file. After examining the file format specifications for each dataset, Primordial created Python scripts to extract Tentacle-usable video frames from the original files. The CLIF06 and CLIF07 datasets consisted of a single RAW (Raw image format containing minimally processed data) file for each frame, so the Python script converted each raw file to a Tentacle-usable PNG image file. The CLIF dataset videos had a resolution of ~4000x2600px at 2 frames per second. The CSUAV files each contained numerous frames depending on the video resolution. The high resolution EO (Electro-Optical) video files (4004x2672px) contained 50 frames per file, including telemetry metadata. The infrared video files (640x512px) contained 400 frames per file, including telemetry metadata. Primordial's Python script parsed out the metadata, wrote it to a text file and saved each frame as an individual PNG image. Primordial then used a third-party video creation application to combine the PNGs for each set into a video file (one for each set) that could be used by Tentacle as input.

To verify that the output was correct, Primordial downloaded and installed a government-recommended UAV video viewer application named “GV” (7) to check the correct output for each of the files. It was not possible, however, to export the frames from this application en masse, but only to view the videos and metadata within that specific application. However, this was used as a “sanity check” to verify that Primordial’s parsing of the UAV metadata and video frames matched what was considered the correct output based on the government-supplied tool. This preprocessing step was intended for a first-pass test to prepare the videos for use by Tentacle. In future applications, such preprocessing would have had to be integrated into the core Tentacle system to be used in a production environment. Because each file format is different, a preprocessing solution would need to have been devised for each particular dataset. Also, Primordial needed to use a third-party application to combine the frames into a working video file; for a production setup, this process would have to be implemented in code as well which could add significant development time. An additional concern is that, because the files are so large, there can also be issues with memory capacity on the processing machine. Primordial had several crashes during processing due to a lack of available memory, even on a very robust machine, namely a computer with an Intel i7 quad-processor and 16GB (Gigabyte) RAM (Random Access Memory).

Figure 41 shows some examples of the datasets selected by the Primordial team for use in testing.

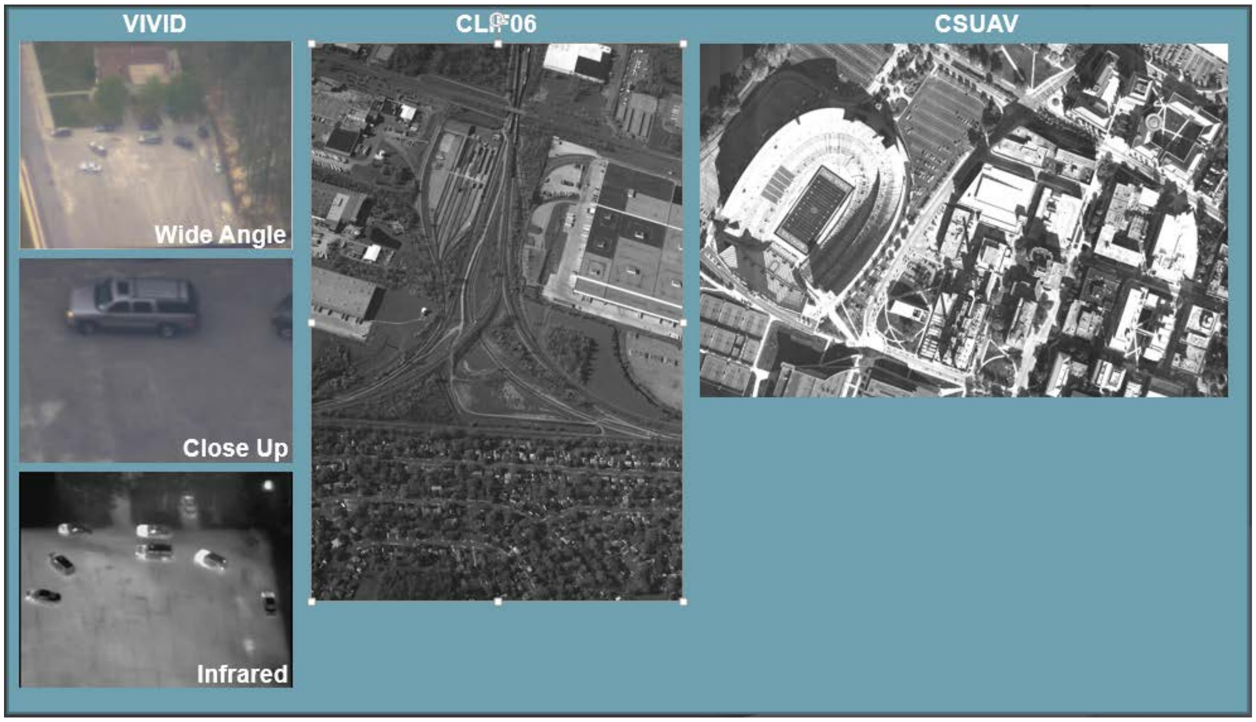


Figure 41. Samples of the VIVID, CLIF and CSUAV Datasets

Alongside Primordial’s examination of AFRL UAV datasets, IntuVision investigated the applicability of vehicle detection and classification algorithms developed previously to a wide area UAV video dataset they received. The data set contained imagery in grayscale and IR (Infrared) format. In this dataset, the grayscale images were captured at two resolutions, e.g. 18549 x 9216 and 26368 x 21248, which results in vehicle dimensions of around 20 x 10 pixels in the first and 25 x 25 pixels in the second resolution respectively. The infrared images are much smaller in resolution, 640 x 512, and the vehicle size is still very small at about 25 x 10 pixels.

Figure 42 shows sample images from the grayscale and infrared data set.

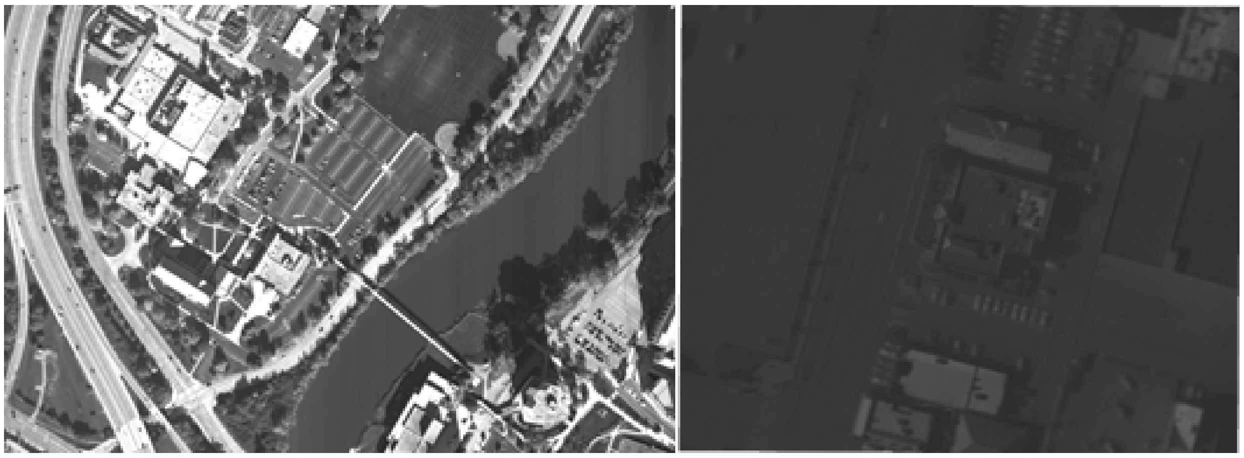


Figure 42. Sample Images from the Grayscale and Infrared Dataset

IntuVision's object classification algorithm uses scale invariant appearance based features including orientation of object edges. Edge orientations in principle describe the object silhouette and constitute an important feature for classification.

Figure 43 illustrates the edges detected (gradients) for different moving objects such as people, and vehicles along with the histograms of edge orientations (Histogram of Oriented Gaussians: HOG). As shown in the figure, HOG features show distinct differences for vehicles and people. These gradients are usually calculated over kernels that are multiple pixels in size and pick up the edge features of an object.

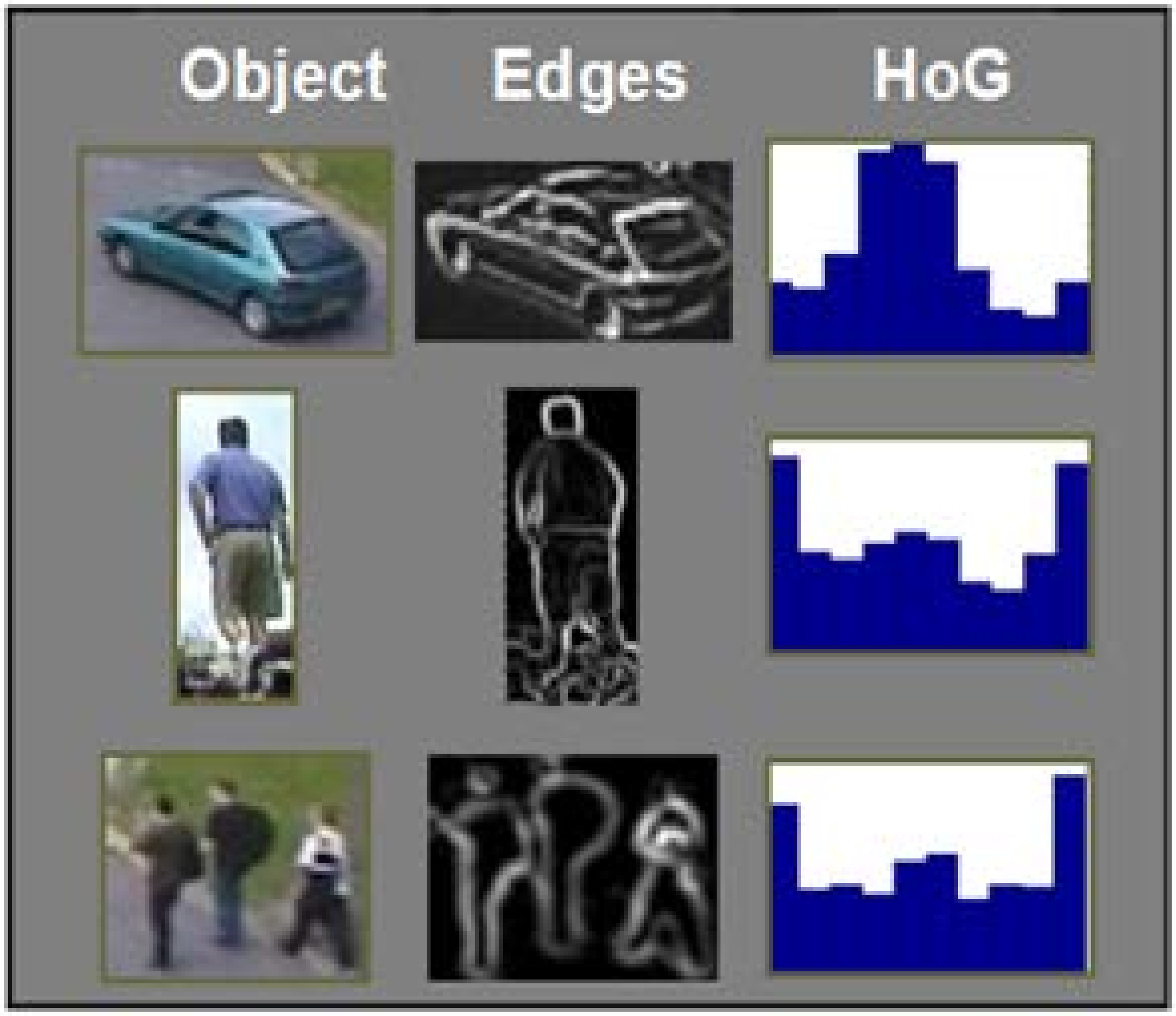


Figure 43. Objects, Their Detected Edges and Histogram of Gradients

In the UAV data collection, however, the objects of interest are very small, hence their edge features cannot be detected as clearly. The minimum object size in an image for reliable classification is roughly 50x40 pixels.

In **Figure 45** cropped regions of images given in **Figure 44** with vehicles are shown along with the detected edge. For the grayscale image on the left, the outer edges of the vehicles can be observed in the gradient image for some of the cars while for the infrared image the vehicles can be barely seen in the gradient image. The vehicle classification requires better gradients for classification.

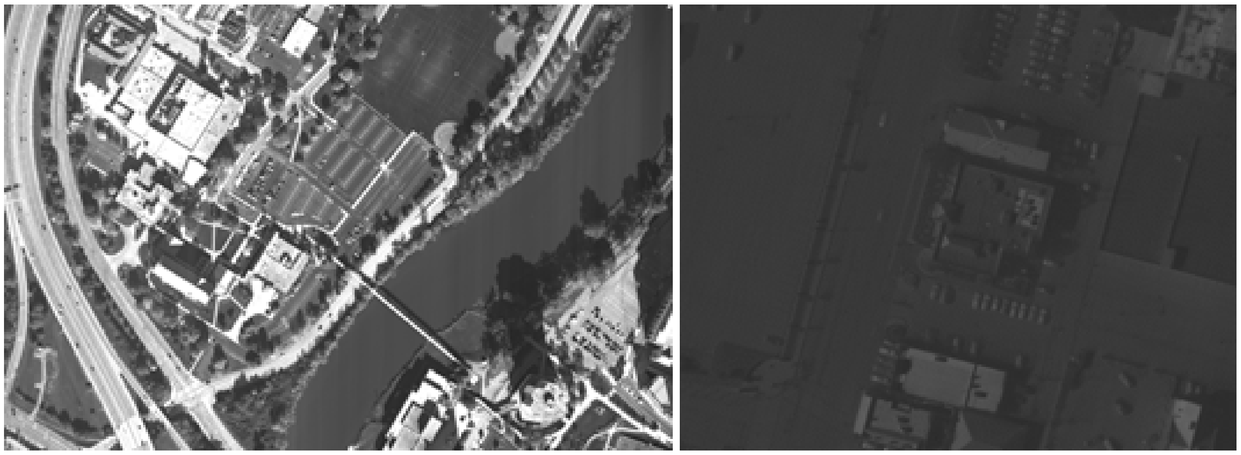


Figure 44. Entire Frames. Grayscale (left) and Infrared (right)

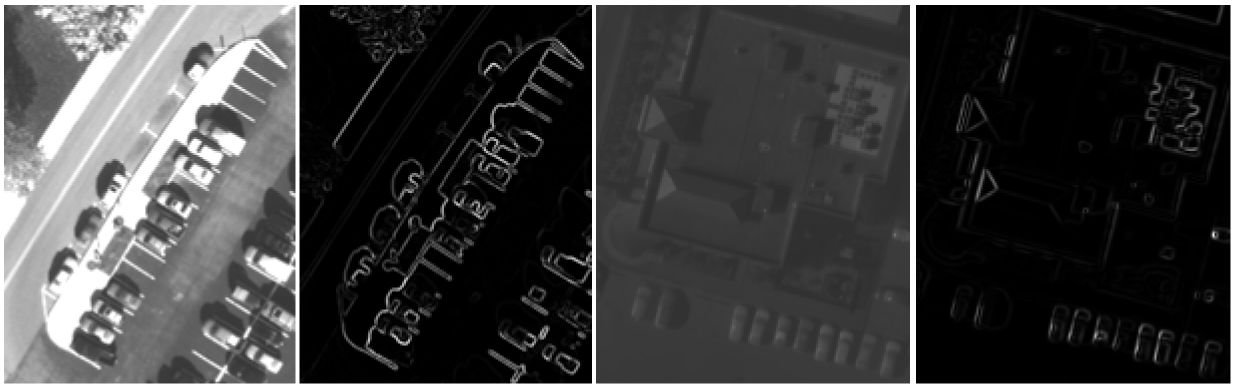


Figure 45. Cropped Regions of Images Including Vehicles, and Their Corresponding Gradients, in the Grayscale and Infrared Dataset

Since the vehicle classification was not applicable to the data set, IntuVision also investigated the use of a frame-based center surround saliency measurement to identify regions of interest in an image for use in tracking. Center-surround saliency mimics the human visual attentional mechanism to make objects that are most different from their surroundings "pop-out" from background clutter. In this application, IntuVision used the pixel intensity information of the image to calculate the saliency and gradient strength values; this helped to clean up the saliency and obtain a binary saliency mask. This use of gradient is different from the classification method, as the goal here is to identify only the presence or absence of an object. **Figure 46** and **Figure 47** show two different regions from the grayscale image from **Figure 44** along with the salient detections marked by blue rectangles. In **Figure 46**, the sections are: (Left) Input image, (Middle) Saliency image, (Right) Saliency binary mask with vehicle detection regions marked.

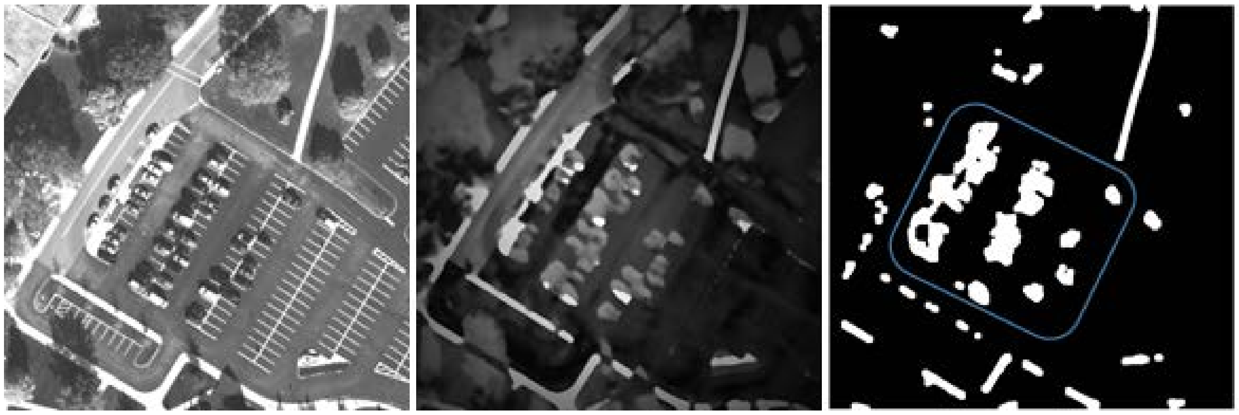


Figure 46. Use of Saliency to Highlight Areas of Interest

The saliency image of the parking lot in **Figure 46** highlights the vehicles and the pathways, as they are very distinct from the surrounding. The saliency binary mask shows the detected vehicles, highlighted by the blue rectangle. The saliency measurement is object agnostic, and responds to regions that shows difference from its surroundings. Even though this gives rise to some false detections on the high-contrast curb sections, further filtering can be applied to eliminate those detections. For the highway view in **Figure 47**, the saliency measure correctly detects the presence of the 6 vehicles. In **Figure 47**, (Left) Input image, (Middle) Saliency image, (Right) Saliency binary mask with vehicle detection regions highlighted.

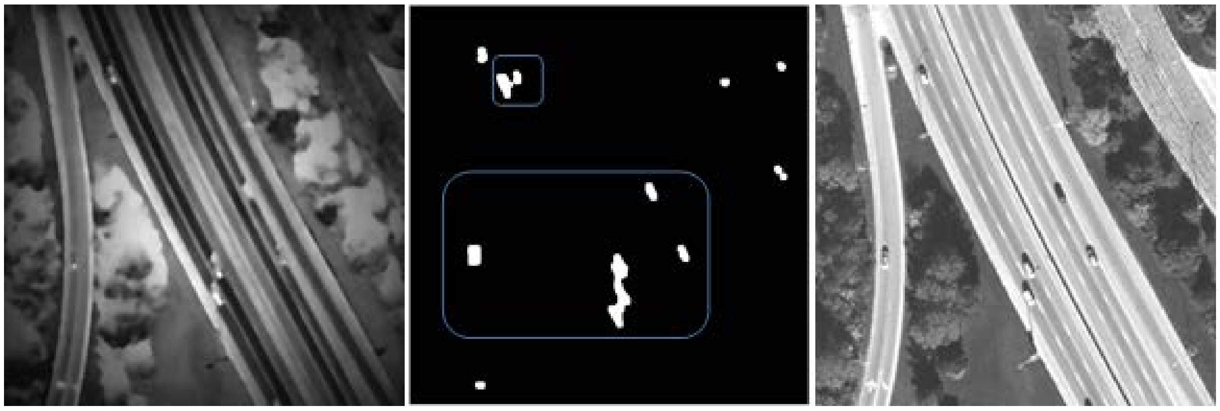


Figure 47. Use of Saliency to Highlight Areas of Interest

Figure 48 shows this method as applied to four of the test videos for tracking purposes.

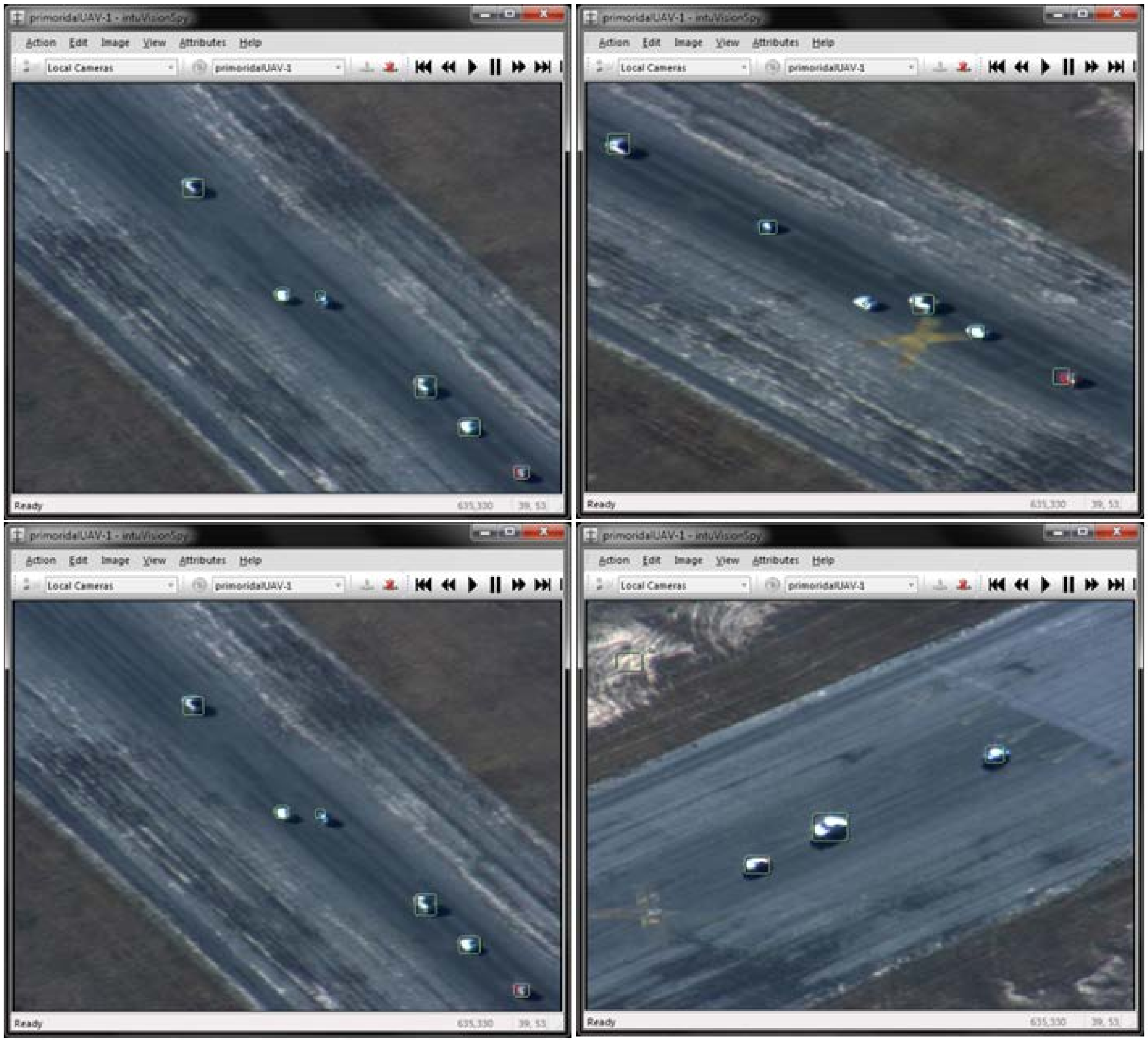

primoridalUAV-1 - intuVisionSpy
Action Edit Image View Attributes Help
Local Cameras
primoridalUAV-1
Ready

Figure 48. Target Tracking Using Background Subtraction and Center-Surround Saliency

In order to better focus the entire team's work effort to make the most of the Tentacle system, Primordial, IntuVision and CMU decided to target the VIVID UAV dataset as the standard video focus for the final demonstration for the project. As a result, Primordial selected videos from the set in order to best showcase Tentacle's capabilities in a real-world scenario. The following figures show videos that were chosen with an explanation of what the video represents in terms of UAV targeting (**Figure 49**, **Figure 50**, **Figure 51**, **Figure 52**, **Figure 53**, **Figure 54**, **Figure 55**, **Figure 56**, **Figure 57**, **Figure 58**, **Figure 59**, **Figure 60**, **Figure 61**, **Figure 62**).

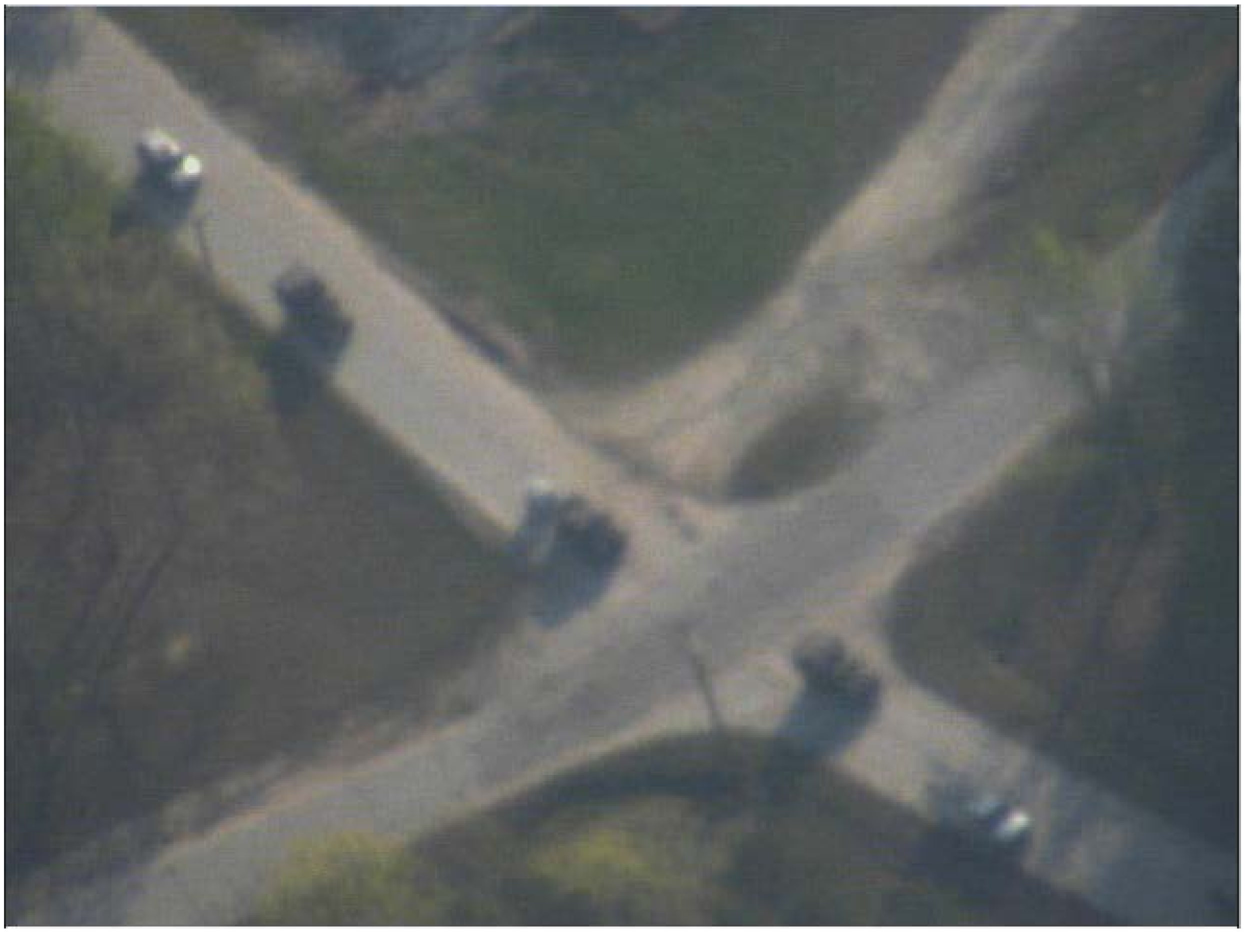


Figure 49. Clustering (VIVID)

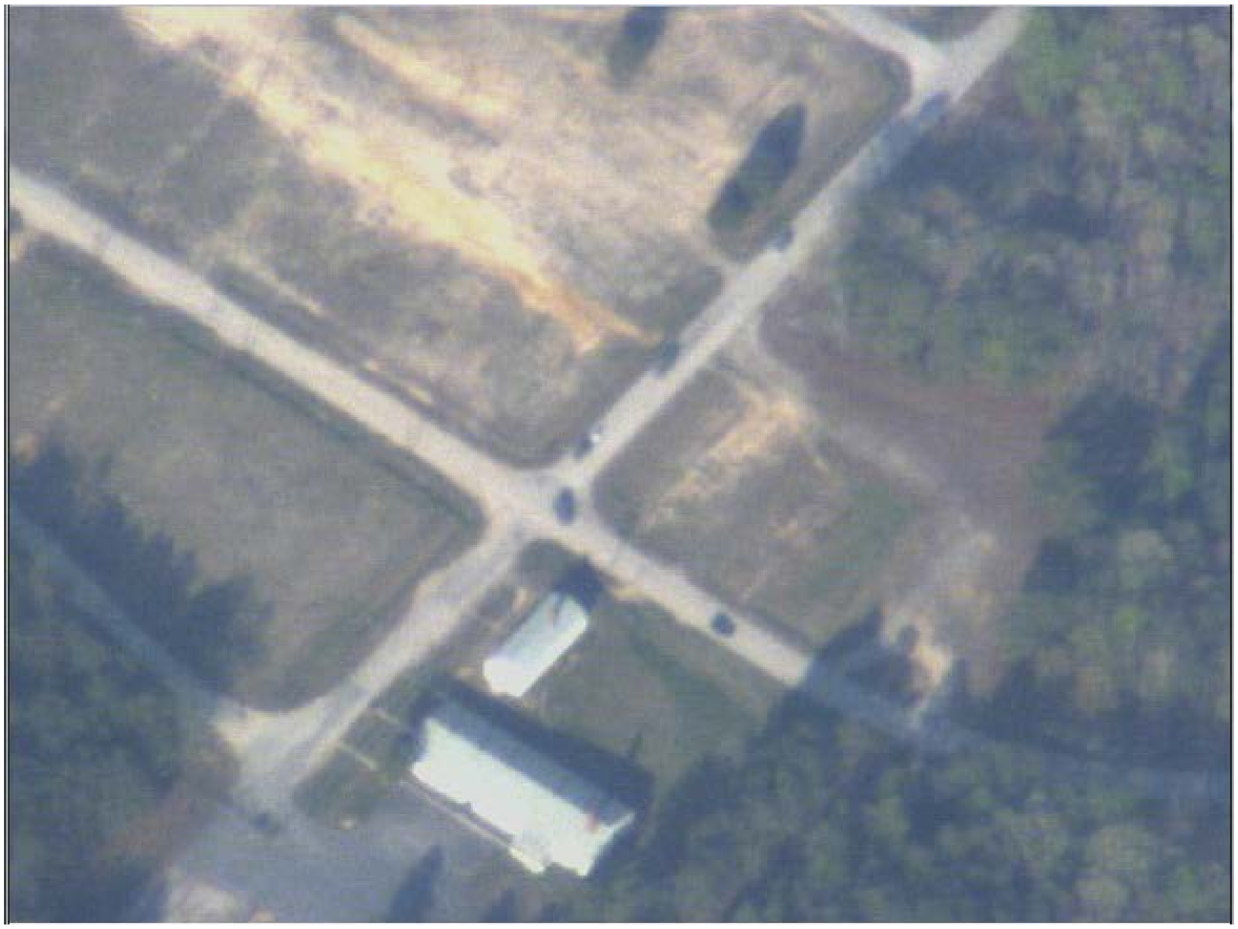


Figure 50. Clustering, Proximity (VIVID)

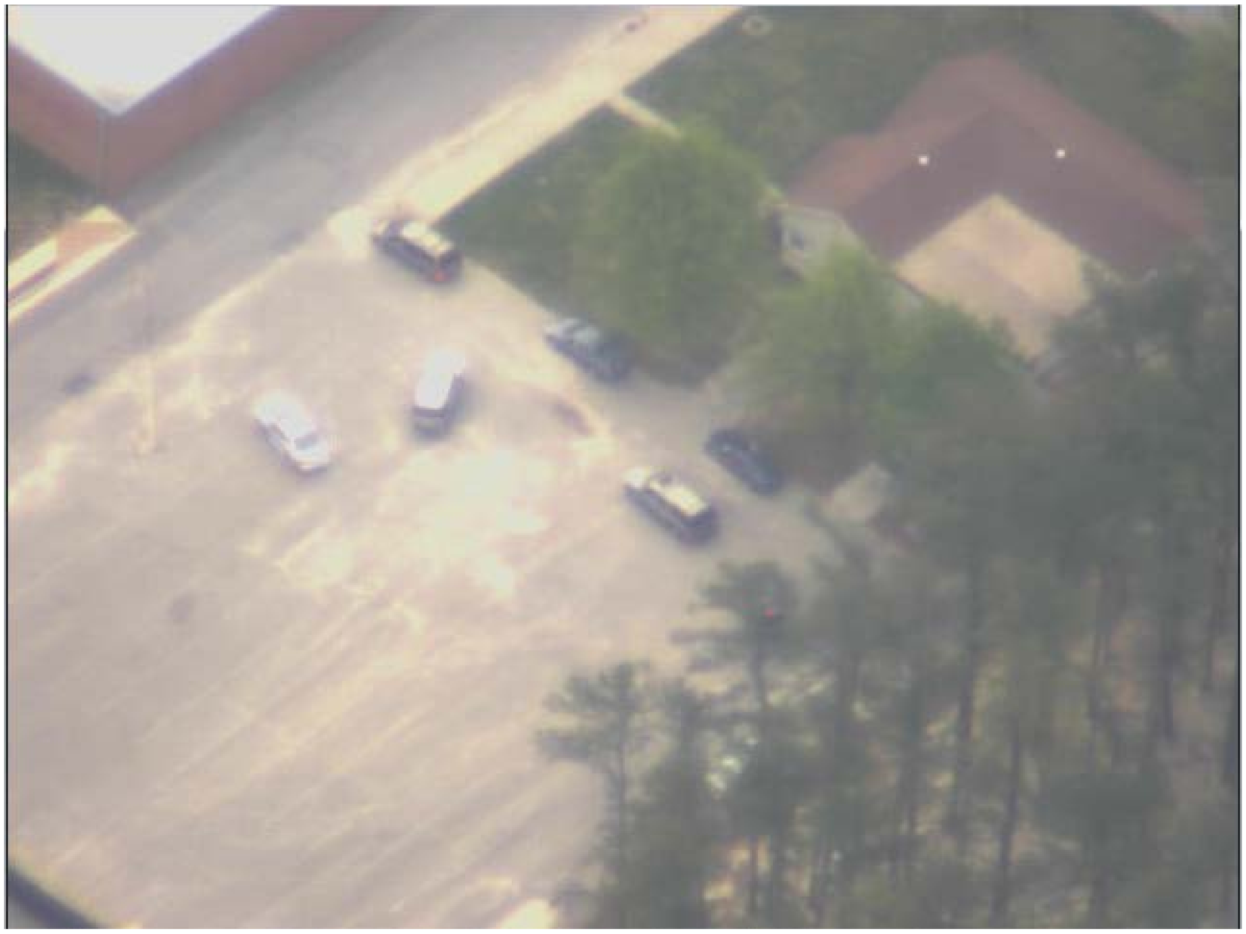


Figure 51. Loitering, Proximity, Cluster (VIVID)

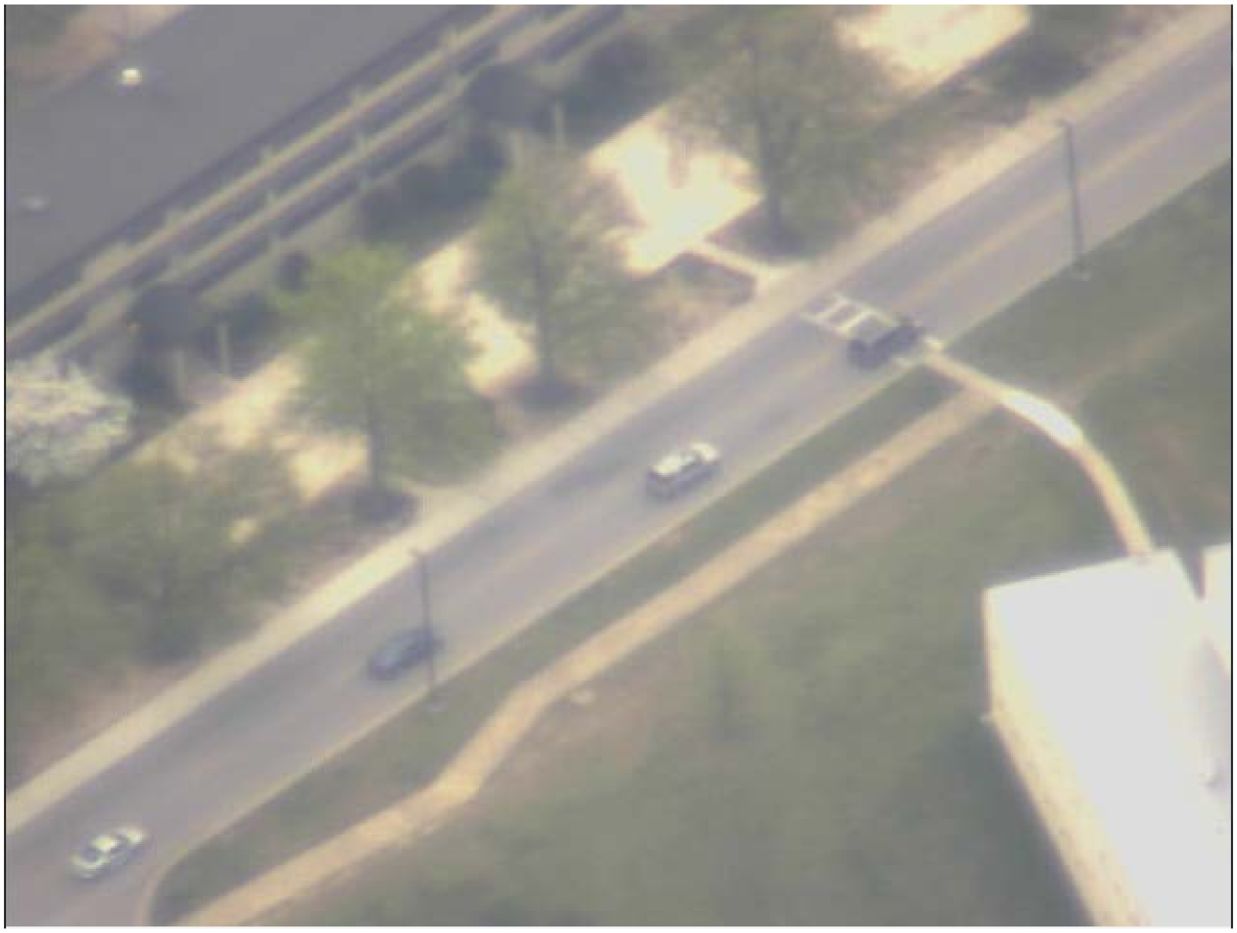


Figure 52. Cross, Cluster, Proximity (VIVID)

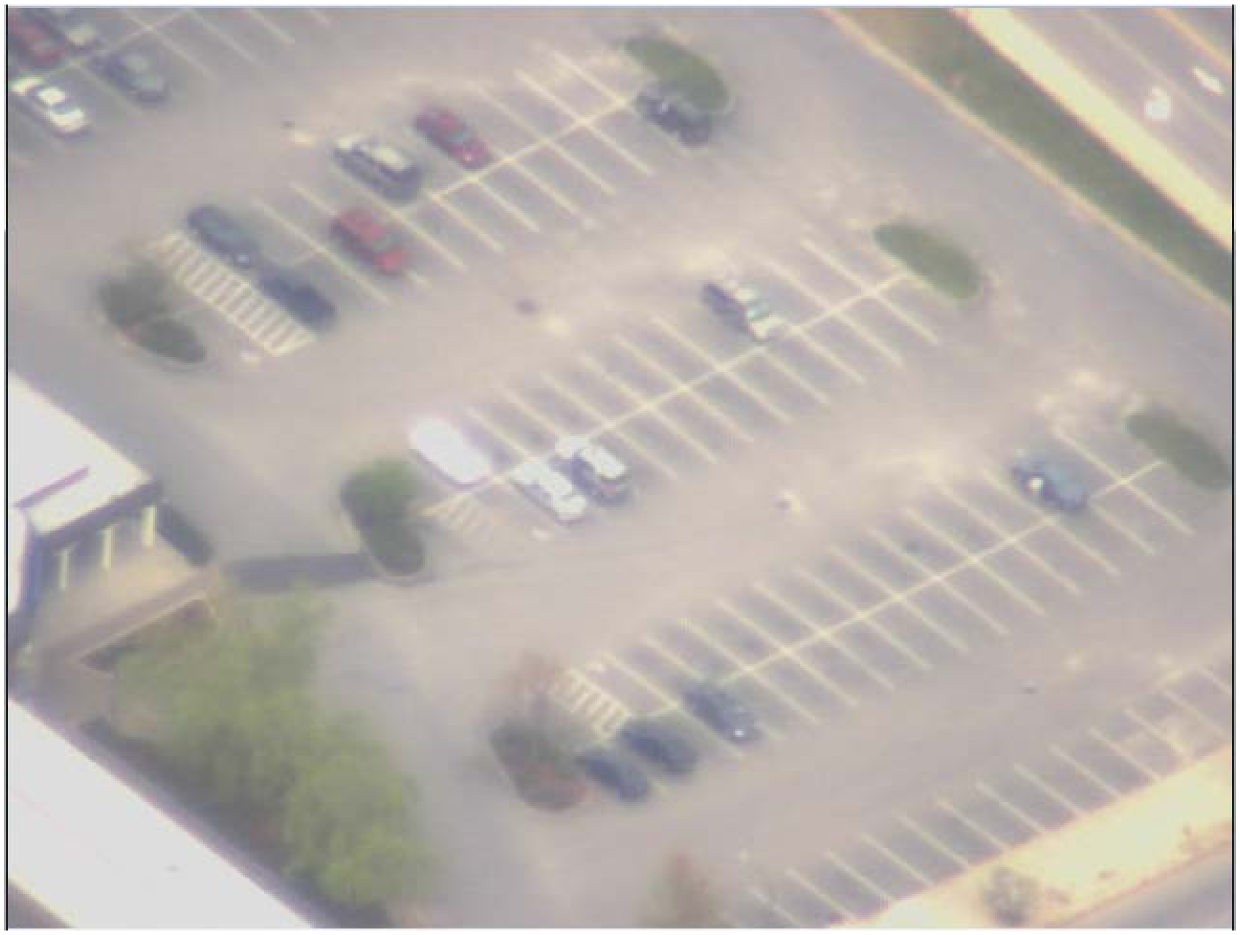


Figure 53. Loitering, Cluster (VIVID)

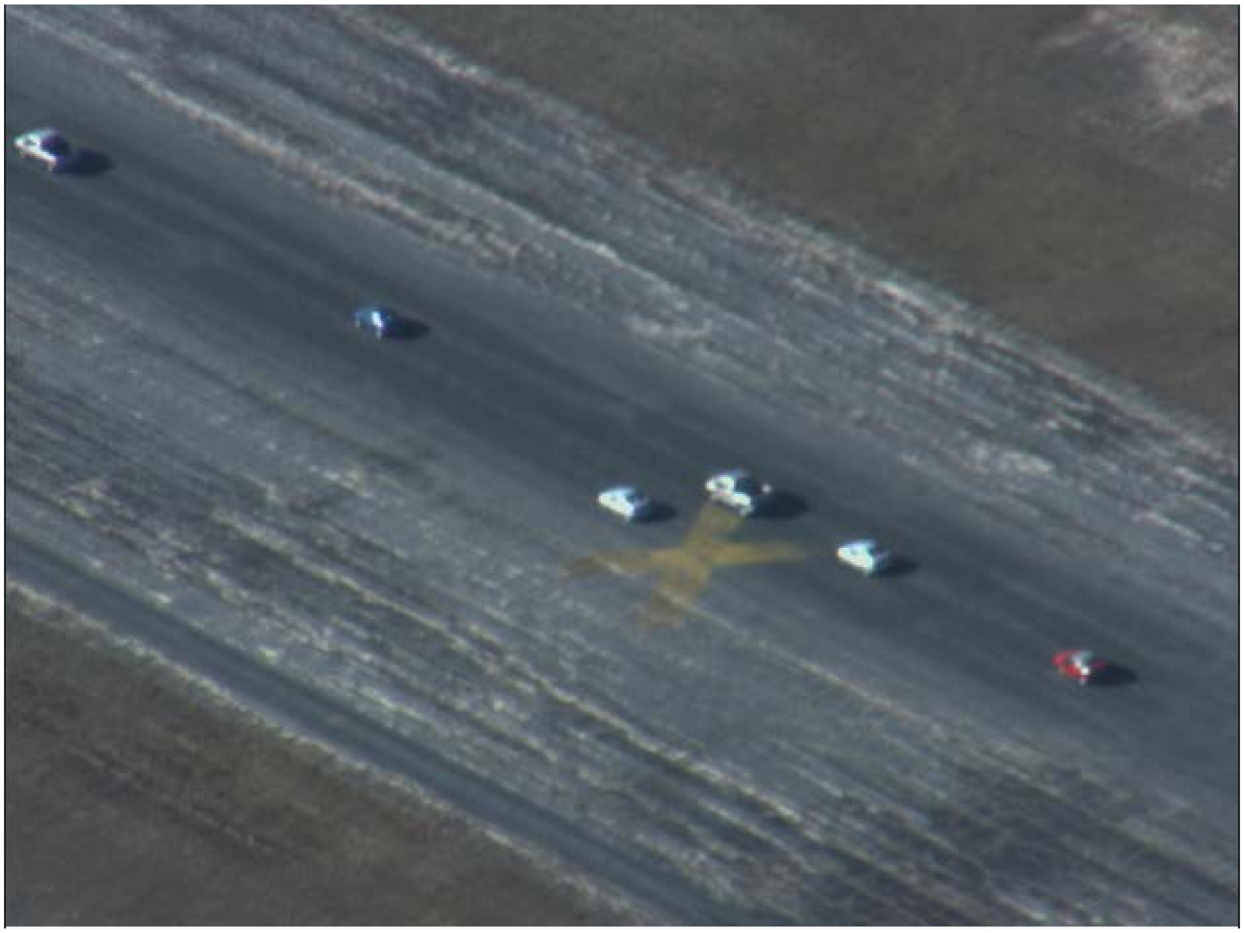


Figure 54. Cluster (clearer quality) (VIVID)

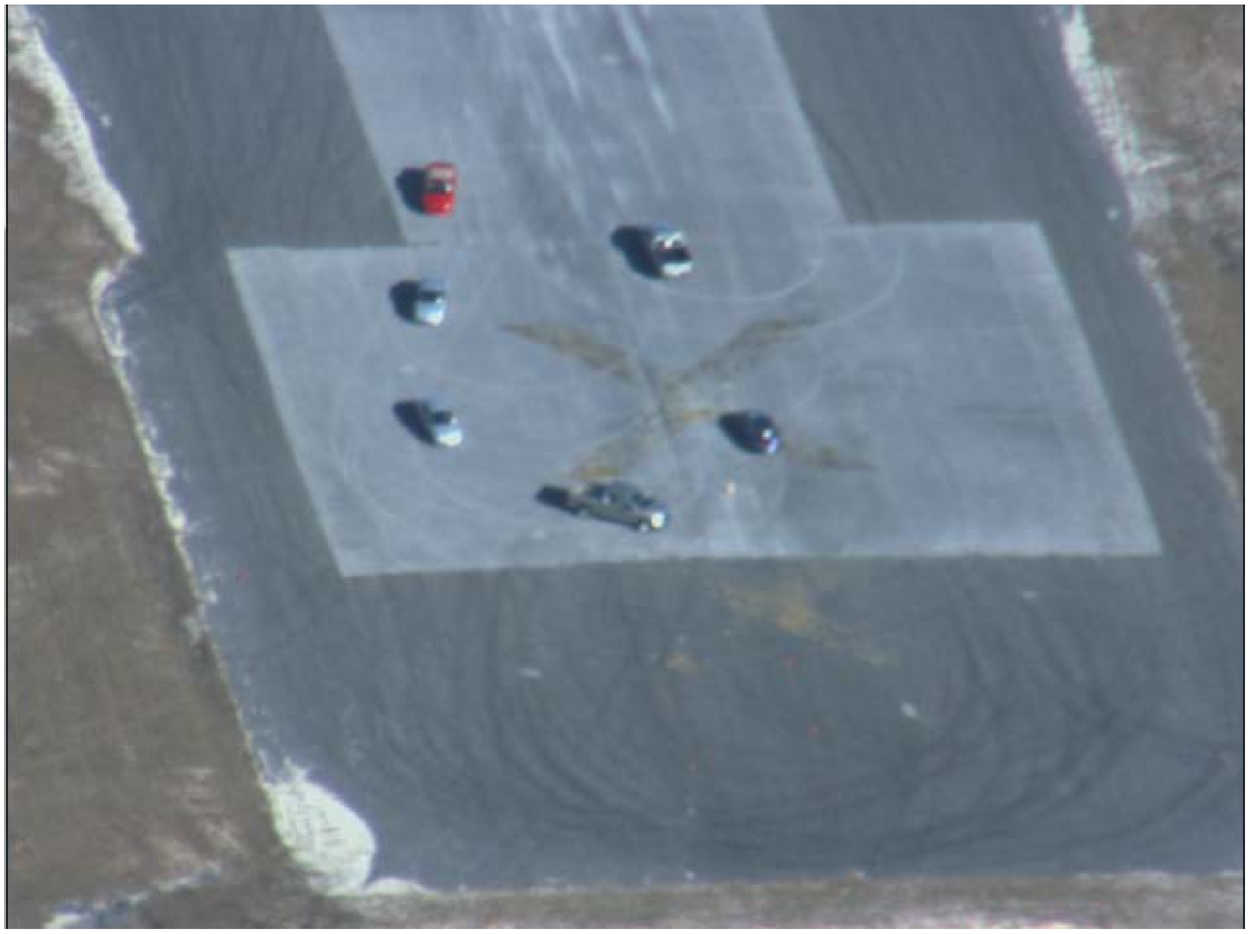


Figure 55. Cluster

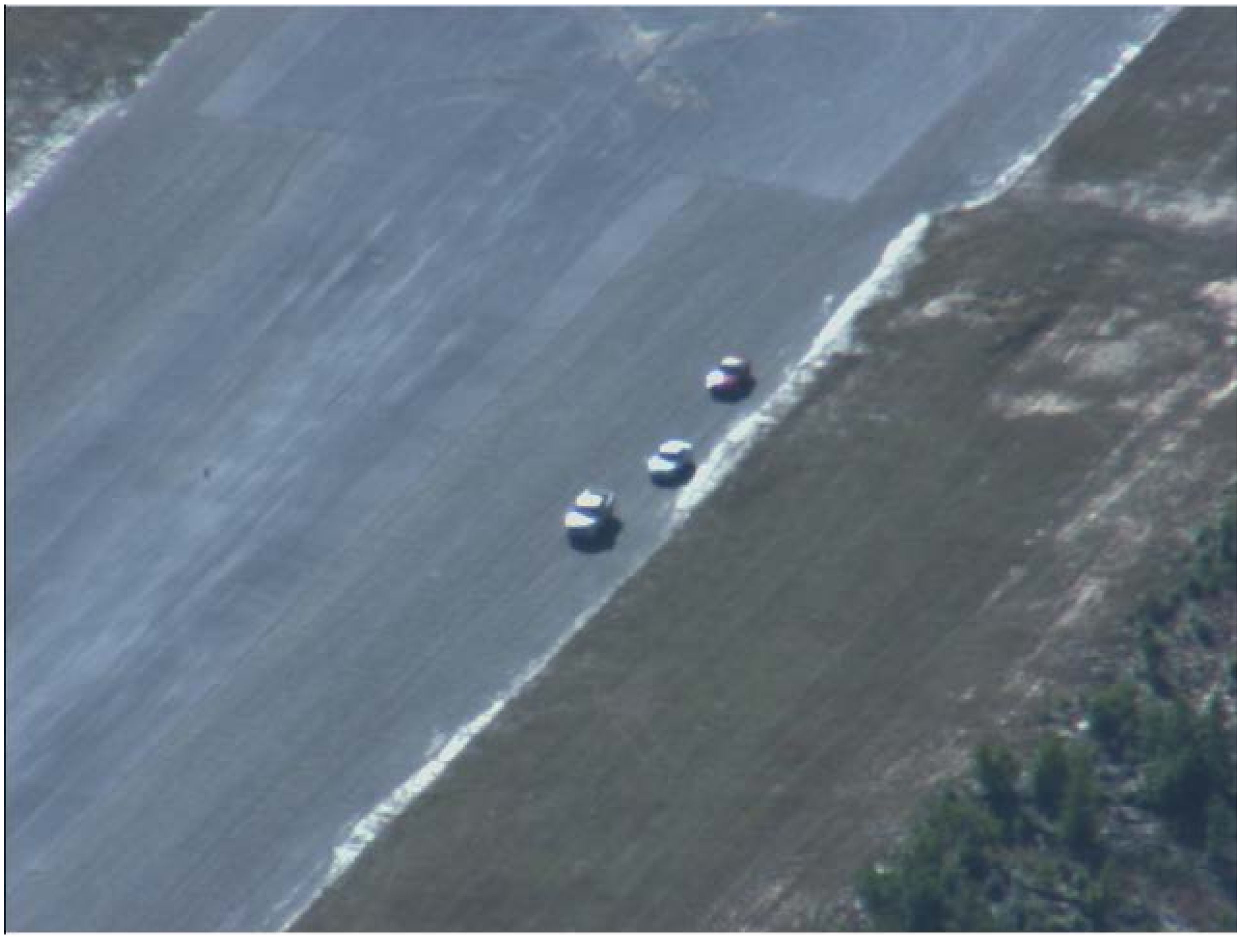


Figure 56. Loiter, Cluster (VIVID)

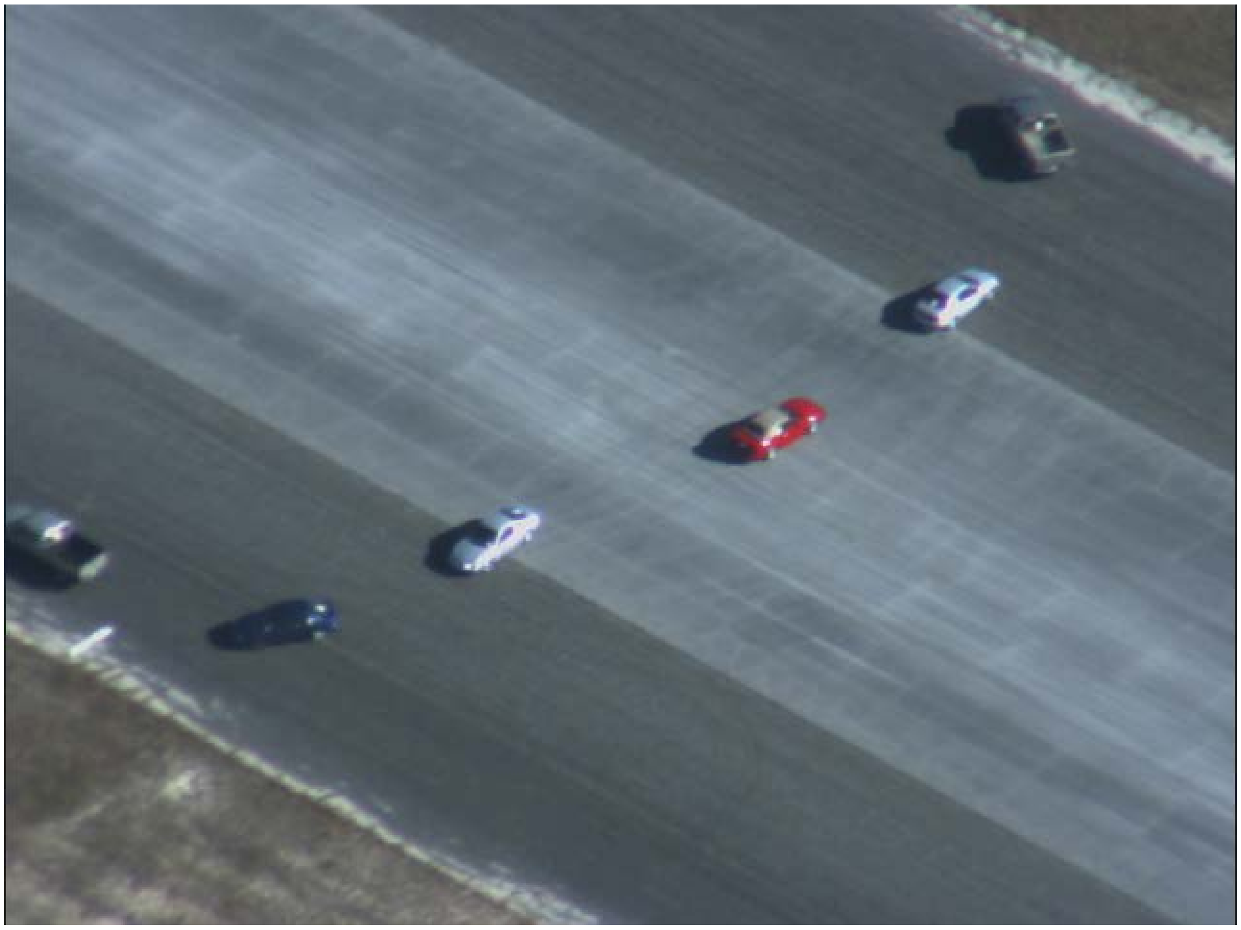


Figure 57. Cluster

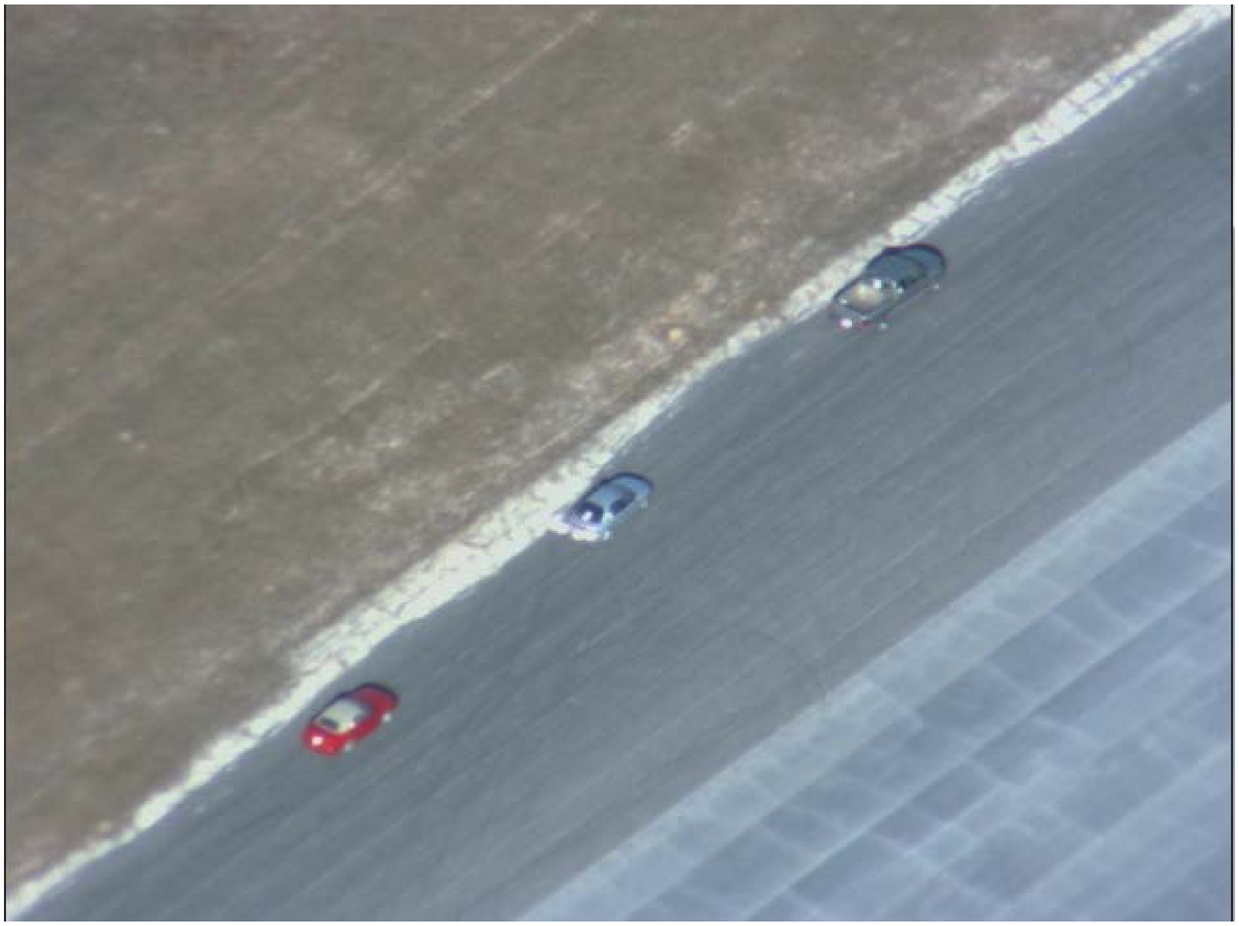


Figure 58. Cluster

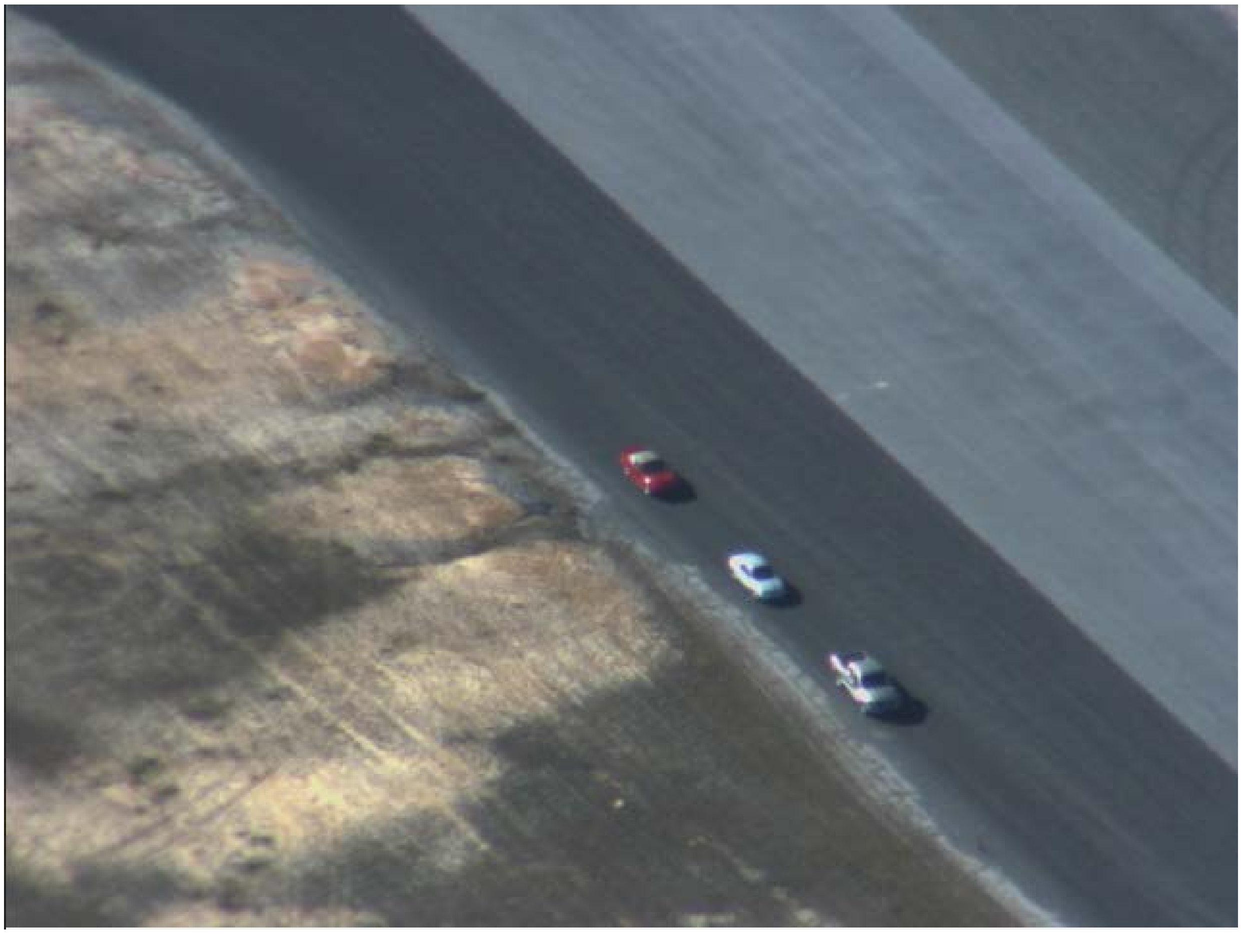


Figure 59. Cluster

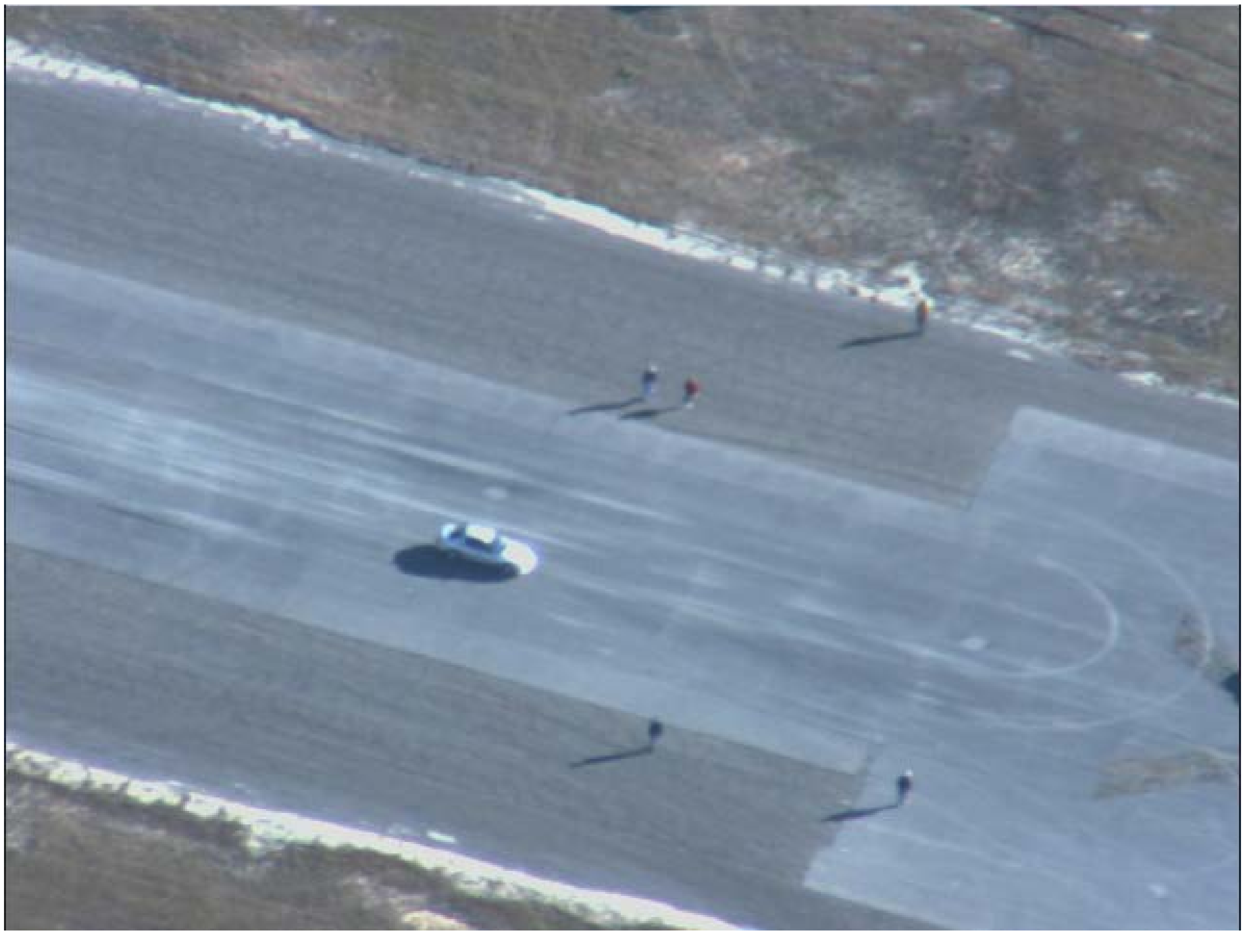


Figure 60. People and Cars Clustering, Loitering

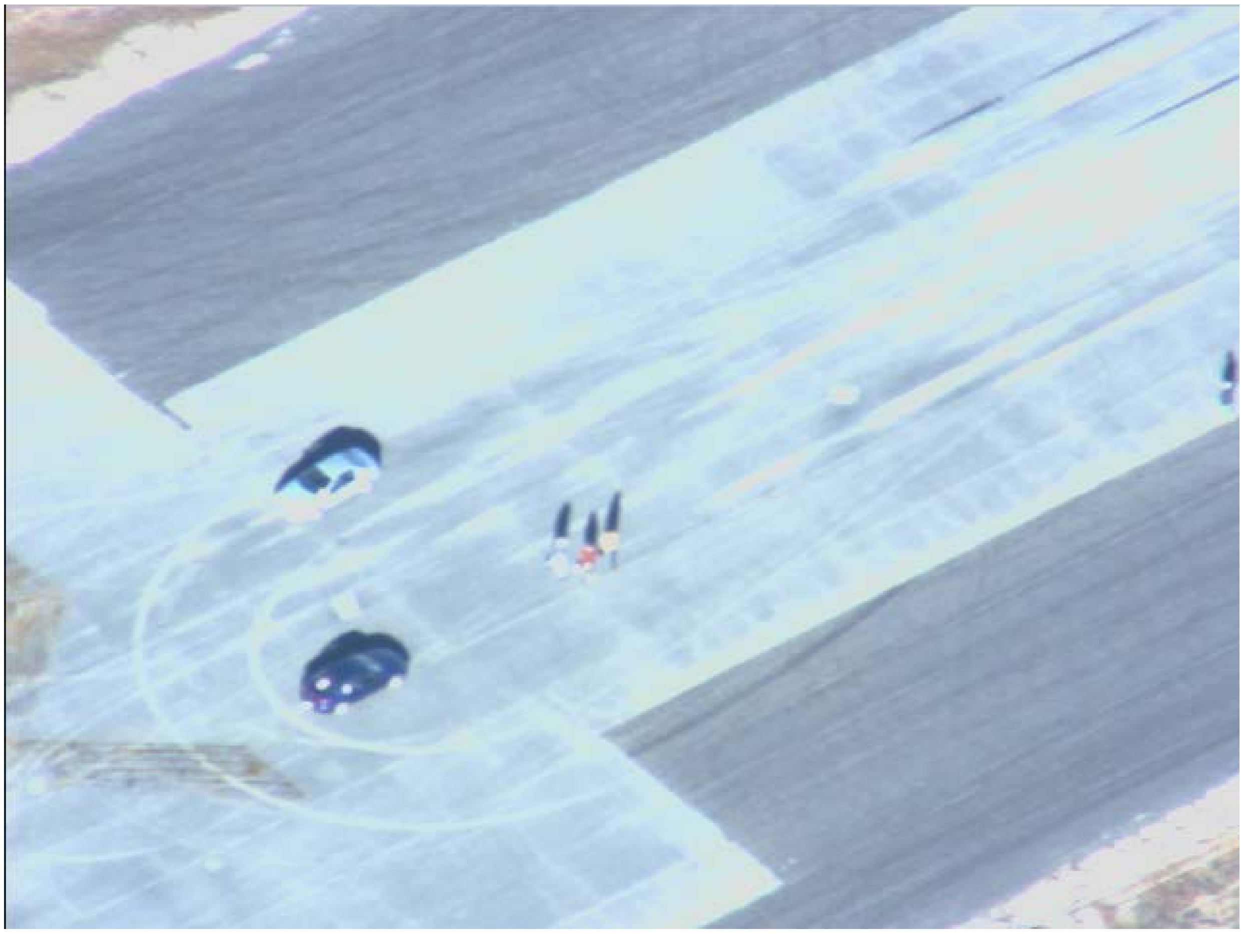


Figure 61. People and Cars Clustering, Loitering (VIVID)

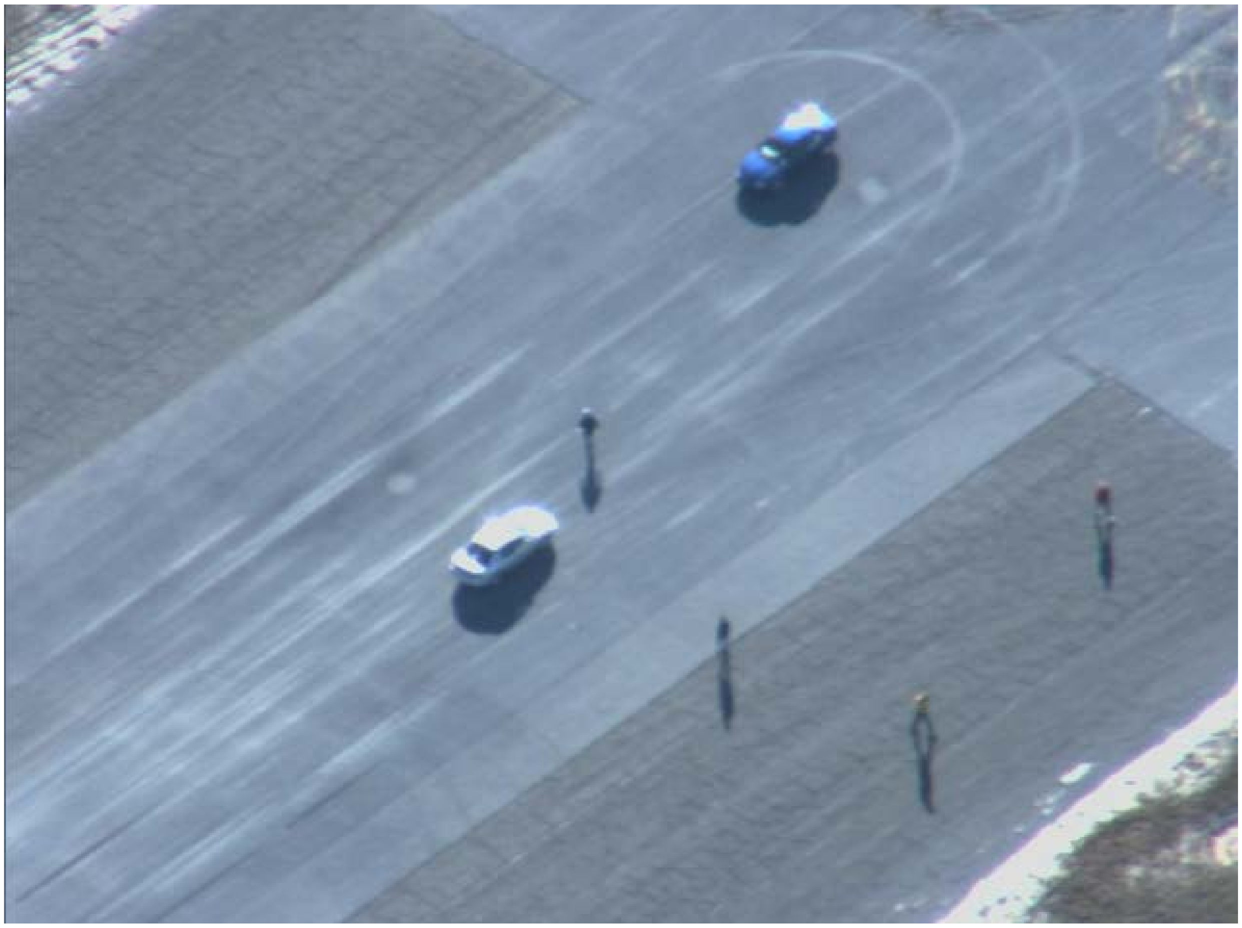


Figure 62. People and Cars Clustering, Loitering (VIVID)

The corresponding VIVID video files for the above figures are as follows:
Figure 49: VIVID3_4\d4-apr20p3-1_6\Pass3\Clip_005\V4V100004_005.avi
Figure 50: VIVID3_4\d4-apr20p3-3_6\Pass3\Clip_035\V4V100004_035.avi
Figure 51: VIVID3_4\d4-apr21p1-1_9\Pass1\Clip_003\V4V100007_003.avi
Figure 52: VIVID3_4\d4-apr21p1-1_9\Pass1\Clip_004\V4V100007_004.avi
Figure 53: VIVID3_4\d4-apr21p1-1_9\Pass1\Clip_005\V4V100007_005.avi
Figure 54: VIVID3_4\VIVID_DS3-D1-2_3\Day1Scenario4_5\Clip_005\V3V100003_005.avi
Figure 55: VIVID3_4\VIVID_DS3-D1-2_3\Day1Scenario4_5\Clip_007\V3V100003_007.avi
Figure 56: VIVID3_4\VIVID_DS3-D1-2_3\Day1Scenario6\Clip_003\V3V100004_003.avi
Figure 57: VIVID3_4\VIVID_DS3-D1-2_3\Day1Scenario6\Clip_004\V3V100004_004.avi
Figure 58: VIVID3_4\VIVID_DS3-D1-2_3\Day1Scenario6\Clip_005\V3V100004_005.avi
Figure 59: VIVID3_4\VIVID_DS3-D1-2_3\Day1Scenario6\Clip_006\V3V100004_006.avi
Figure 60: VIVID3_4\VIVID_DS3-D1-3_3\Day1Scenario7-1\Clip_008\V3V100005_008.avi
Figure 61: VIVID3_4\VIVID_DS3-D1-3_3\Day1Scenario7-1\Clip_009\V3V100005_009.avi
Figure 62: VIVID3_4\VIVID_DS3-D1-3_3\Day1Scenario7-1\Clip_009\V3V100005_010.avi

The current state of the system required the different phases of UAV target detection to be processed separately: First by CMU’s SDK (for tracking targets), then by IntuVision’s SDK (for metadata extraction), then by the Tentacle detection system and then finally the core Tentacle server. At this stage, there was still some disconnect between the CMU SDK and the rest of the Tentacle system (which was otherwise integrated). This was largely due to the fact that the CMU SDK had to be located on a separate platform and operating system. In order to better integrate the two parts, Primordial began investigating IPC (inter-process communication) methods which would allow passing of data from the CMU SDK platform to the rest of the Tentacle system in a more straightforward manner. Primordial provided notes on this research to CMU for evaluation and suggestions. Primordial also set up a virtual machine with an appropriate Linux system and the CMU SDK for use in the final demonstrations. While CMU was reviewing Primordial’s IPC findings, Primordial implemented an adapter class to allow CMU’s SDK output to be streamed directly into the Tentacle system as if it were a normal camera feed. Using this class, the CMU SDK would track objects, then write out the video frames and bounding boxes for entities to files which would be ingested by the adapter class through a network mapped location. This would act as a fall-back solution in the case that an appropriate IPC solution could not be found.

IntuVision continued work on improving their Support Vector Machine object classification for optimal performance with the acquired UAV data sets. They experimented with different gradient strength thresholds for the corresponding oriented gradients to contribute to the detected histogram features. They also experimented with the size the detections are resized to before features are calculated. **Figure 63** shows the False Acceptance –Missed Detection for the before (red curve) and after (blue curve) scenarios for vehicle detection. The area under the curve decreased from 0.12 to 0.10, with the bulk of the improvement occurring in the low false acceptance region. IntuVision provided updated classification models as part of the next SDK delivery to Primordial.

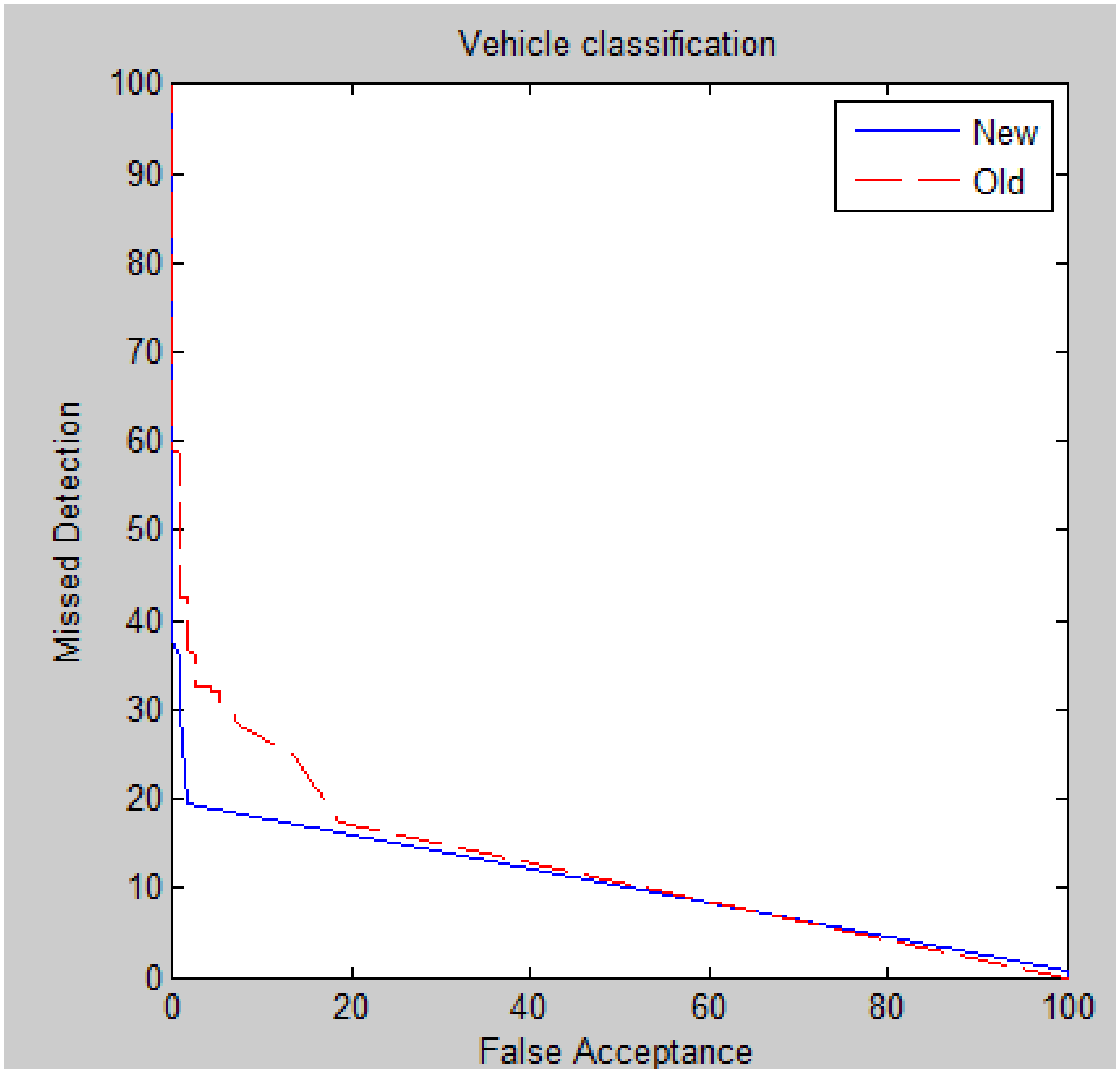


Figure 63. Improvements to SVM Object Classification, Vehicle Classification Example

To assist in metadata extraction, IntuVision requested that Primordial provide alpha-channel target masks in the video frames being fed into their SDK from the tracking modules created by CMU. Primordial modified the CMU SDK tracking application using the OpenCV library to output video frames including an alpha channel where an opacity value of 0 indicated a pixel intersecting with a target location. An illustration of this output is shown in **Figure 64** and **Figure 65**.

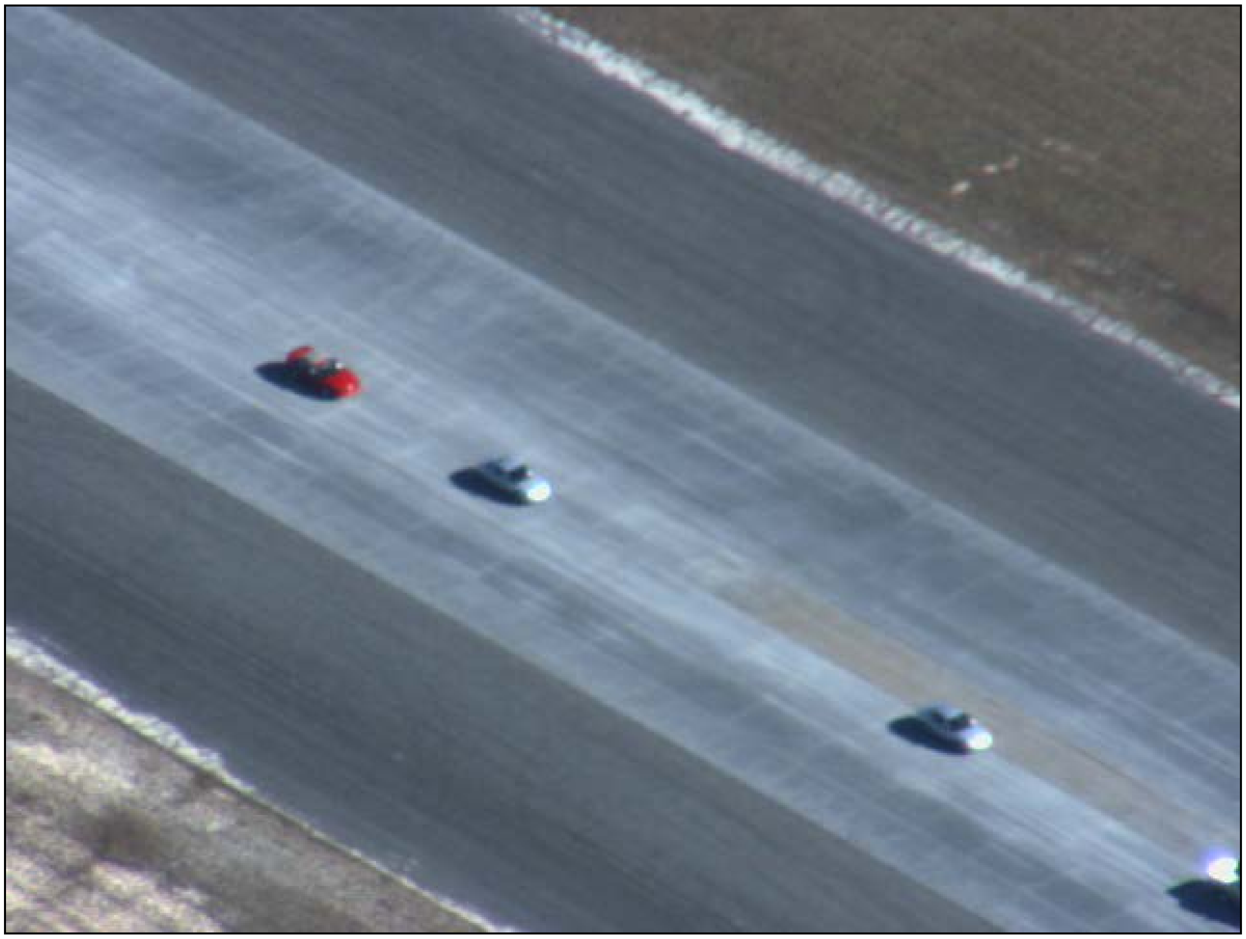


Figure 64. Initial Video Frame

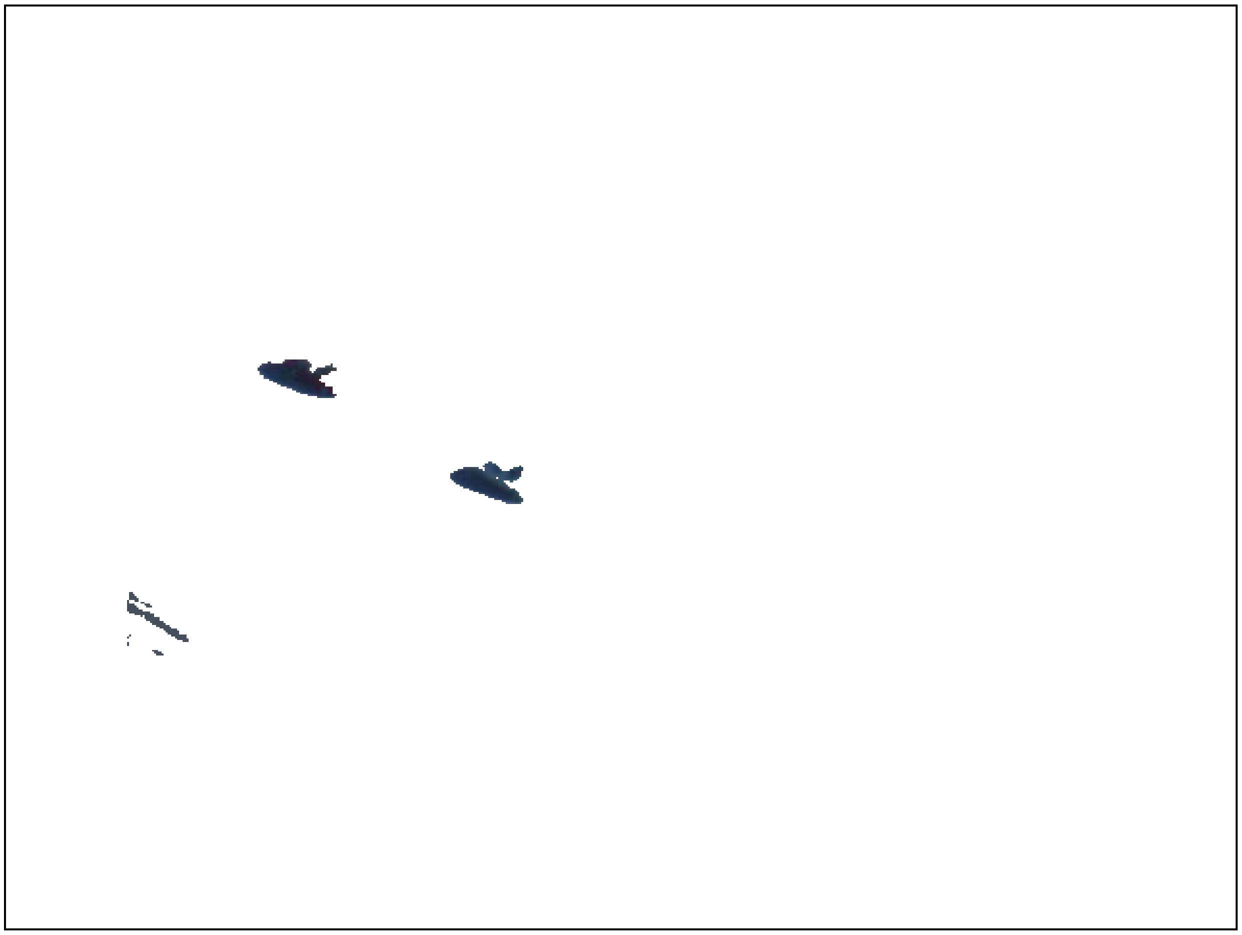


Figure 65. The Resulting Processed Image Including the Target Mask Alpha Channel

IntuVision made internal modifications to their SDK to support the new alpha mask format and delivered an updated version, which Primordial integrated into the Tentacle system’s detection application.

Continuing with earlier work to integrate the software from all of the Tentacle team partners, Primordial experimented with using the Redis.IO keystore system for passing frame information between the CMU SDK and the main Tentacle system. They set up several test applications on both Linux and Windows machines and were able to successfully pass data between the various disparate parts of the system as it would happen if integrated into the main Tentacle structure. As an alternative, Primordial created a similar structure using RabbitMQ, a third party message passing library. They tested similar scenarios to compare the library with that of Redis.IO. Both systems worked well and the results were satisfactory. However, further work on this aspect of the project by Primordial was halted to focus efforts elsewhere due to the upcoming project deadline.

IntuVision continued work on their SDK, improving general stability, processing and accuracy. Additionally, they added other features which would help to improve classification for Tentacle.

The first of these was improved accuracy of their SVM training/classification subsystem. This would be potentially able to use a larger training set of UAV videos with foreground pixel masks when available, and provide more robust classification results than the previously developed BIM classification models.

A second feature added to the SDK was template matching. In images captured from a high altitude, vehicles are usually visible only as small blobs. To aid the detection of such vehicles, IntuVision investigated the use of template matching. As the vehicle's top profile is rectangular, IntuVision chose a square edge template and matched it (using correlation matching) to the gradient magnitude image. **Figure 66** shows an example with the view of a highway with 6 vehicles clearly visible in the scene. The subsections of the figure are: (Left) View of a highway. (Middle) Template matching response. (Right) Vehicle detections overlaid onto the original image. The template matching response has peaks corresponding to the location of these vehicles and these detections are overlaid onto the original image. All 6 vehicles are correctly detected, there is one false positive for the street light pole and the technique also picked up the shadow of the vehicle that has just exited the frame.

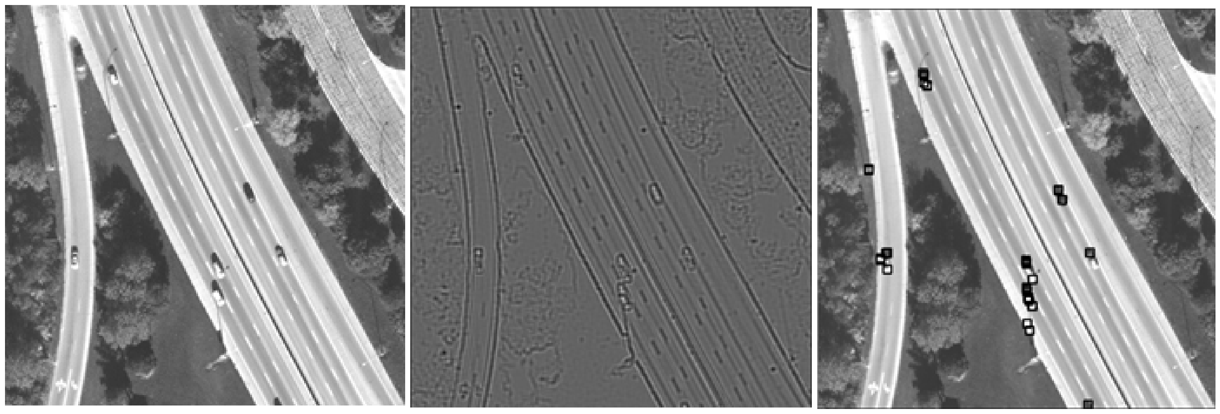


Figure 66. Template Matching

The moving target tracking task was the most difficult task of the effort, but the team achieved good results, despite some limitations.

3.5 Improve Metadata Extraction

In Phase II of the project, the Tentacle team sought to improve on the initial metadata extraction capabilities developed during Phase I. Initially, IntuVision began work on improving color metadata extraction by adding higher-granularity color histograms. A second task was to add more color regions for associating colors with different parts of detected entities (left, right, upper, lower, overall). These changes were intended to also help improve entity correlation during the detection process. In addition to this, IntuVision implemented the ability for their SDK to extract metadata from pre-extracted UAV entities produced by the CMU SDK's tracking output. The initial changes were promising. A test video used for reference, which included a woman wearing a white shirt and tan pants, was much more accurately represented after the changes made than it was during previous stages (see **Figure 67**).

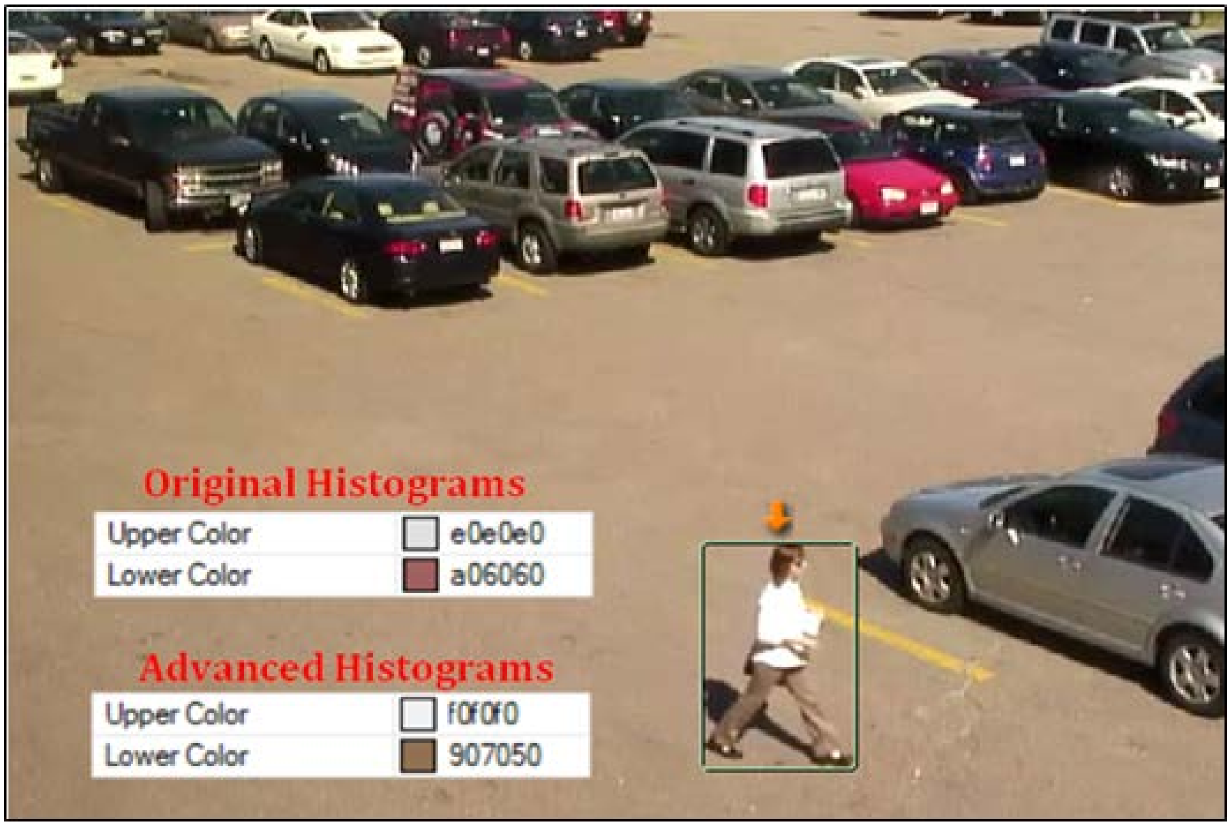
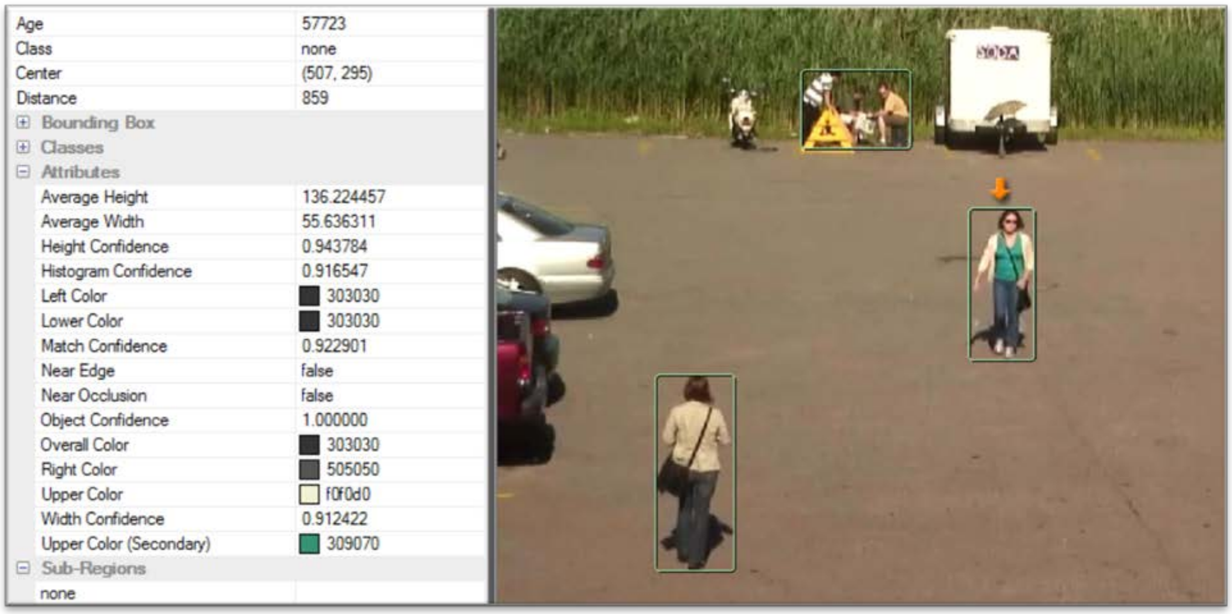


Figure 67. Color Extraction Test Video Used by IntuVision for Implementation Comparison

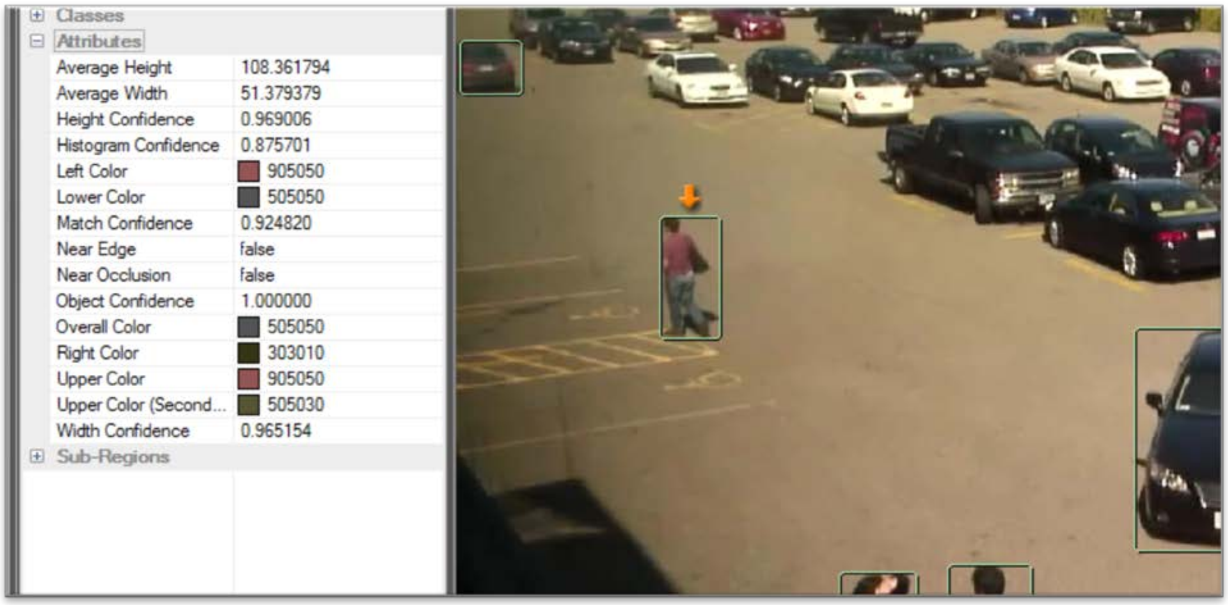
Notice that the woman wearing the white shirt and tan pants was more accurately represented with the advanced histograms. Previous color representations were based on the 2-part 64–bin object color histograms used for frame correlation. The mode of these color histograms were used as the metadata for Upper and Lower color of the object. In order to provide a more accurate color representation, IntuVision implemented a new histogram model for tracked objects which uses bins: 8 bins per color channel, resulting in 512 total color bins. In addition to the previous two object regions (upper and lower), the new model also provides three additional object regions to generate the object colors: Left, Right, and Overall object color. As seen in **Figure 67**, this scheme results in more accurate color metadata produced for the tracked objects.

For the Upper and Lower colors, IntuVision examined the histograms to see if other significantly different and dominant colors were present. In cases where a person was wearing multiple colored clothing, IntuVision provided Upper Secondary and Lower Secondary colors, as illustrated in **Figure 68** and **Figure 69**. These colors were determined by finding the first color bin value that is at least 75% of the maximum bin, and has a hue at least 45 degrees away.



Age	57723
Class	none
Center	(507, 295)
Distance	859
⊞ Bounding Box	
⊞ Classes	
⊟ Attributes	
Average Height	136.224457
Average Width	55.636311
Height Confidence	0.943784
Histogram Confidence	0.916547
Left Color	⬛ 303030
Lower Color	⬛ 303030
Match Confidence	0.922901
Near Edge	false
Near Occlusion	false
Object Confidence	1.000000
Overall Color	⬛ 303030
Right Color	⬛ 505050
Upper Color	⬜ f0f0d0
Width Confidence	0.912422
Upper Color (Secondary)	🟩 309070
⊟ Sub-Regions	
none	

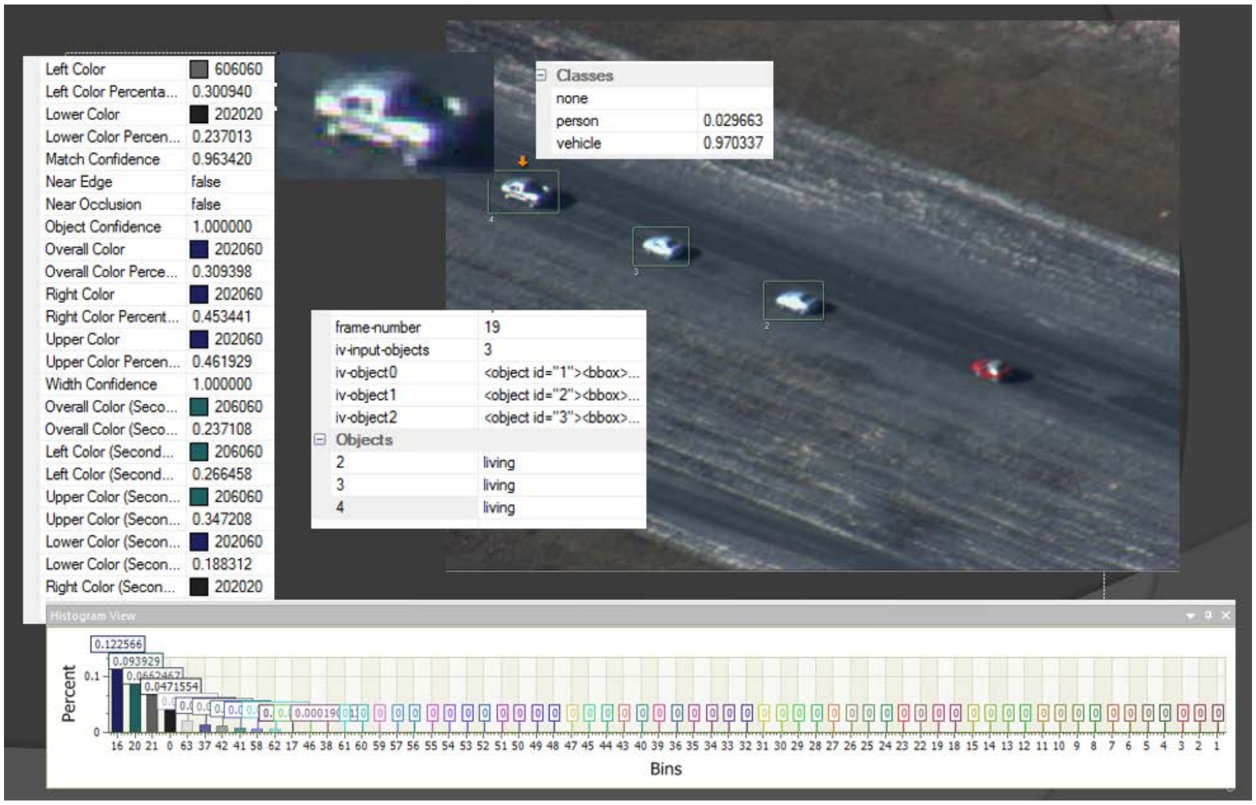
Figure 68. Woman Wearing Sweater and Shirt with Completely Different Colors Represented by "Upper Color" and "Upper Color (Secondary)" Tags



⊞ Classes	
⊟ Attributes	
Average Height	108.361794
Average Width	51.379379
Height Confidence	0.969006
Histogram Confidence	0.875701
Left Color	🟥 905050
Lower Color	⬛ 505050
Match Confidence	0.924820
Near Edge	false
Near Occlusion	false
Object Confidence	1.000000
Overall Color	⬛ 505050
Right Color	⬛ 303010
Upper Color	🟥 905050
Upper Color (Second...	🟫 505030
Width Confidence	0.965154
⊞ Sub-Regions	

Figure 69. Man with Backpack Represented by "Upper Color" and "Upper Color (Secondary)" Tags

When objects are divided into multiple zones, high-granularity color information is available. The primary and secondary colors in those zones is then calculated. **Figure 70** shows an example of how this is performed.



Left Color	⬛ 606060
Left Color Percenta...	0.300940
Lower Color	⬛ 202020
Lower Color Percen...	0.237013
Match Confidence	0.963420
Near Edge	false
Near Occlusion	false
Object Confidence	1.000000
Overall Color	🟦 202060
Overall Color Perce...	0.309398
Right Color	🟦 202060
Right Color Percent...	0.453441
Upper Color	🟦 202060
Upper Color Percen...	0.461929
Width Confidence	1.000000
Overall Color (Seco...	🟩 206060
Overall Color (Seco...	0.237108
Left Color (Second...	🟩 206060
Left Color (Second...	0.266458
Upper Color (Secon...	🟩 206060
Upper Color (Secon...	0.347208
Lower Color (Secon...	🟦 202060
Lower Color (Secon...	0.188312
Right Color (Secon...	⬛ 202020

Classes	
none	
person	0.029663
vehicle	0.970337

frame-number	19
iv-input-objects	3
iv-object0	<object id="1"><bbox>...
iv-object1	<object id="2"><bbox>...
iv-object2	<object id="3"><bbox>...
Objects	
2	living
3	living
4	living

Figure 70. Color Extraction with High Granularity

In addition to IntuVision's color extraction improvements, they also began work on updating vehicle classification in their SDK. The base vehicle and person classification models were provided to Primordial within a release of IntuVision's SDK. The initial models IntuVision provided for this project were trained with overhead views of vehicles and people (views from a second-story window, see **Figure 71**).

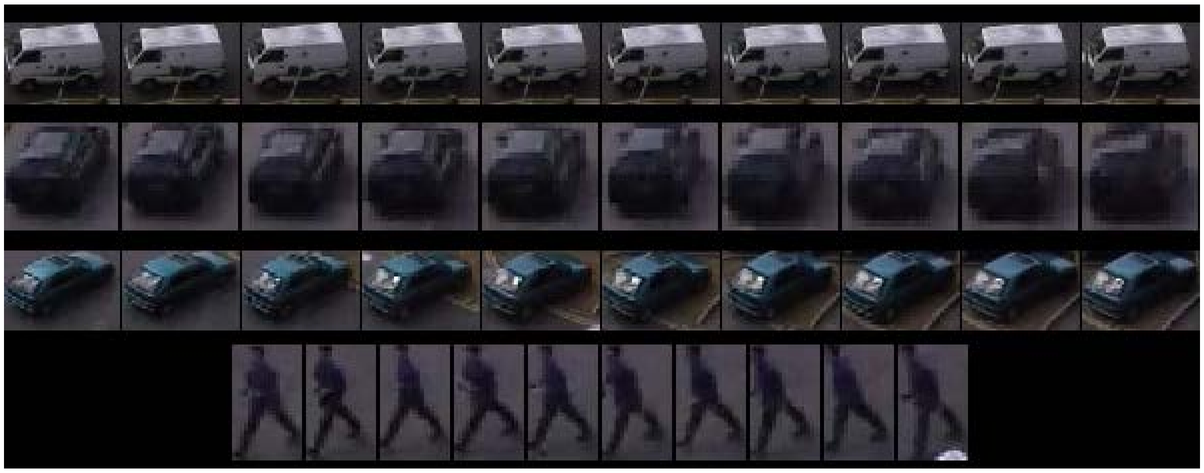


Figure 71. Overhead Vehicle and Person Training Data Examples

These models work most reliably when applied to camera views with similar view angle to the data used when training the model. In order to expand the classification models

IntuVision began collecting video from other views, and training new vehicle models based on lower angle cameras. At this angle, vehicle orientation plays a much larger role in classification (the front and rear of a vehicle ‘look’ very different). IntuVision began training new models based on low-angle front and rear vehicle examples (see **Figure 72**).

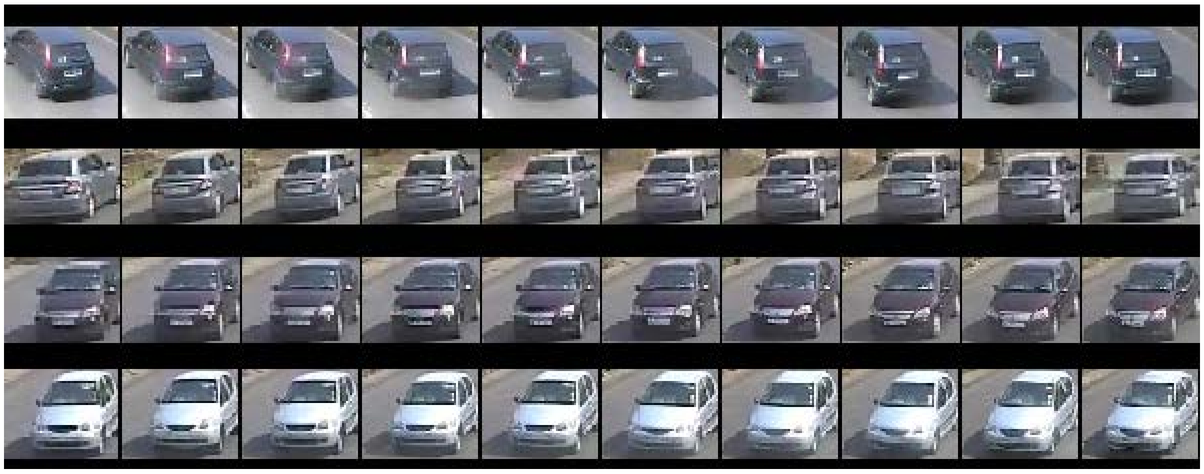


Figure 72. Front and Rear Vehicle Training Data Examples

In addition to alternate viewing angles, IntuVision also began training on different vehicle types (cars, buses, trucks and motorbikes). Where needed, this would allow the SDK to differentiate between these various classes of vehicles (see **Figure 73**).

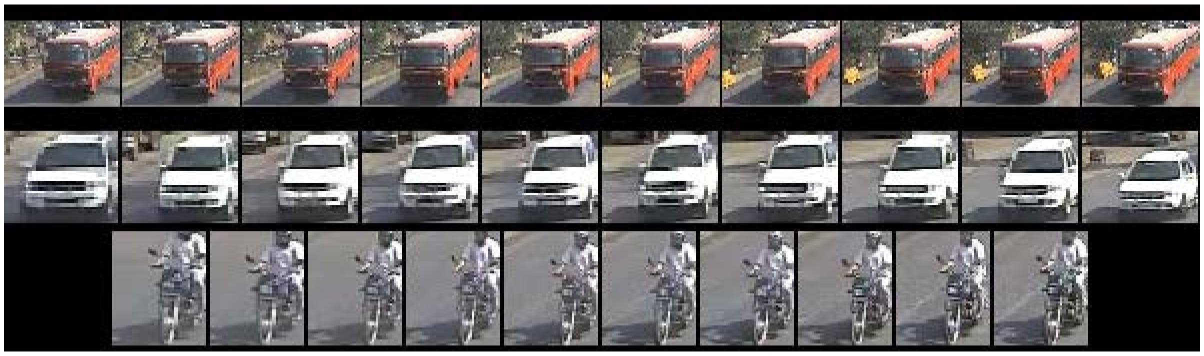


Figure 73. Bus, Truck and Motorbike Training Data Examples

While IntuVision was working on improving their SDK, Primordial began upgrading Tentacle's video processing code in the detection application to handle HD-quality video footage. This improvement was recommended by IntuVision to help improve the performance of the classification and metadata extraction for the overall system, as the higher the resolution of the input video, the better the metadata and classification extracted would be.

Aside from color and classification improvements, IntuVision also began working on gender and ethnicity detection techniques. IntuVision began by using a method known as Biologically Inspired Model (BIM) for classifying gender and ethnicity from facial images. These models had been developed over several years from an ever-expanding set of training data. They performed well in IntuVision's own test sets. Unfortunately, the baseline videos provided by Primordial did not contain large enough faces for IntuVision's face detection and classification algorithms, so the following examples in **Figure 74** came from test sets created by IntuVision internally.

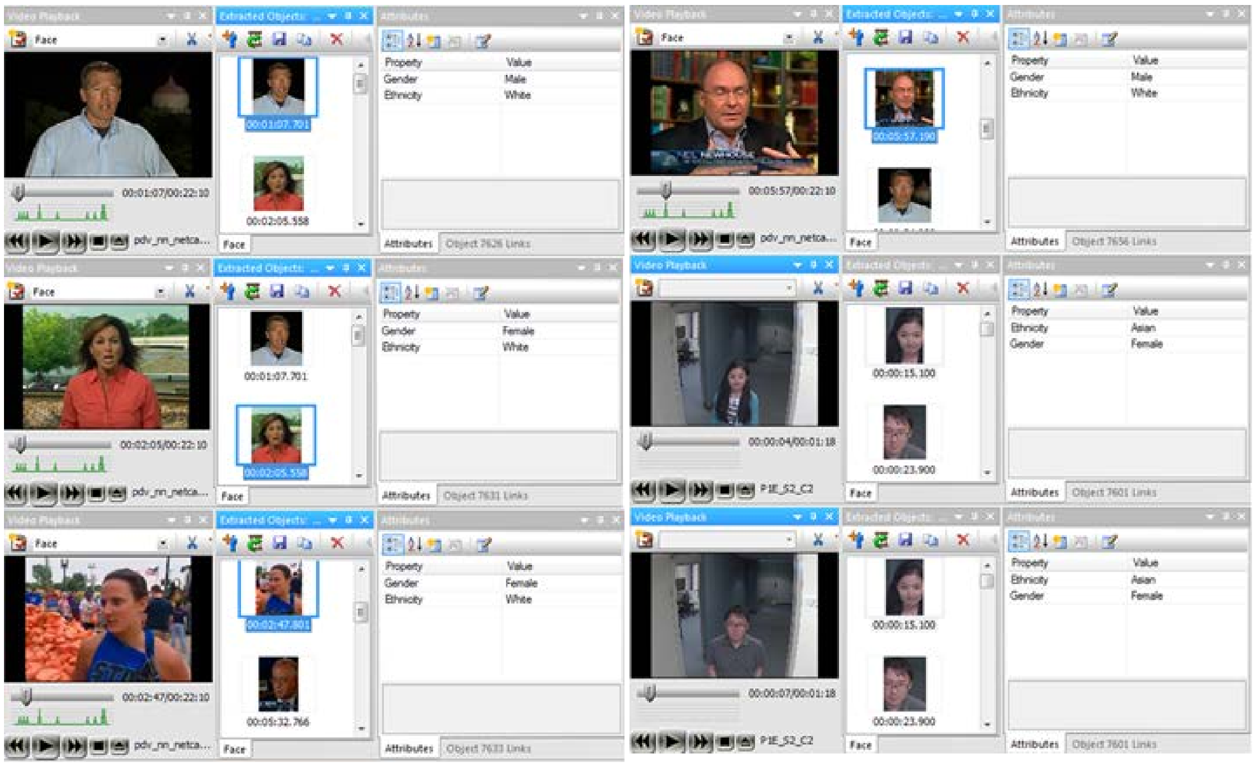


Figure 74. Gender and Ethnicity Classification Examples from Different Camera Views

In order to address the task of facial recognition for this project, IntuVision began integration of the commercial product Luxand Face SDK into the IntuVision SDK. Face SDK provides face detection and recognition in a commercial package for purchase. The face detection is performed in real time, and only on already detected objects which are large enough to contain a detectable face (to limit computational intensity). While faces as small as 20 pixels wide can be detected by Luxand, it is only when the face width approaches 60 pixels that the facial matching portion begins producing stable results. **Figure 75** shows examples of the Luxand Face SDK as integrated into IntuVision's test software. In a 720p HD camera, this is the required person size for face detection (top left). People must be much closer to a low-definition (bottom) camera to get an adequately sized face for matching.

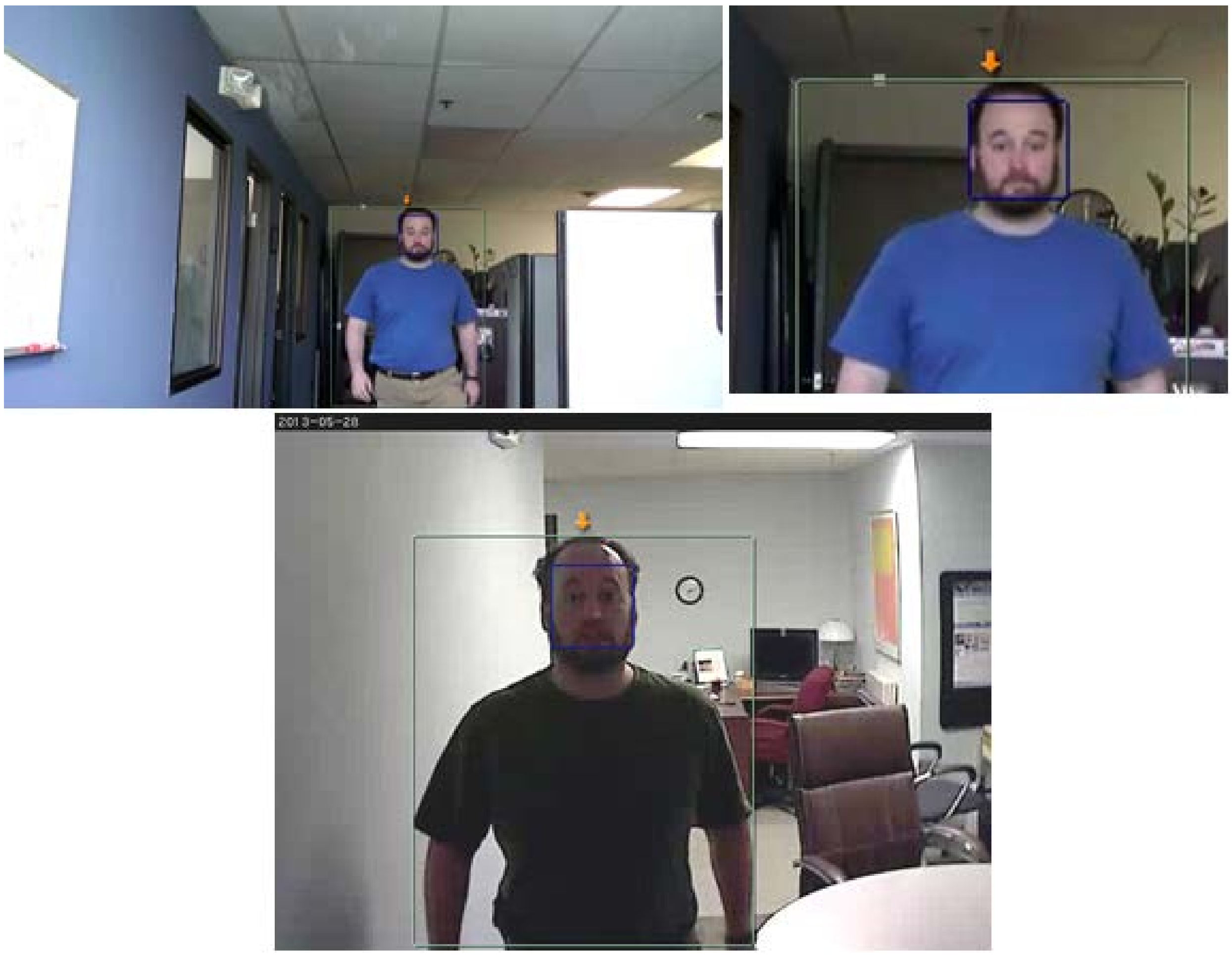


Figure 75. Luxand Face SDK

Accurate face recognition requires larger face images for applications such as ID confirmation with very low false acceptance and false rejection rates. Despite this, IntuVision's experiments showed that the algorithm can still help with matching people across camera views as an additional feature. If a test face produced a weak match to any of a small set of candidate faces, this, combined with other weak matching mechanisms (object type classification, object colors, location, etc.), was able to improve tracking objects between cameras.

The face recognition algorithm needs face models to be trained for each person to check against the detected new face. In IntuVision's experiments, between multiple test subjects the face match feature produced good results within a camera view (i.e. face models are trained and tested within the same camera view), when the tracked faces were at least 60 pixels wide and the people were facing directly towards the camera. Even though there are multiple algorithms proposed in the literature for face recognition that are invariant to illumination, face pose and face expression; the performances reported in those algorithms have not yet achieved the degree of robustness for widespread applications. Luxand's Face SDK does implement a degree of view and illumination invariance but their recommended viewing geometry allows for +/- 30 degrees of in-plane rotation and only +/- 5 degrees of out-of-plane rotation.

The example face match scores for two people in the same camera view as their models were trained are shown in **Figure 76**.

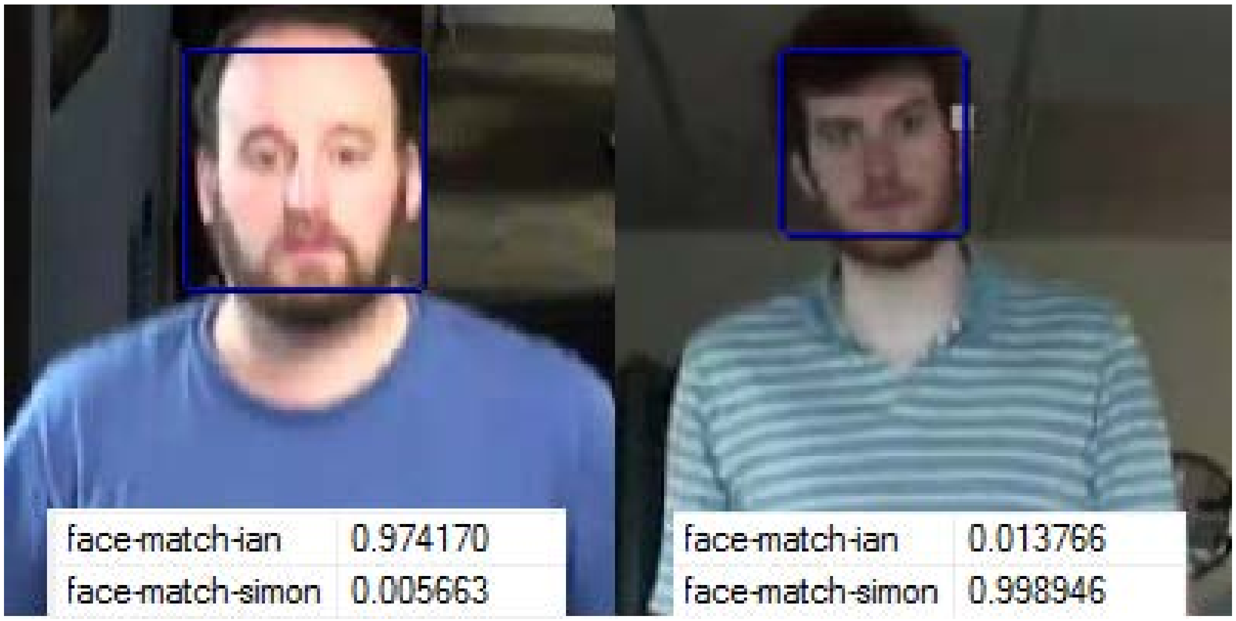


Figure 76. Models are Trained for the Test Subjects: Ian and Simon

The left-hand (Ian) model was trained in a ceiling mounted camera; the right-hand (Simon) model was trained on a table mounted camera. Both viewing angles fall within Luxand's recommended +/- 30 degrees of in-plane rotation. The match score is generated as the inverse of the False Acceptance Rate for a given match. A score of 0.99 equates to a 1% possibility that the face belongs to someone other than the model.

However, for face matching across cameras, a given test face is matched against models which are produced in another camera with different viewing geometry. The Luxand face matching was less accurate in this scenario. In IntuVision's second experiment, as illustrated in **Figure 77**, Ian's face is detected in three different camera views and matched against the same two models discussed above (those belonging to Ian and Simon).

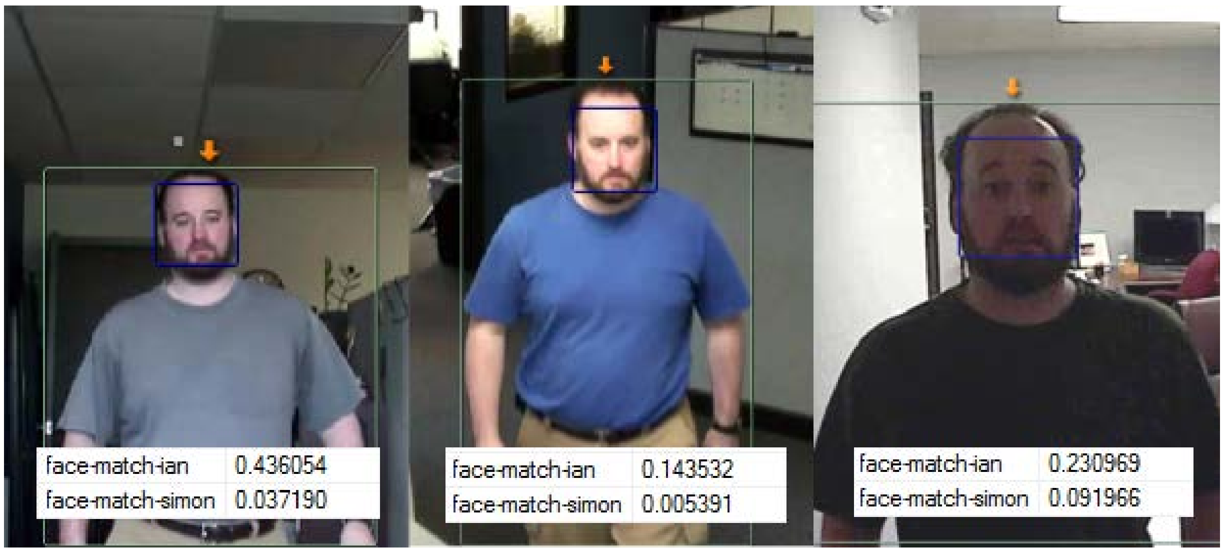

face-match-ian 0.436054
face-match-simon 0.037190
face-match-ian 0.143532
face-match-simon 0.005391
face-match-ian 0.230969
face-match-simon 0.091966

Figure 77. Confidence of Matching the Same Face Across Camera Views is Consistently Higher than Matching Against a Different Face Model

As seen from these results, the match scores for the same face across different cameras are much lower than the match range IntuVision observed within a single camera. The highest match score of Ian's face for himself (as denoted by "face-match-ian" in the figure) is obtained for the camera with most similar view geometry to the one in which the face model was trained. This left-hand image comes from a table-mounted camera with the same resolution, and similar out-of plane rotation. The second image comes from a camera with similar resolution, mounted higher than the original (greater in-plane rotation), and representing a greater out-of-plane rotation. The third image was taken with a lower resolution wall-mounted camera which again shows a straight-on (low out-of-plane rotation) view. Even though the match scores for the same face over different cameras are lower than over a single view, the match scores between different faces are significantly lower (a magnitude of order) as observed in the match of Ian's face to Simon (as denoted by "face-match-simon" in the figure). Hence, the face match score can serve as an additional feature in camera hand-off for people in ground surveillance scenarios.

At this stage, the IntuVision SDK was set up to generate and match facial recognition models within a single camera. These models were able to be manually retrieved and passed into other cameras for matching (as was done in the second experiment above). The integration plan at this point was for IntuVision to pass these models to the Primordial framework as they are generated. The framework would then pass the models to other near-by cameras for short-term use in generating match confidences. These match confidences would be combined with other matching features (as discussed above) to arrive at a final camera hand-off decision.

Aside from the above work, IntuVision continued to work on UAV object classification using separate classifiers for different view directions, as well as BIM models. Additionally, they investigated using local edge detectors to form features.

Figure 78 shows the results of some of the BIM model modifications as performed on the VIVID video dataset.

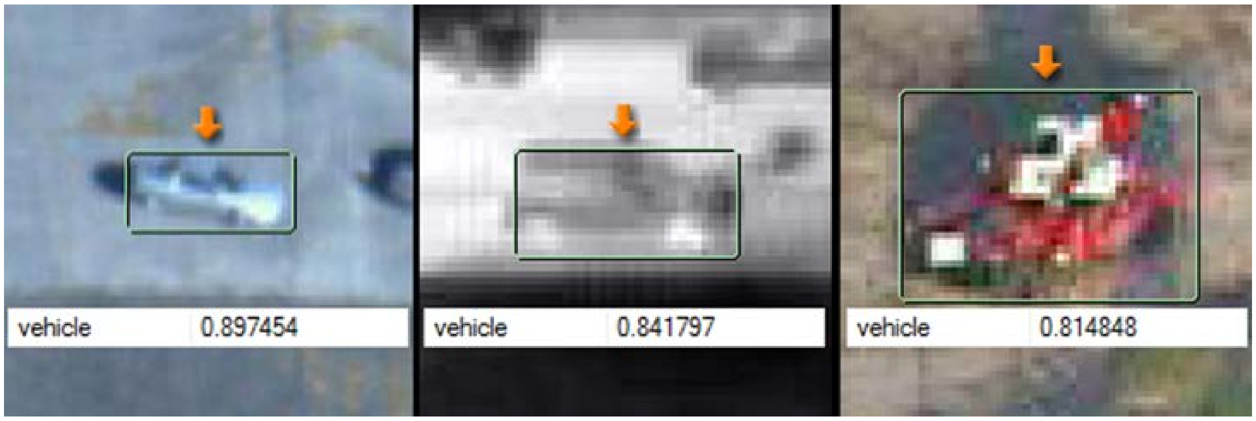


Figure 78. UAV BIM Models, Performing Vehicle Classification in DARPA VIVID Videos

IntuVision then began adding support for vehicle vs pedestrian classification in UAV video frames using their previously developed BIM models and local-edge based features. **Figure 79** shows the output of those integrations.

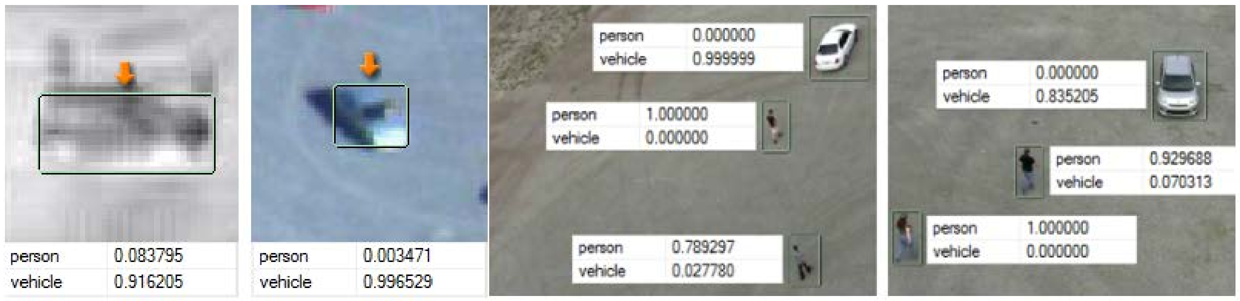
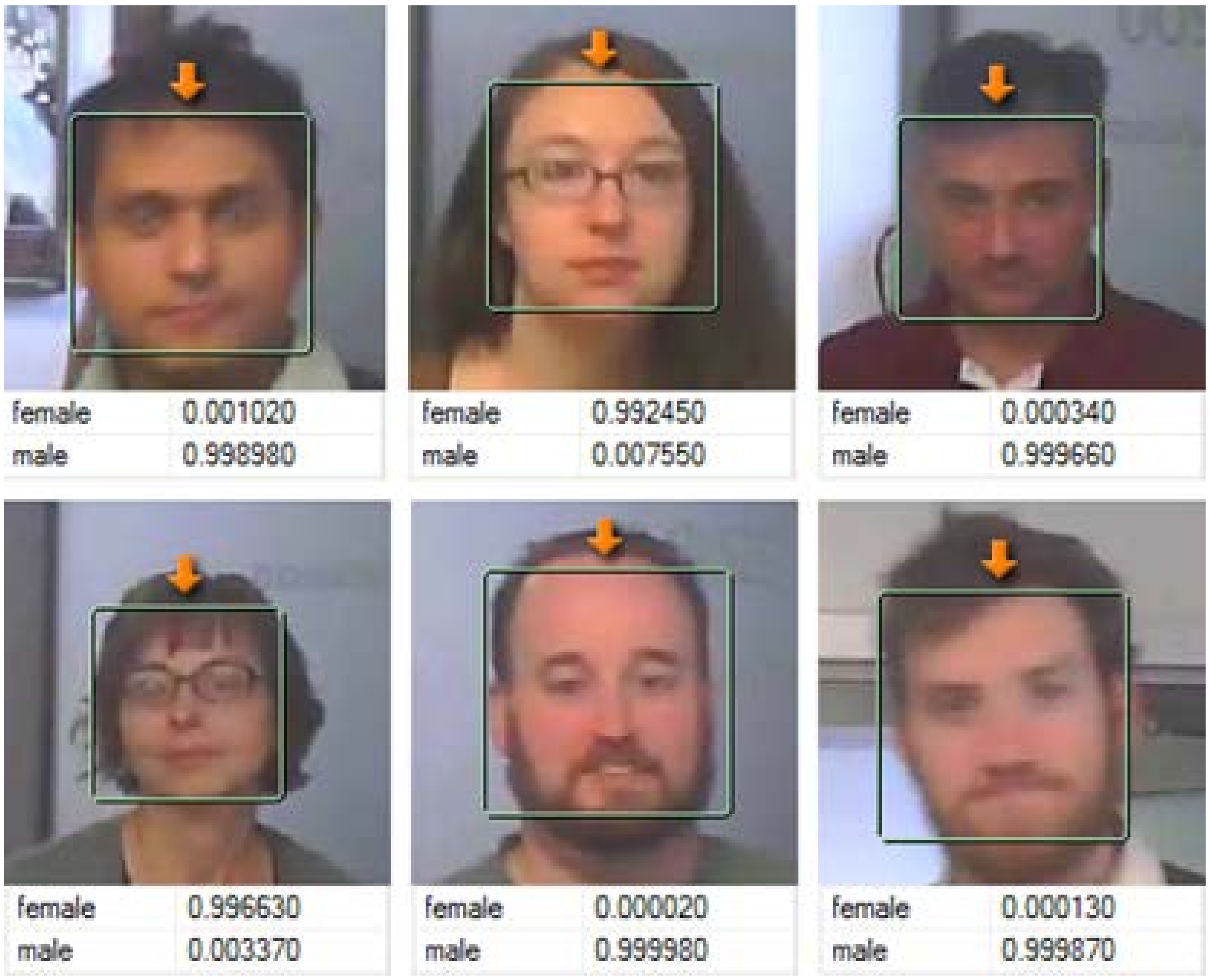


Figure 79. Vehicle and Pedestrian Classification Examples

More information about these approaches can be found in section of this document describing the approaches to moving target detection.

Luxand released an updated version of their Face SDK plugin during this phase. This update added gender recognition to their SDK's feature set. As a result, IntuVision began investigating its gender classification capability. The results were promising, so IntuVision updated their SDK to include this functionality as well. The SDK was modified to report gender classification results to the Tentacle framework along the same path as all other weak classifiers.

Figure 80 shows the capability of the gender classification system.



female	0.001020
male	0.998980

female	0.992450
male	0.007550

female	0.000340
male	0.999660

female	0.996630
male	0.003370

female	0.000020
male	0.999980

female	0.000130
male	0.999870

Figure 80. Gender Classification on the IntuVision Staff Using the Luxand Face SDK

Once Primordial received the updated IntuVision SDK, which included improvements for face detection and UAV video tracking, they began testing it for integration into the main Tentacle system. Primordial also registered for and installed the Luxand Face SDK which is required for the facial recognition and gender classification features. After testing several videos without success in detecting faces, Primordial worked with IntuVision to get the correct configuration settings. This allowed the Primordial team to fix an issue with the configuration file that allowed for successfully detecting faces in the sample video Primordial received from IntuVision. It seemed that there may be issues with lesser quality videos for this purpose, as Primordial was only able to successfully detect faces with the sample video provided by IntuVision. Due to the impending project deadline and perceived diminished interest from AFRL, Primordial opted against integrating the facial recognition and gender classification functionality into the main Tentacle system's detection pipeline.

IntuVision continued work on training and improving the models for UAV-based vehicle and people classification to support the UAV-tracking effort for this project. They generated receiver operating characteristic (ROC) curves to assess and compare the performance of each of the classification model types and determined recommended operating regions for true detection and false acceptance. Additionally, they provided an updated SDK, documentation and some background information on Biologically Inspired Models (BIM) to Primordial.

At the end of this effort, IntuVision provided a final version of the IntuVision SDK, a plugin enabling face and gender detection and matching and person/vehicle models detailed in the report. Additionally, they supplied supplementary documentation on the use of Tentacle-specific bounding box ingestion, object color reporting, object classification and face detection and matching.

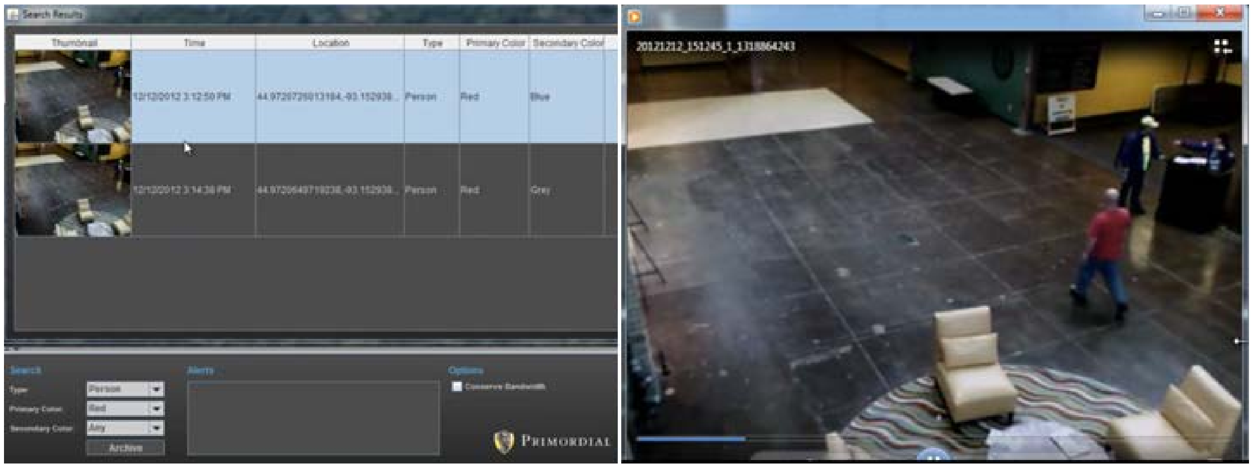
3.6 Implement TIPL Using STANAG 4559

STANAG (NATO Standardization Agreement) 4559 (8) is a standard interface for querying and accessing a library of ISR (Intelligence Surveillance and Reconnaissance) products, or IPL (Product Library Implementation). The principle role of an IPL is to provide a search function that provides users visibility to all ISR library product information. The NSILI (NATO Standard ISR Library Interface) provides a standard bridge between the user interface that supports this search and display of library product information, and the IPL itself. Multiple users can simultaneously reference or execute identical queries. After reviewing the search results, users may request selected library holdings (e.g. archive video of live streams) for delivery.

Primordial’s goal for this task in Phase II in creating the TIPL (Tentacle IPL) was to create an archive system for tagging and storing recording video related to video tracking events and allow the user to retrieve it based on automatically extracted metadata information.

This proposed functionality was demonstrated in a mocked up video during phase I. For this initial task, the Tentacle Server module was updated to store meta-data to a central Microsoft SQL database, as it arrives from the Tentacle mote. The metadata being stored for the first-pass implementation included: Entity ID, Entity Location, Shirt Color, Pants Color, and Entity Type (e.g. Vehicle versus Person).

As the input video is being streamed, the video is automatically tagged with metadata and broken into short clips by the Tentacle detection application, covering only the sections of video relevant to tracking, with the metadata stored to a relational database. The clips are placed on a remote networked storage location, whose address is included in the metadata frame associated with the target. A user can then use drop-down menus in the Tentacle client and request an archive search for all clips matching the selected criteria. This query is sent to the Tentacle Server over the network, and the user is then given access to the relevant matching video clips. For example, if a user selects red and person as metadata for the query, he/she will be provided with a list of all clips including red people, with the option to request and play a desired video clip (see **Figure 81**). In **Figure 81** the user has just performed a search for clips containing people wearing red shirts. The left pane above shows the list of clips matching with this search. The right pane shows a screenshot of the selected clip, which was requested over the network by double clicking the first row in the archive search results.



Thumbnail	Time	Location	Type	Primary Color	Secondary Color
	12/12/2012 3:12:50 PM	44.9720726013184,-93.152938...	Person	Red	Blue
	12/12/2012 3:14:38 PM	44.9720649719238,-93.152938...	Person	Red	Grey

Type: Person
Primary Color: Red
Secondary Color: Any
Conserve Bandwidth

Figure 81. Screenshot of the Archive Search Functionality in Tentacle Client

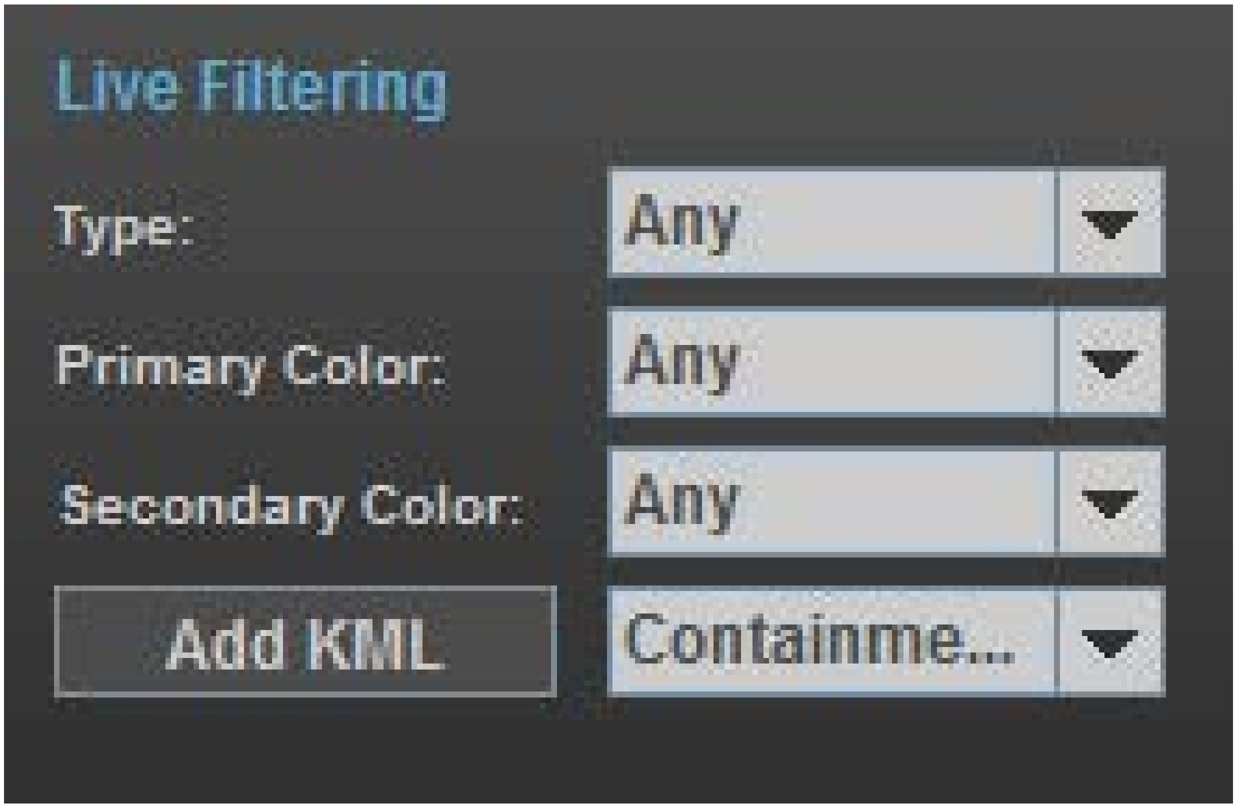
When the user clicks the video icon in the results, the application would launch an external video viewing application and play the relevant video clip from its network location.

Primordial created a demonstration video of this capability (8).

3.7 Implement Spatial Query Engine

The Tentacle team's goal for this task in Phase II was to implement and integrate archived and live spatial query functionality into the Tentacle system.

In Phase I, filtering and processing of data was on a rudimentary level. **Figure 82** shows the interface used for filtering incoming entities in Phase I.


Live Filtering

Type: Any

Primary Color: Any

Secondary Color: Any

Containme...

Figure 82. Incoming Entity Filtering in Phase I

In Phase II, potential spatial queries to be supported included: cluster, count, cross, direction, enter and proximity. Entities would be initially tracked from UAV or static camera videos, but potentially also other sources would be supported in the future e.g. GMTI (Ground Moving Target Indicator), seismic sensors, or marine AIS (Automatic Identification System), etc. **Figure 83** shows the overall processing flow of the Tentacle system.

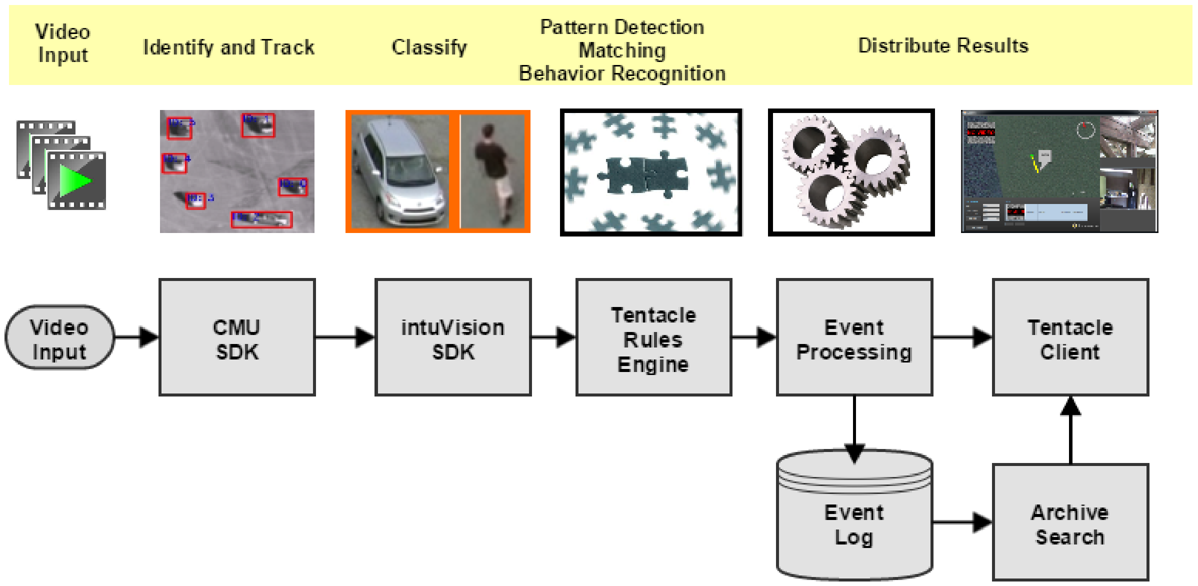


Figure 83. High-Level Architecture of the Entire Tentacle System

When this task was started, the Tentacle system already supported performing archived and live searches of video, based on extracted metadata (e.g. shirt color, entity classification, etc). However, the purpose of the spatial query feature was to support a broad range of searches of tracked entities based on geo-spatial and temporal relationships. For example, a user might query for entities which have entered a user-specified polygonal region (defined by latitude/longitude coordinates), or have come within a given proximity of a given region/location. The spatial query engine was intended to also support queries based on spatial "behaviors". An example of this might be when detected entities are clustered into groups or when an entity is loitering. These complex behaviors inherently involve geospatial criteria for detection.

To begin, Primordial investigated existing systems with related functionality, as well as available tools/libraries for spatial storage and analysis.

Primordial also switched the Tentacle system's archive search functionality to use a Microsoft SQL server database, due to its built-in geospatial feature support. This allowed the Tentacle system to store points, tracks, polygons or other geospatial shapes and would provide a more flexible architecture to support later changes. This also provided functionality for searching for positions within defined areas, as well as indexing the entered entity locations for faster querying.

Primordial then updated the Tentacle client to support loading, displaying and selecting KML-based polygonal regions for use in spatial queries. This required implementation of a KML-parser, developing a hashtable-based storage structure and creation of supporting code for retaining and accessing the user-provided shapes, as well as GUI objects for user manipulation of the input for the various queries and for display of the shapes on the client's WorldWind map.

Additionally, Primordial added and modified networking code to transmit the spatial queries and associated KML shapes to the Tentacle server as well as receiving and processing the archived responses.

To start, Primordial implemented a first-pass version of the first spatial query: proximity. This would return archived results based on proximity to a user-specified geographic region. The user provides a KML polygon, polyline or point which is sent to the server along with a threshold distance. The underlying code used SqlServer’s native classes to calculate spatial distance between geographic locations. The server would scan the database for records where the recorded entity location was within a given distance of any point on the KML shape. When a record qualified as “in proximity”, the code would determine all associated points for the entity’s chronological track using a database table association. These detected tracks were then returned to the user.

At this time, Primordial began examining SRI TerraSight as a possible Phase III/Commercial integration target for the Tentacle system. SRI International produces a product called TerraSight that performs real-time video processing, geolocates entities in video and provides many other similar features to those included in the Tentacle system. Because they have such a similar product which is commercially successful, Primordial felt that they would be an excellent candidate for integration to commercialize the Tentacle system in the long-term. Additionally, SRI had previously expressed interest in spatial query functionality based on Primordial’s Ground Guidance mapping system. Primordial representatives met with SRI TerraSight to discuss future opportunities and possibly commercializing Tentacle as a product but no definite plans were made at this time.

Primordial created a demonstration video sent to AFRL of the proximity functionality at this stage (9).

A screenshot of the video is also shown in **Figure 84**. The pink polylines represent main supply routes (MSRs) and the red line represents a spatially proximal track which had been previously recorded and stored in the Tentacle system’s database.

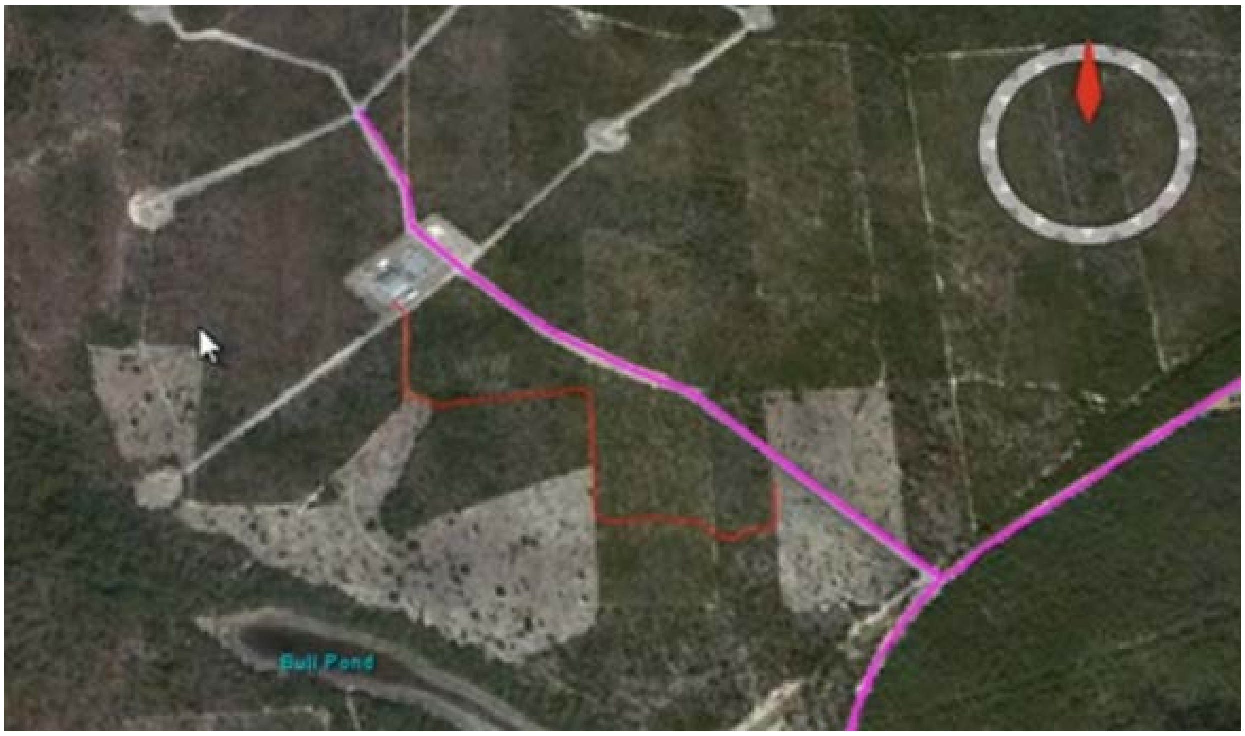


Figure 84. First Pass Implementation of a Proximity-Based Spatial Query

After Primordial successfully demonstrated the proximity query implementation, they began work on implementing a containment query option. The containment search returned all historical tracks which were contained either fully or partially within the user-supplied polygon shape. In addition, Primordial implemented a count option which is a related search that returns the numerical count of entity locations inside the region, rather than returning the tracks themselves.

Primordial created a video demonstration of the containment/count query implementation at this point (9).

Once a first-pass version of the containment query was complete, Primordial began adding support for polygon enter/exit queries, which are essentially the same query in opposite orders of entity location evaluation. The system took in a user-specified polygon shape from the client which was then transmitted to the server with the associated search request information. The system then searched for points which were contained in the polygon shape using the same algorithm implemented the containment query. For each point found, the system would retrieve associated points in the same track via a database table association and would then process the track points surrounding the contained points. If the previous point was outside the polygon, the track would be identified as having entered the shape. If the next point after the contained point was outside the shape, the track would be labeled as exiting the shape. Obviously, it was possible for a track to be associated with both types of events if the track entered and exited the shape.

Figure 85 provides an illustration. When performing the search the user could specify whether they were interested specifically in tracks that entered or exited the region, or they could specify 'any' to receive all enter/exit tracks for the region. The large red box represents a polygonal area of interest; the thin red polyline represents a historical track which has entered/exited the shape. It was also possible to limit the search specifically to either the exit or enter case.

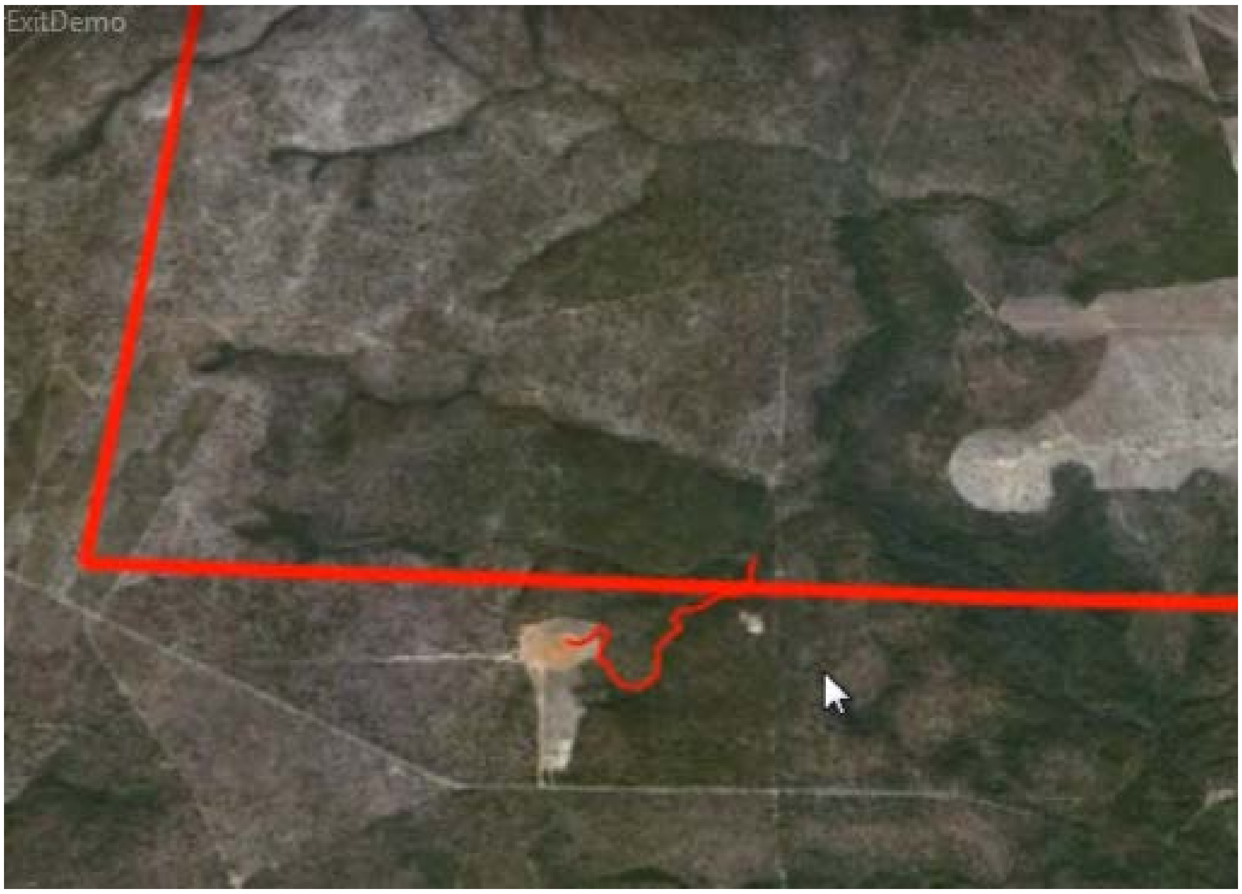


Figure 85. Enter/Exit Query

Primordial created a video demonstration of the enter/exit queries at this point (10).

A similar query type, polyline crossing, was added at this point as well. This used a similar detection mechanism. Points in proximity to the polyline shape were detected. Tracks were retrieved for the associated points. The tracks for the retrieved points were processed point by point in the order in which they were recorded. An algorithm would detect whether a point and the point adjacent were on opposite sides of the polyline. If so, the track would be flagged as having crossed the polyline shape. To specify the polyline shape, the user would use the same client GUI components as for the other query types.

In addition to proximity, enter/exit, polyline crossing and containment/count queries, Primordial also began implementation of "loiter" detection algorithms (see **Figure 86**). The high-level approach for loiter detection was to search the historical tracks stored in the Tentacle database for subsections of points which remained within a distance threshold for a given amount of time. The distance and time thresholds being used in the calculation were configurable in the user interface. Due to different conceptual definitions of what constitutes "loitering", Primordial also experimented with an alternate loitering detection approach that does not constrain the loitering to involve sequential points, but any points within the track. This would allow it to detect (for example) if the same vehicle travels around the block multiple times, or if an entity returns repeatedly to a location, as if scouting or monitoring it. In **Figure 86** an area of interest is defined by a KML shape (the green box) and the search returned a historical track (red polyline) stored in the Tentacle database.

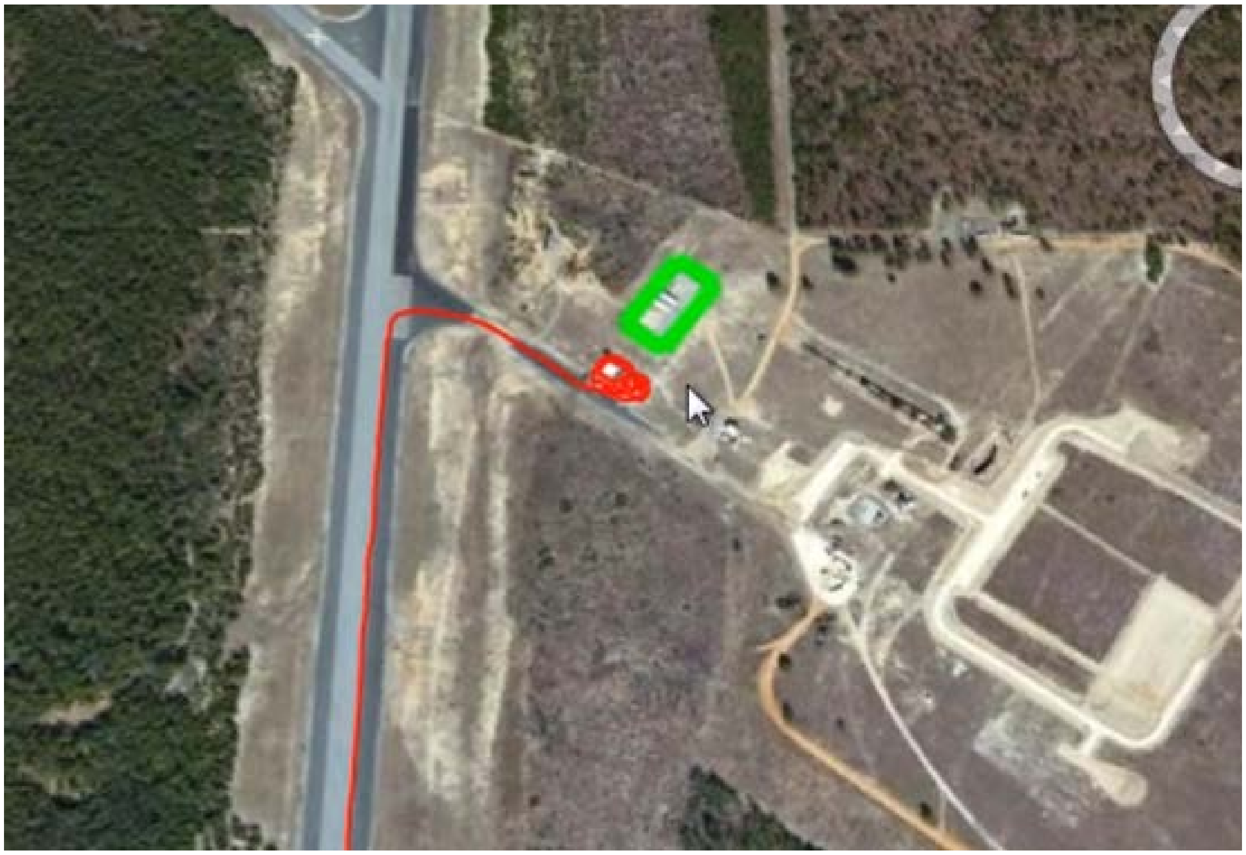


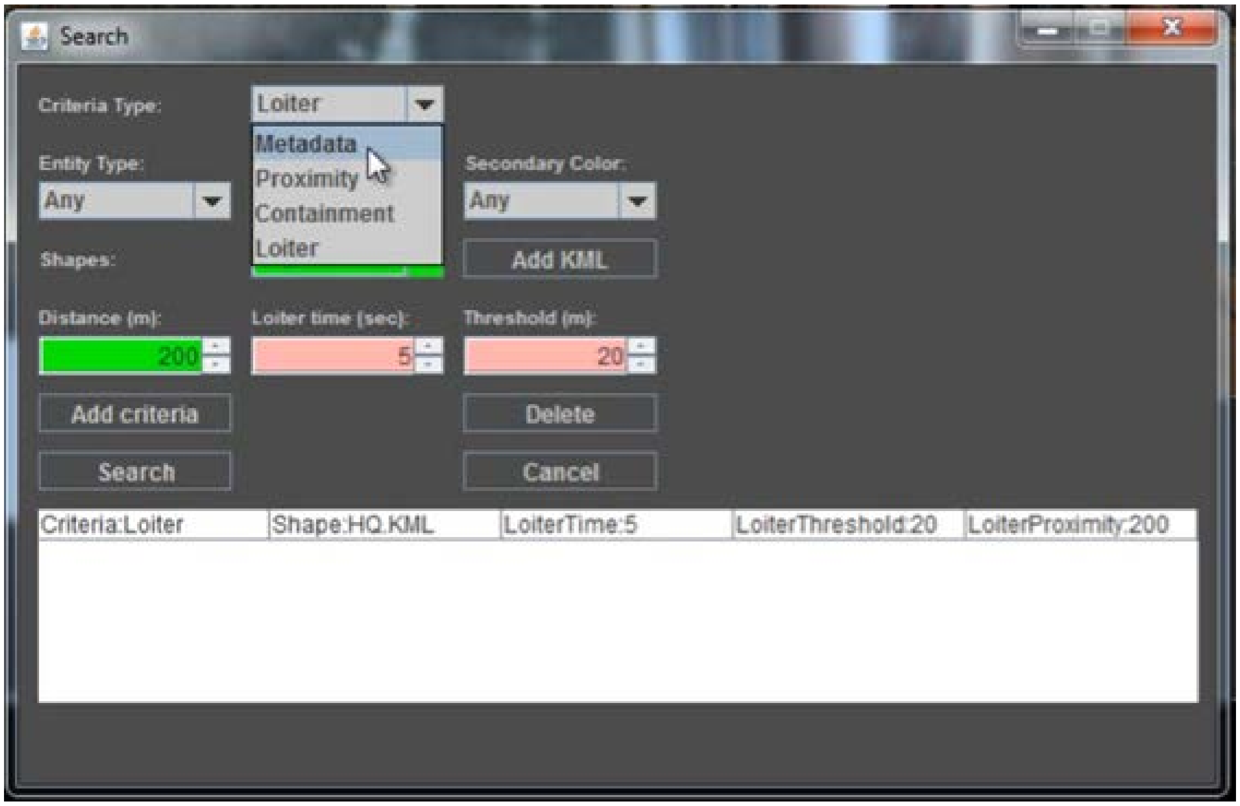
Figure 86. Loitering Detection

A video demonstration of the loitering functionality at this stage (11).

In addition to these preliminary spatial query types, Primordial began work on supporting complex Boolean query combinations. This allowed the system to support combinations of multiple query types in a natural way for the user without unnecessary effort. Using this, the user can perform combined searches such as “find all blue cars which entered a given region of interest”, or “find all people loitering in proximity to an MSR (Main Supply Route)” without having to do each subquery as a separate search (see **Figure 87**). This addition required updating the underlying core query logic, which was already reasonably complex, as well as updating the UI (User Interface) to support adding multiple queries to a single archive search request.

This task provided significant difficulty for implementation. One challenge was that if a user searched for “loitering in proximity to MSRs” for example, he/she likely is not interested in tracks which loiter far from an MSR and then at a later time travel past the MSR. Therefore, when combining these searches with an “AND” operation, Primordial updated the system to only return tracks which meet both criteria during the same time period (e.g. loitering next to the MSR). This proved to be challenging to implement.

The first pass of this functionality supported user-defined lists of the existing primitive queries (loitering, proximity, entity metadata), combined with an “AND” boolean operator. When multiple queries such as these were performed in sequence, it was not necessary to query the database each time (i.e. the results from the previous query in the boolean combination can be analyzed in memory), so for each primitive query the system supported two query approaches, one which applies spatial queries directly to the database and another which searches a subset of tracks in memory. The first query in the sequence would be done on the database, which is the more intensive option. Subsequent queries would only work on the “found” points from the earlier query, thus the system would iteratively “pare down” the points to be evaluated.
After the implementation of an “AND” combination of queries, Primordial started work on supporting nested combinations of “AND” and “OR” query requests e.g. “find vehicles that are (loitering AND (proximate to ambush locations OR proximate to MSRs)).



Criteria:Loiter	Shape:HQ.KML	LoiterTime:5	LoiterThreshold:20	LoiterProximity:200

Figure 87. Archive Search Interface

Figure 89 shows the updated archive search dialog, which allowed multiple primitive queries to be combined into more complicated search queries. In particular, the user could specify a particular query (e.g. proximity) and the relevant parameter values, and then add the query to the list of queries in the table at the bottom (only a single loiter query has been added in this example); once all of the desired queries have been added, the user could then press “search”. This dialog reflects a partially completed first pass, with improvements to the interface in progress.

Primordial completed a video demonstration of the Boolean combined queries at this stage (12).

In addition to the polygon-based queries described previously, Primordial also added support for performing spatial clustering of the archived entity locations in the database, with the resulting associated tracks representing hotspots of activity (See **Figure 88**). In general, spatial clustering refers to finding natural groupings of point locations e.g. finding points which are close to each other (either spatially, temporally or both) and yet comparatively distant from other points not involved in the same cluster.

The clustering algorithm which Primordial implemented is known as DBScan; this algorithm takes two input parameters: the minimum cluster size, and a distance parameter which determines how close points must be in order to be considered part of the same cluster. Unlike many other typical cluster algorithms, such as k-means or expectation maximization, DBScan has the advantage that it does not require the expected number of clusters to be specified in advance, something which, for Tentacle, would be essentially impossible to specify.

The output format of the cluster search was somewhat different from the other spatial queries developed previously. In particular, rather than returning a list of tracks, it returned a list of clusters. The user could then view all the tracks in a cluster by selecting a given result row, or view details for the cluster, which allowed the individual tracks in the cluster to be viewed in a separate table (as well as on the client's map view). This involved significant modifications to the code within the server which returned database results, as well as the client GUI in order to support such a different result from previous query types. **Figure 88** shows an example of the results from a cluster query. The figure shows the results of an archived search for cluster locations; in this case three tracks were involved in the displayed cluster. The clustering algorithm used is DBScan. This takes as inputs the minimum cluster size (number of points) and a distance threshold for determining when points are close enough to be part of the same cluster. In this search, the distance threshold was 10m and the minimum cluster size was 5 points.

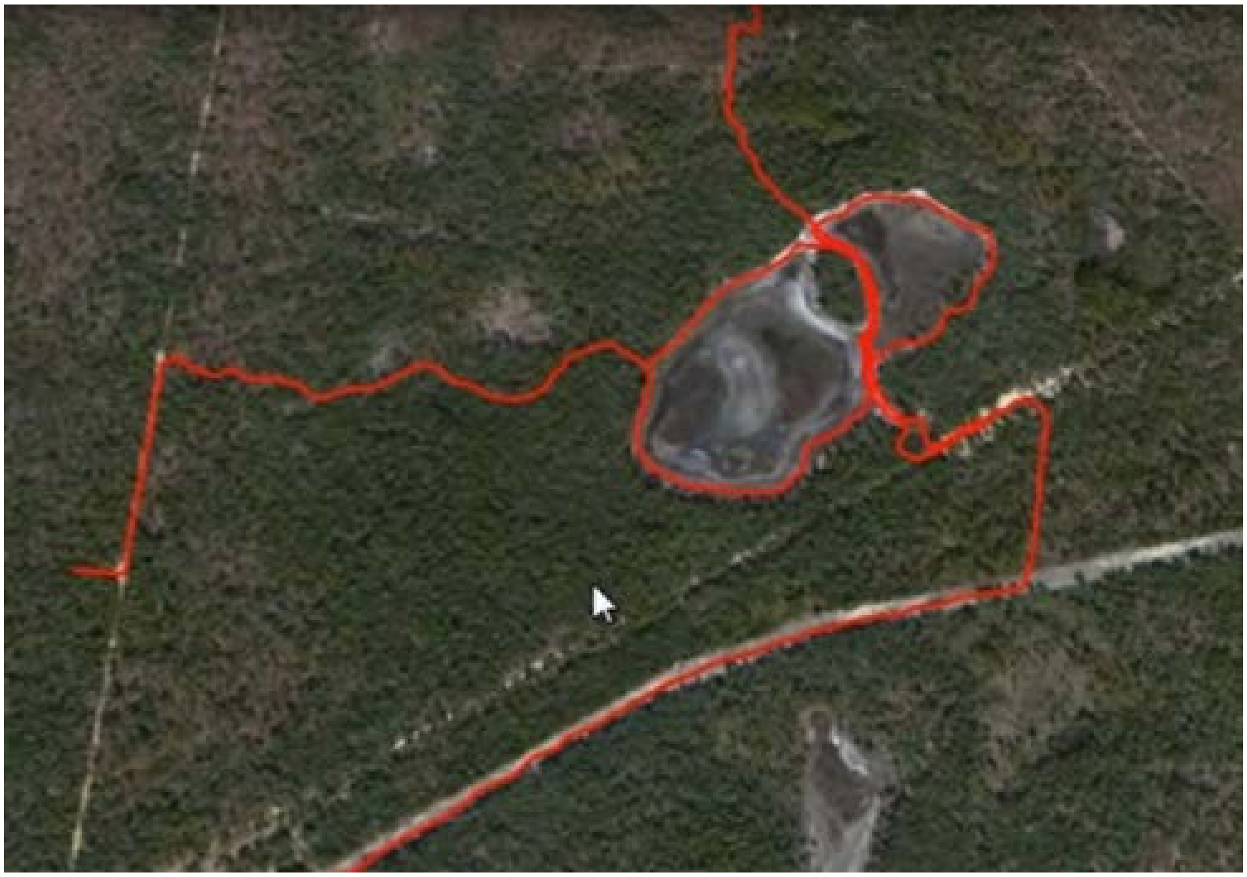
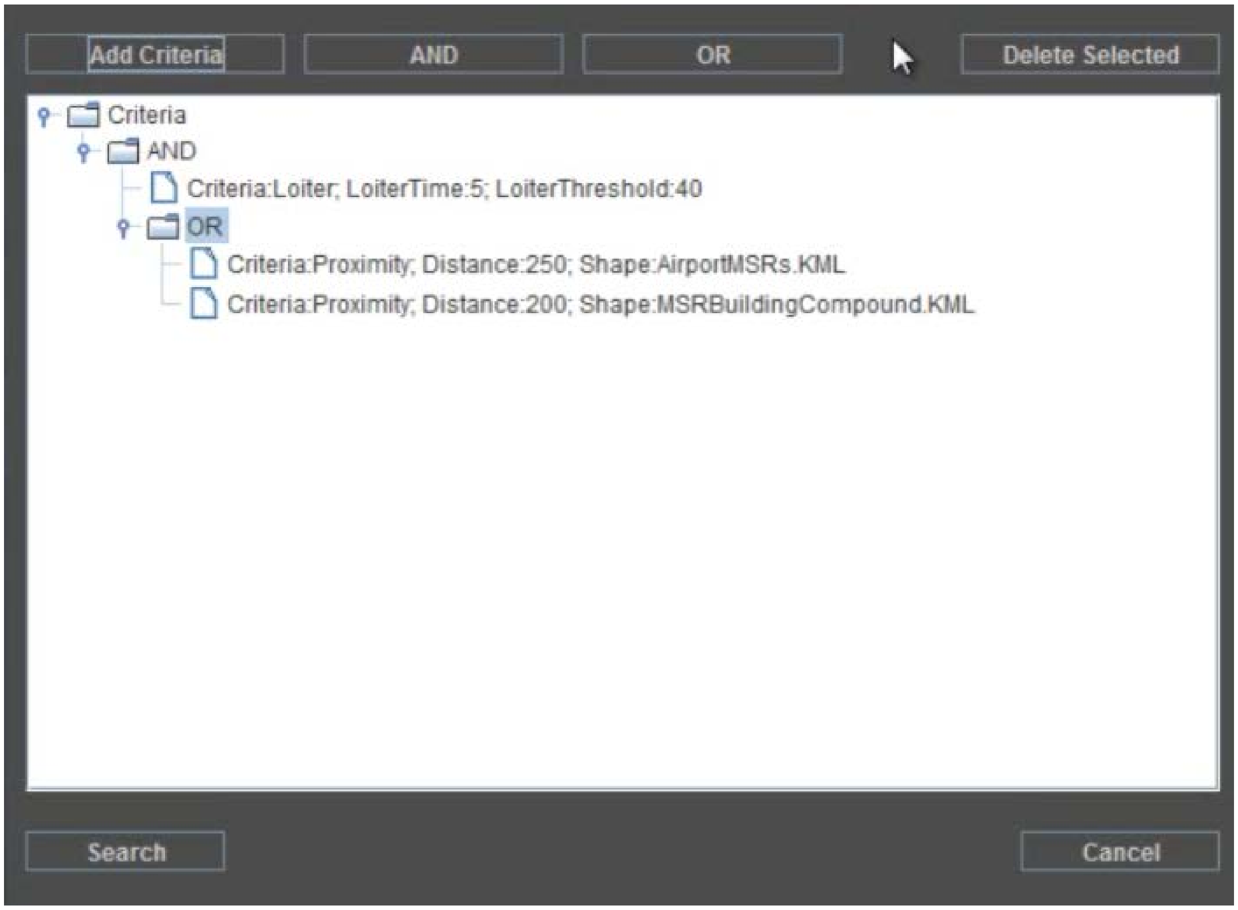


Figure 88. Cluster Query

Primordial created a video demonstration of the cluster query implementation at this point (13).

Previously, Primordial had added support for combining the available primitive database queries using Boolean (AND/OR) operators. However, Primordial wanted to support more complex query types and so the query functionality was updated to support hierarchical nested query combinations. For example, the system now supported performing a search for entities that were ('loitering' AND (in 'proximity' to an MSR OR in 'proximity' to a base)). In addition to implementing the underlying nested query logic, the user-interface was modified to allow the user to specify a tree structure representing the hierarchical queries (see **Figure 89**).

In order to support these complex hierarchical queries, Primordial implemented an updated approach for transmitting queries to the server over the network using XML. This was a major improvement from the delimited string format used previously. These modifications improved the robustness of the networked queries and allowed much more deeply nested query types to be created.



```
Criteria
  AND
    Criteria:Loiter; LoiterTime:5; LoiterThreshold:40
    OR
      Criteria:Proximity; Distance:250; Shape:AirportMSRs.KML
      Criteria:Proximity; Distance:200; Shape:MSRBuildingCompound.KML
```

Figure 89. Hierarchical Archived Search

After the fundamental query system structure was completed and a basic set of archived query implementations was complete, Primordial switched to focus on implementing "live" versions of queries. These versions would allow users to be automatically notified when complex events (such as loitering, proximity detection, etc) occured in the live incoming sensor feed. This feature was closely related to the existing archived spatial query functionality, except that it was intended to provide alerts regarding the real-time sensor stream as opposed to the archive of past recorded entity data.

As a foundation for implementing this functionality, Primordial investigated available 3rd party libraries for complex event detection. The team selected the open source Nesper library as the most promising option (see **Figure 90**). The Nesper software provides built-in capabilities for queuing and querying streams of temporal data in real-time, extremely efficient processing and was, on the surface, sufficiently extensible for the project's needs.

After downloading Nesper, Primordial set up a stand-alone test application to verify Nesper's ability to integrate well with the core Tentacle system as well as to provide the desired functionality. Primordial created a first-pass implementation of the base query types using Nesper within the test application, so as to keep the current Tentacle system functionality unchanged. However, this initial implementation allowed Primordial to effectively assess the value of the Nesper library and get a rough baseline for the real-time performance of the queries. The query types included in the first-pass implementation were:

1. Proximity, to fixed objects as well as other moving entities
2. Clustering
3. Containment, within polygon(s)
4. Polyline crossing
5. Enter/Exit, from polygon(s)
6. Loitering

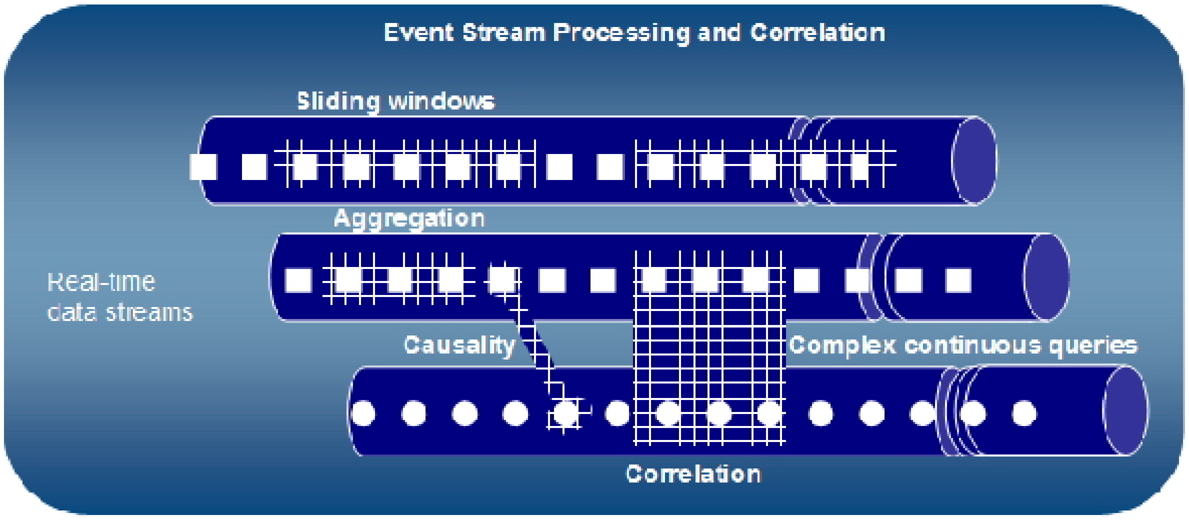


Figure 90. Nesper Event Processing

Primordial investigated available 3rd libraries for complex event detection, and implemented a prototype of the Tentacle spatial alert functionality building on top of the open-source Nesper library. This figure, taken from Nesper documentation (14), depicts some of the built in Nesper stream processing capabilities including automatically maintaining query-able sliding windows of event data.

Up to this point, testing of the spatial query functionality had primarily involved simulated entity track data which was manually created by the Primordial team using software developed in-house. However, for the current round of testing, Primordial obtained a more realistic real-world dataset from Flight24 (15), an aircraft tracking website. Specifically, a world-wide stream of airplane locations was used, which was available online and consisted of approximately 8000 planes being tracked at any given time (see **Figure 91**). Based on this live dataset, Primordial performed initial timing estimates of a range of live alerts, including proximity of airplanes to known nuclear power plant locations, containment within military base perimeters, etc (see **Table 2**).

As noted above, the live alert functionality was still in the process of being implemented/tested and optimized, so the time estimates were only a very rough baseline. Additionally, the timing estimates were focused on measuring the underlying computations and did not include the network latency associated with downloading the locations from the airplane location web site used for the testing, which could produce significant delay.

Caveats aside, the initial timing results suggested that some of the alert types would likely be quite feasible to provide in real-time (e.g. proximity to point locations), whereas others (such as proximity to polygons) would need to optimized significantly to perform in real-time on a dataset of this size.

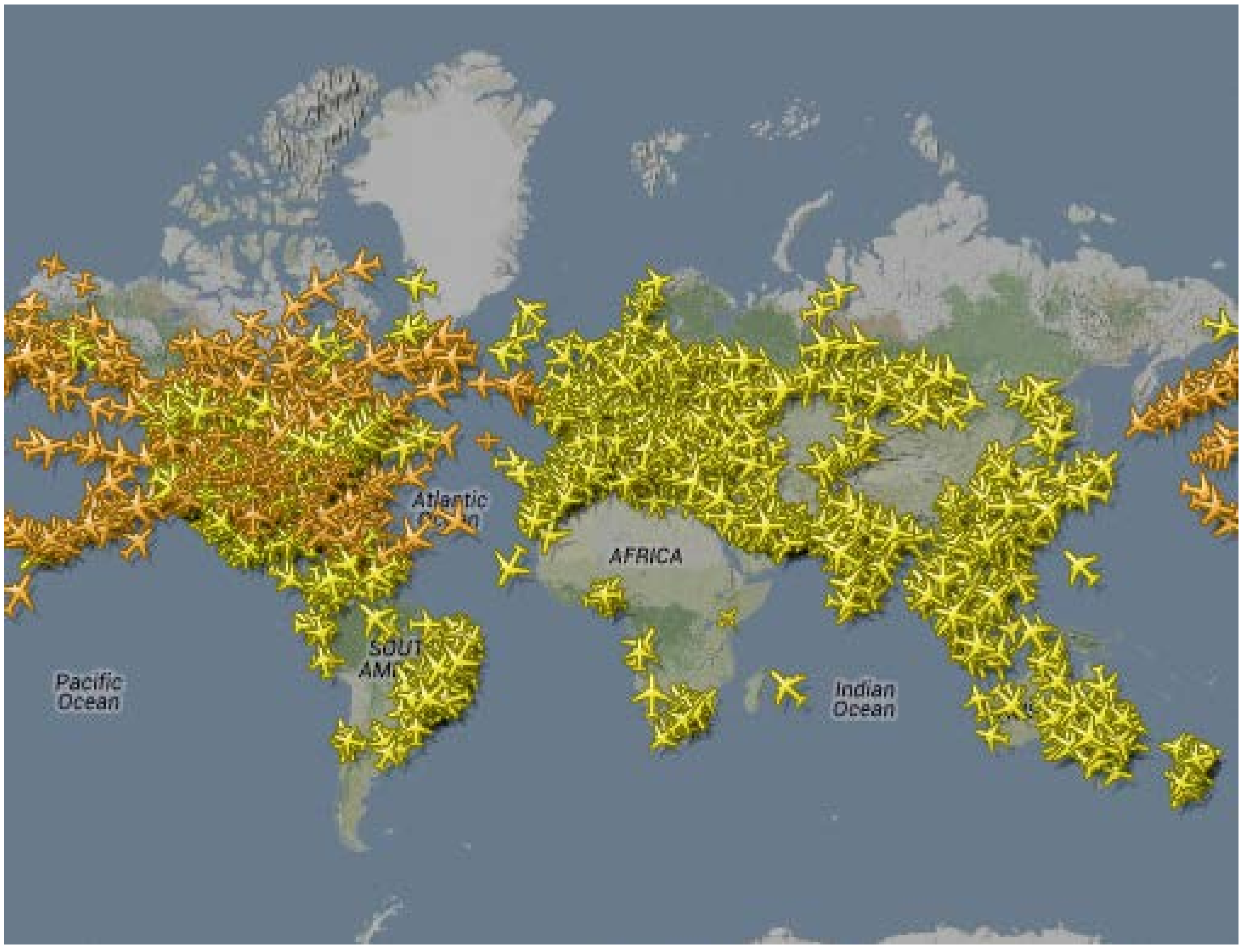


Figure 91. World-wide Airplane Locations

Table 2. Timing of Live Alerts

Live Alert Types	**Processing times**
Proximity to nuclear power plants	< 1 sec
Proximity to other flights	21 sec
Clustering	59 sec
Containment within military base perimeters	< 1 sec
Polyline cross detection	40 sec
Entering 3D airspaces	< 1sec
Exiting 3D airspaces	< 1 sec
Loitering	<1-11 sec
Proximity to military perimeters	10 min 4 sec

Total: (excluding polygon proximity)	2min 33sec (approx)

A first-pass at the live alert functionality for the existing spatial queries was implemented and this table shows the initial baseline time measurements for analyzing a single time-slice of world-wide airplane locations, typically containing approximately 8000 flights. The testing dataset also included 512 nuclear power plant locations, 1200 military base perimeters and 1206 airspace polygons. Clearly, some of these alert types would require substantial optimization or reduced search domains in order to function in real-time.

Primordial's next task was to improve the real-time performance of the live alert queries. In particular, quad trees (3) were incorporated into the system, as well as a more efficient function for computing distance (see **Table 3**).

The new quad-tree data structure allowed the system to more efficiently query for nearby locations, by limiting spatial searches to a traversal down the height of the tree structure as opposed to a full linear scan of the entities. For instance, the airplanes and airport locations were now stored in separate quad-tree structures, to facilitate finding airports (or other airplanes) in proximity to a given airplane location. This adjustment provided a roughly 7.3x speedup in a proximity-based search with the airplane data stream.

In addition to including quad-trees, Primordial also discovered that the SQLGeography distance routines used in the initial Tentacle query implementations were quite slow. These were native functions available from a Microsoft software library for interacting with Microsoft's SQL database system. Primordial experimented with replacing those distance calculation method calls with a hand-coded Haversine distance computation (18). This provided a roughly 6.7x speedup on the proximity test above. It's not entirely clear why the Microsoft SQLGeography functionality was so inefficient, and it's possible that it could be sped up with more appropriate settings. Another possibility is that it may have been performing a more accurate distance computation under-the-hood, compared to the great circle approximation provided by the Haversine formula, which would take an increased amount of time. However, for Tentacle's purposes, the Haversine formula produced acceptable results in terms of accuracy. See **Table 3** for an illustration of the improvements made by these changes.

Table 3. Real-Time Optimizations

Real-time optimization approaches	Original time	Improved time	Speed-up
Quad-tree for location queries vs. linear search	80 sec	11 sec	7.27 x
Haversine distance formula vs. previous approach (SQLGeography)	600 sec	89 sec	6.74 x

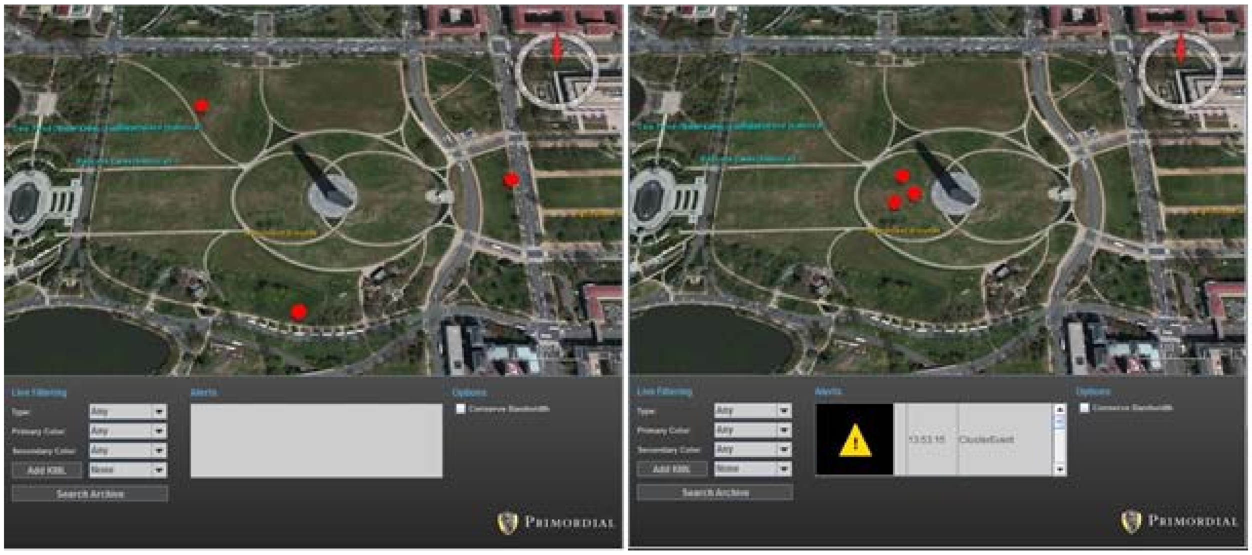
Table 3 depicts the results of initial efforts to optimize the live alert algorithms. In particular, quad-trees were incorporated into the system, and the SQLGeography built-in distance functions were replaced by a hand-coded Haversine distance implementation. These speedups were tested with proximity-alert for airplanes relative to airport locations, with approximately 8k airplane locations and 13k airports.

During these evaluations, Primordial also noticed that the clustering implementation was re-creating a quad-tree for each sliding window event. This caused significant slowdowns during processing due to overhead. The team modified the code to reuse the same quad-tree structure while still removing old entries, which resulted in a 375x speedup in processing.

Once the live queries were at an acceptable state, the next step for Primordial was to integrate the live alert functionality directly into the main Tentacle system. This involved porting the finished version of the live query algorithms from the test application to the main Tentacle system, then modifying the code to be sure the queries would be processed correctly. Additionally, this required modification of the client GUI code (to add elements for specifying such queries) as well as networking and processing code within the server itself. Also at this stage, the quad-tree implementation was optimized to provide better housekeeping by remove out-of-date entities which would to prevent the quad-tree from growing indefinitely over time. During testing, this issue wasn't a concern, but as part of the core system the problem had to be rectified to ensure system stability.

Another modification made at this point was that the clustering real-time alert was adjusted to match the original feature description with entities clustered at a single time slice (e.g. to detect groups of entities), as opposed to clustering points across both time and entity position. This was felt to be a more natural definition of "clustering" for a user.

The following figures show two of the alerts at this stage. **Figure 92** shows a cluster-alert example and **Figure 93**, a proximity alert, tested with simulated entity tracks. In **Figure 92**, an example of a group of entities is triggering a cluster alert after grouping together into a cluster (right image). The left image shows the entities when they are still too far apart to trigger the event. **Figure 93** shows a screenshot of an alert being triggered by an entity entering within proximity of a user-supplied polygon.



13:53:15 ClusterEvent

Figure 92. Clustering Alert

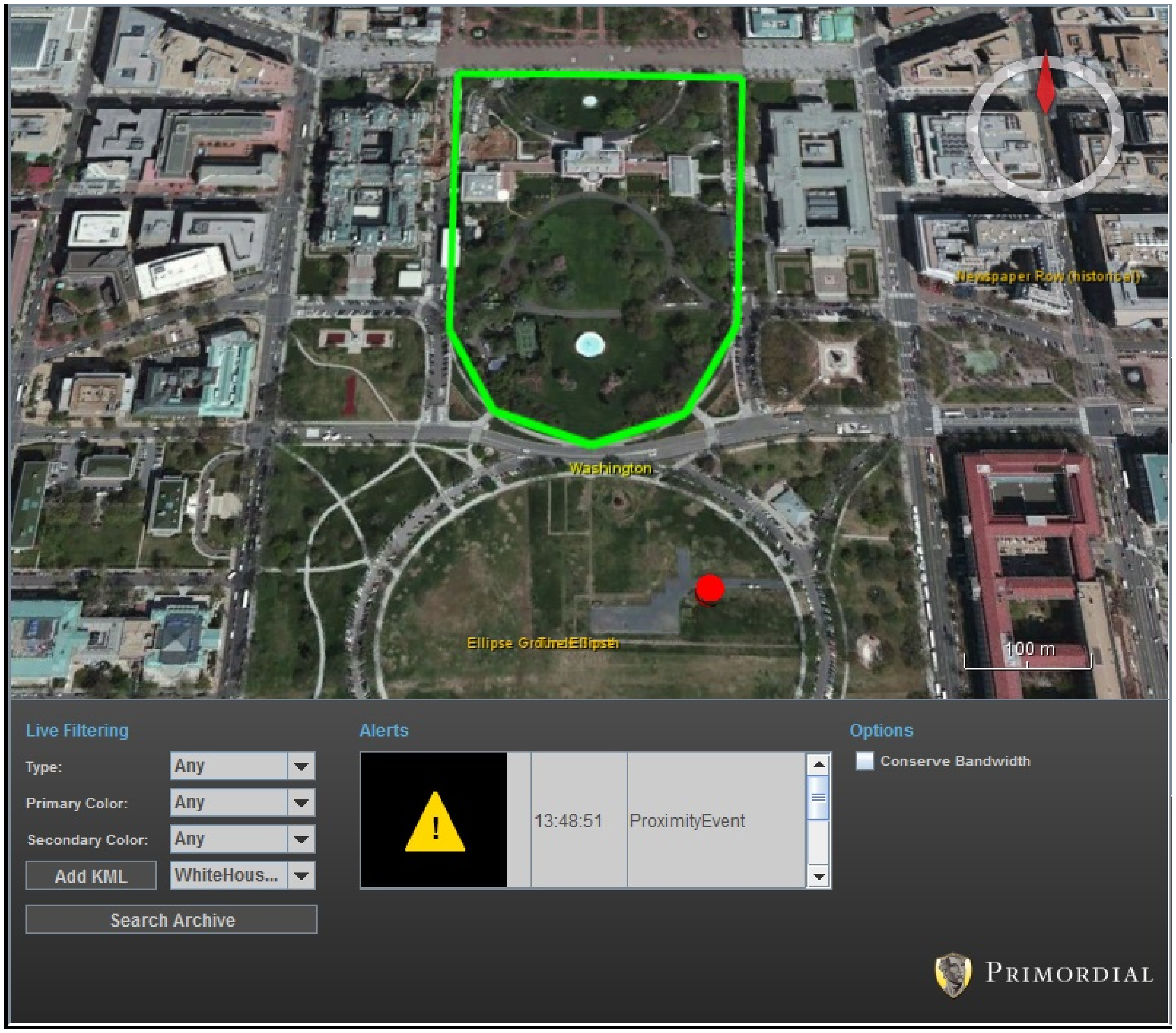


Figure 93. Polygon Proximity Alert

Primordial continued experimenting with optimizations for queries and implemented bug fixes for the polygon proximity query. Prior to these changes this query was taking 28min with a dataset of 8000 tracks and 1200 military base perimeters, and was substantially slower than the other live alerts. Initially, Primordial attempted to optimize the query times by switching from quad-trees to R-trees (4), the latter having the advantage of maintaining a balanced tree. Unfortunately, this change had a minimal impact on query times. However, eventually a bug was uncovered in the code for specifying the bounding box envelope for the tree search, and fixing this issue drastically increased the speed of the search. See **Table 4** for the measured times for these proximity approaches.

Table 4. Optimizations to Polygon Proximity

Optimizations	Time Measurement
Baseline	28 min
Fixed envelope bug	0.12 sec
Switch to R-trees from Quad-trees	0.15 sec

Table 4 shows the time to perform the proximity query with 8000 entities and 1200 polygons, before (baseline) and after optimizations.

Primordial then began work on an informal design for a mechanism to allow users to dynamically subscribe and unsubscribe from alert events. The team considered the following high-level goals of the alert mechanism implementation. First, it was desirable to change the method by which the server reports information to clients from a broadcast model to an individual notification model. This is more in-line with what typical applications provide when dealing with database queries. It would send back data to only the client which requested it, rather than announcing the query results to all clients connected, something only really practical in a prototype/testing scenario. It was also essential to use a unique ID to identify a particular client. This would allow the system to return the correct data to the correct client, as well as to associate specific settings and parameters with a specific client. It would be necessary to modify the client application to allow the user to subscribe to specific alert types and to specify the required parameter information for the alert chosen, as well as to allow the user to unsubscribe from events they'd previously subscribed to. Another essential modification would be to modify the server to allow dynamic creation/removal of Nesper query objects from the user request. In the first implementation, the queries in Nesper were hardcoded. In order to make subscriptions dynamic, it would be necessary to enable/disable queries for specific users on the fly. The server would need to be modified to store and cleanup clients' subscription information, to facilitate sending queries and to remove old data after client unsubscription or disconnect. Additionally, it would be necessary to verify and test that the new alert notifications work in the same manner as the previous hardcoded style alerts did and also to verify and test large-scale use of the alert system (many subscriptions, many incoming events, both at the same time) to discover any issues with scalability.

After the design was completed, Primordial began a preliminary implementation. To start, the system was modified to restrict sent and received information to only the clients who should receive it. Prior to this point, a multicast scheme was used where all clients would receive all information from the server. This improvement was accomplished by assigning each client a GUID (Globally Unique Identifier) ID. Additionally, to support this functionality, Primordial added a dictionary structure to the server to store parameter and Nesper object sets individually for connected clients. This included code to store subscribed alert sets for each individual user and also provide storage cleanup in the event of disconnected clients, as well as a "heartbeat" check to evaluate whether clients were still connected. To support the alert subscription model, the networking code was modified to facilitate sending the subscription requests and responses by including subscribe/unsubscribe types and modified packet types. Primordial modified the Tentacle client application to create subscribe and unsubscribe windows as well as accompanying GUI elements for specifying parameters for each alert type; this also included creating packing code for complex parameters like KML shapes. Primordial then modified the Nesper processing code to allow dynamic assignment and unassignment of EPL (Event Processing Language) queries from processed alert subscriptions. For event types using geographic shapes, Primordial modified the existing code to use the previously created storage system mapping the specified shape/query parameters to the requesting client ID for the individual subscription. This was a significant improvement over the existing mass-storage tree-based system. Primordial then implemented working modifications of the following types: Cluster, Containment, Cross, Enter, Exit, Loiter, Proximity using the subscription system. After implementation was complete, Primordial thoroughly tested the alert subscribe/unsubscribe mechanism. This included verifying that the correct results were returned to the correct clients and that the subscription mechanisms behaved correctly, including when using multiple clients and heavy loads. The following figures (**Figure 94**, **Figure 95**, **Figure 96**, **Figure 97**, **Figure 98**) show the alert subscription process.

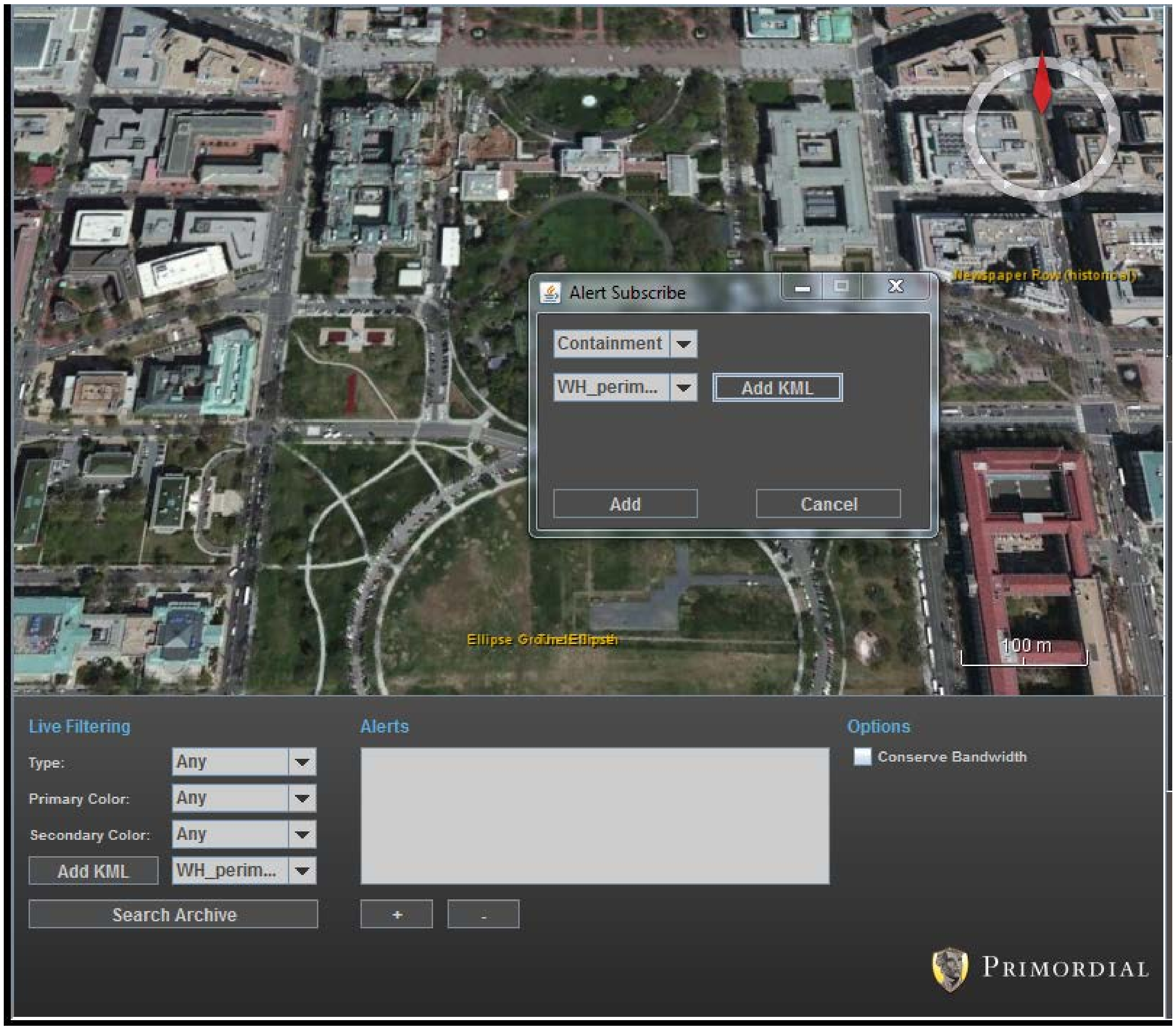
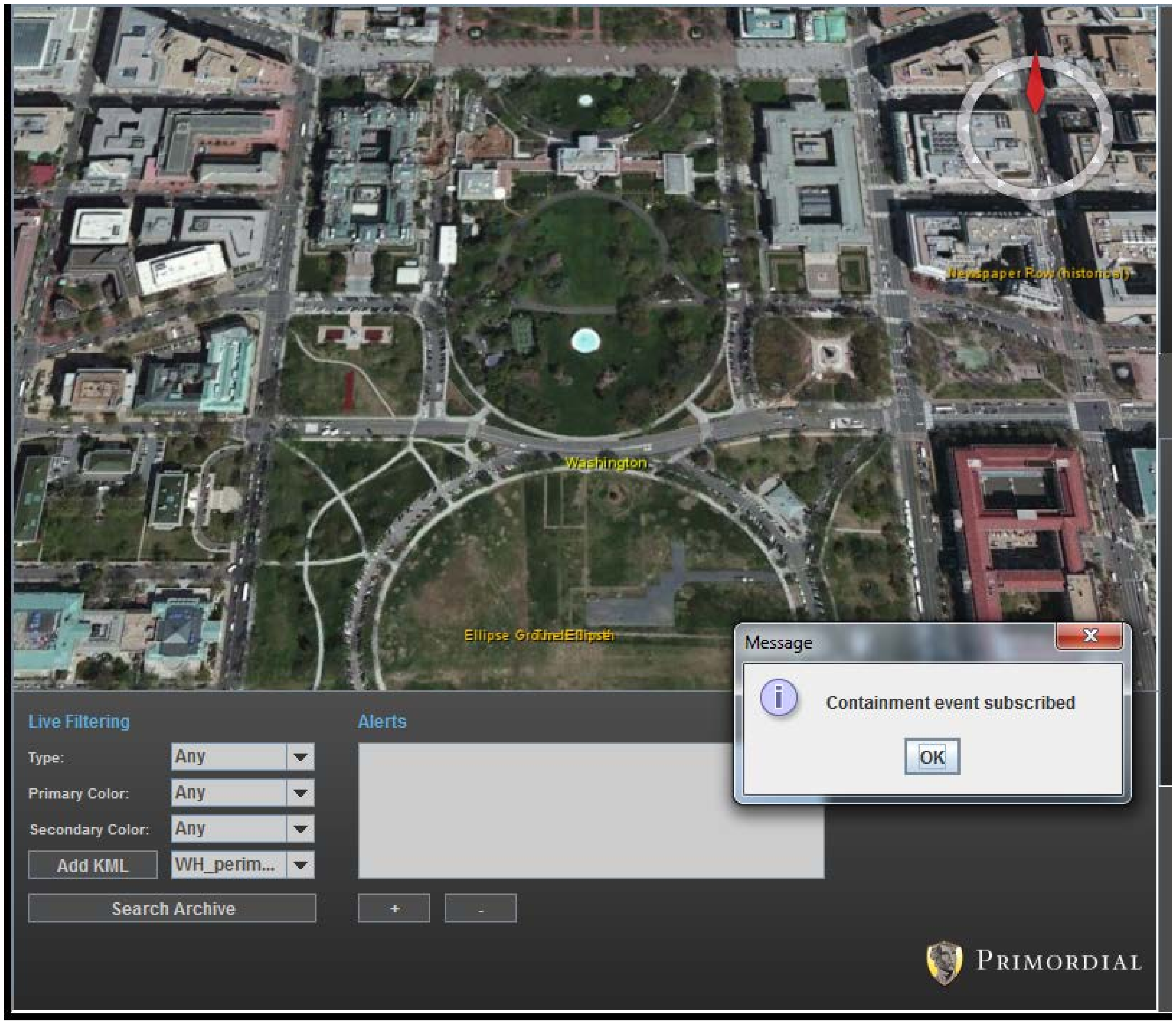
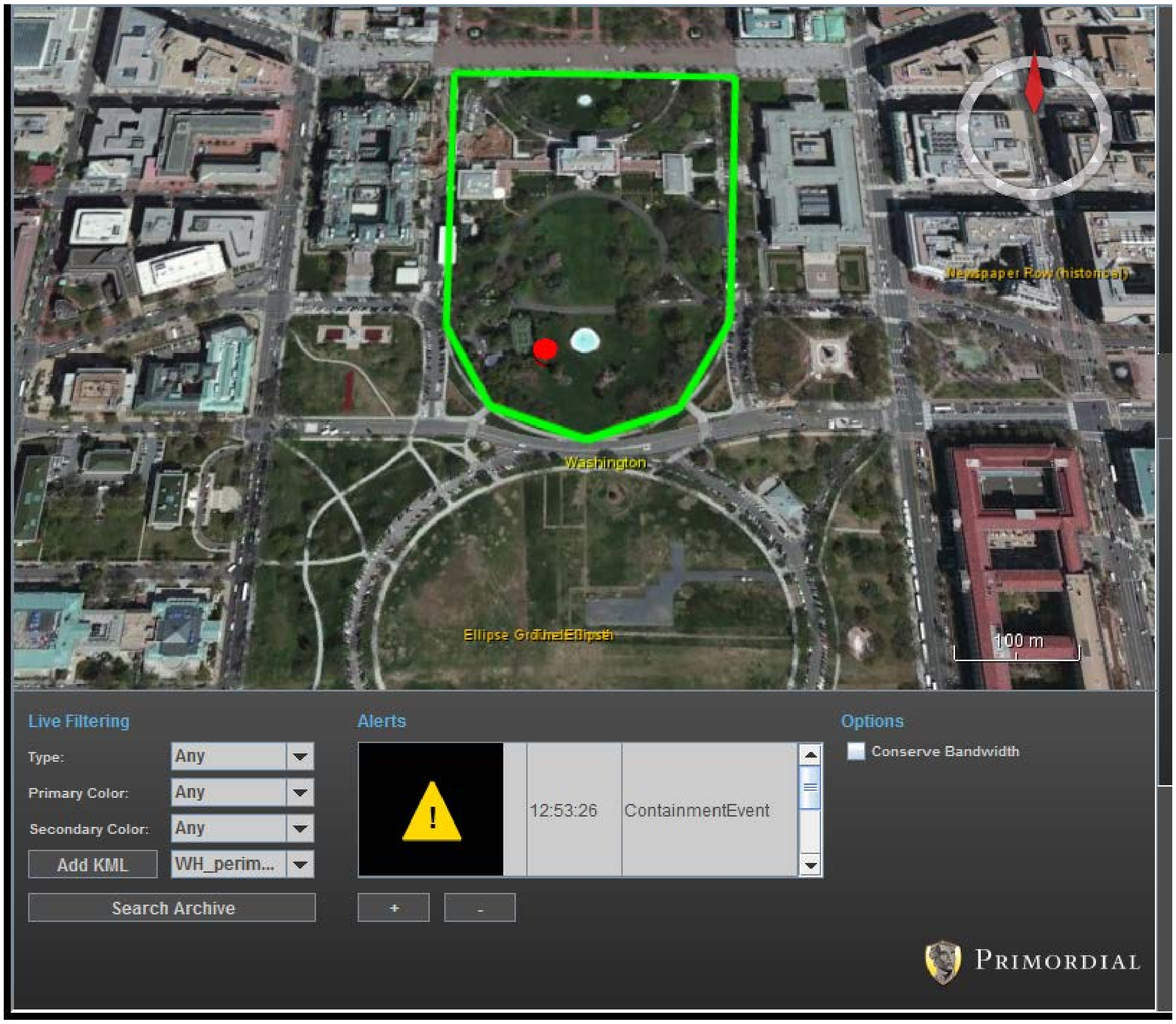


Figure 94. Subscribing to an Alert



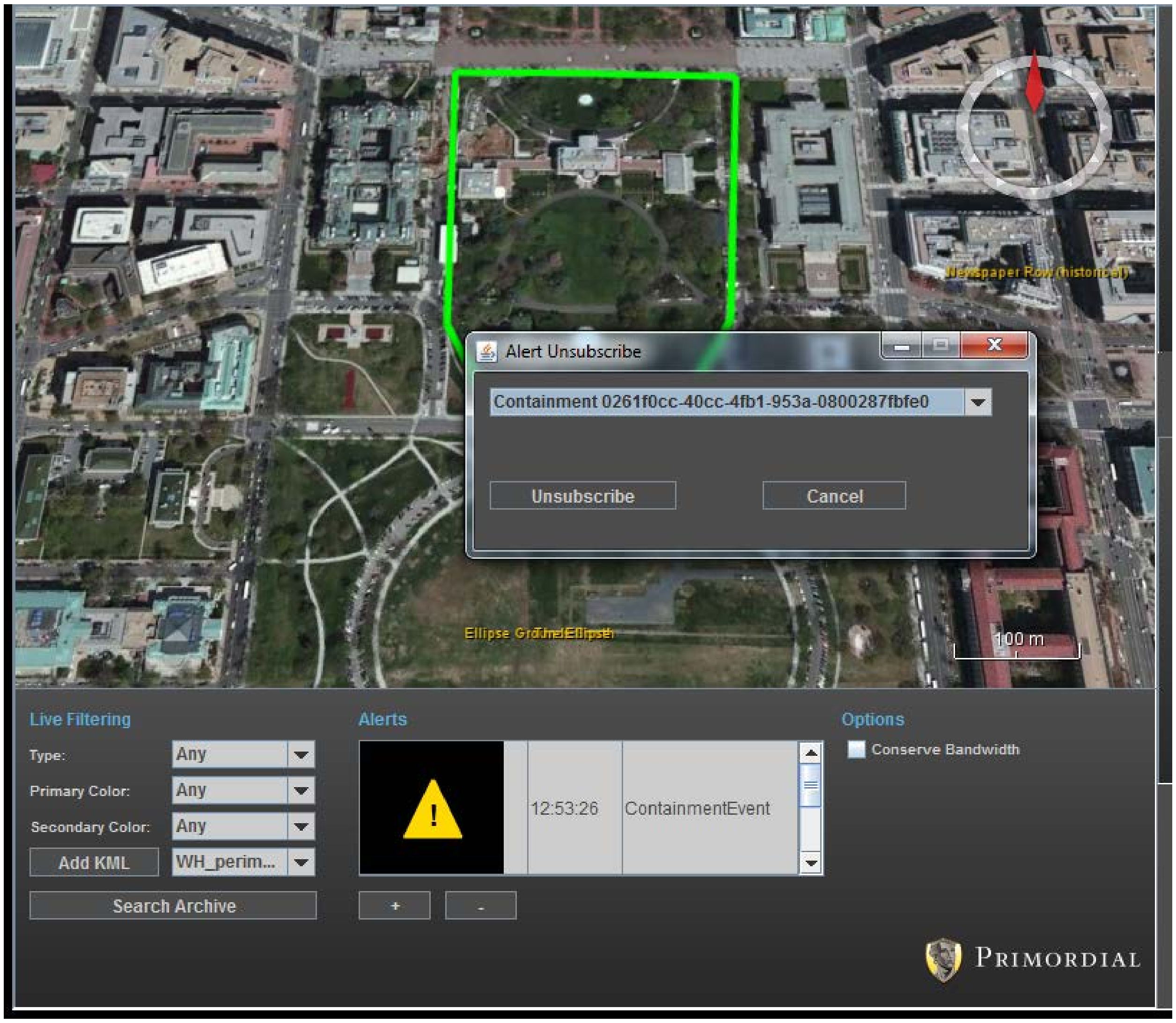
Containment event subscribed

Figure 95. Subscription Confirmation



12:53:26 ContainmentEvent

Figure 96. Alert Notification is Triggered



Containment 0261f0cc-40cc-4fb1-953a-0800287fbfe0

Type: Any
Primary Color: Any
Secondary Color: Any
WH_perim...

!	12:53:26	ContainmentEvent

Figure 97. Unsubscribing From an Alert

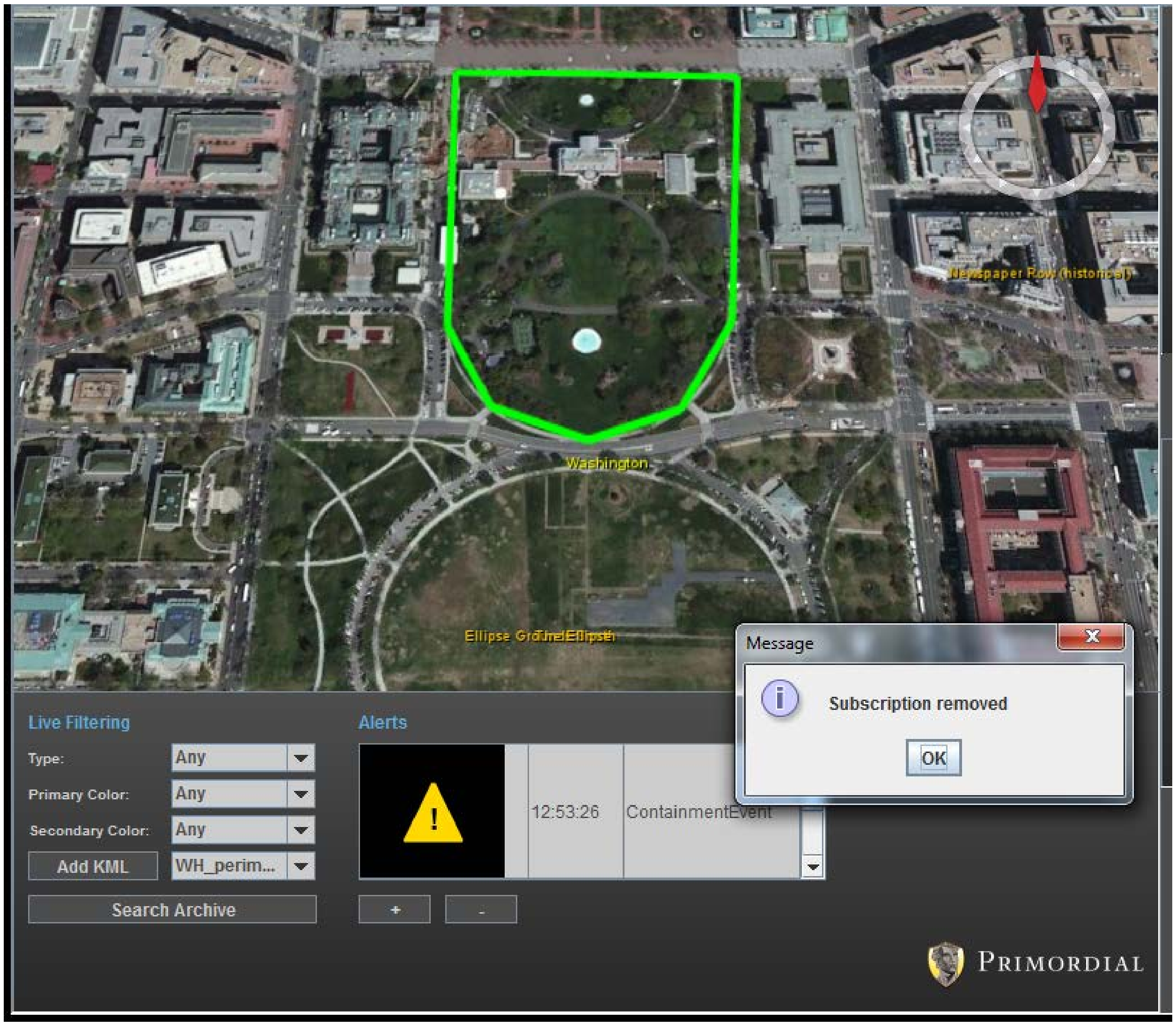


Figure 98. Unsubscription Confirmation

At this point, the Primordial team performed a thorough review of the code. They decided that, due to a significant amount of very similar code between the archived and live alert systems, it made sense to combine them into a single codebase and tool. In this new version, the user would provide parameters in the same manner for archived or live alert queries; however, there would exist a simple switch to determine whether to set the alert for past events or future ("live") ones. This merging of the tools allowed for easier development and testing as the incoming data is treated the same way in the underlying code. The difference from the previous setup is that for archived event searches in the new implementation, the system would preload archived events from the database, preregister an alert, and then feed the preloaded entity records into the system as if they were live (currently reported) events. After the data was fed into the processing system, the alert would be immediately unsubscribed.

To facilitate this change, the Primordial team modified the client user interface to have one single dialogue window for alerts with consistent GUI components on each with a simple pulldown object to allow the user to select archived or live alert types. **Figure 99** shows these changes.

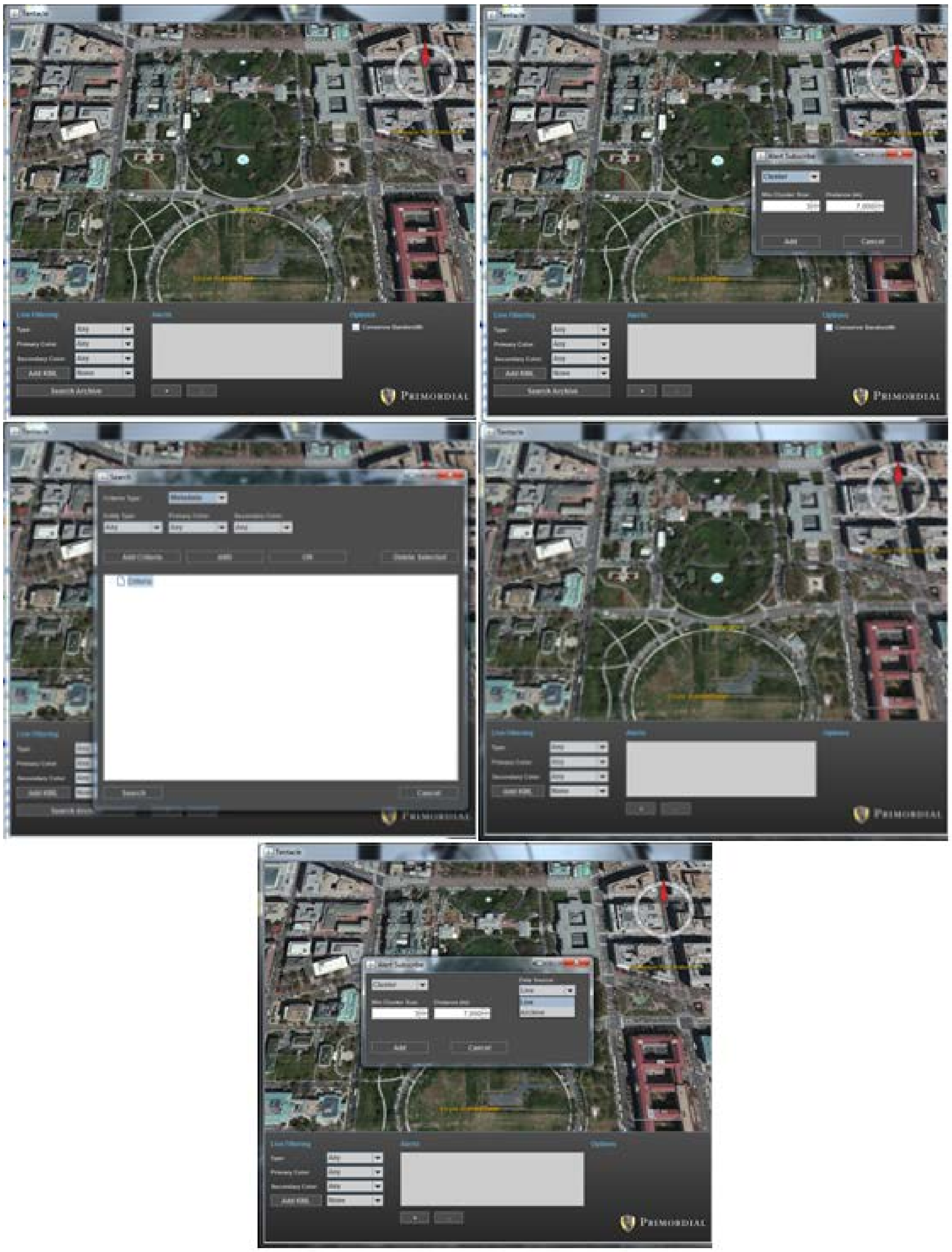


Figure 99. New GUI Interface for Alert Subscription

During the later stages of the project, Primordial determined that the Nesper system was not as flexible as the team desired. Adding new query types involved significant effort to write using Nesper’s proprietary coding syntax. Because of this, Primordial decided to implement its own in-house live event processing solution similar to Nesper. This would allow the team to modify the underlying system as needed in order to more effectively support the project’s objectives and future improvements.

Primordial also implemented new classes to better organize and process the live alerts. These helped to make testing and evaluation of the system more efficient and reliable. Primordial also modified the server and client code to pass the entire list of entities involved in an alert detection event to make forensic analysis of the event report easier based on context. Previously, only the points that flagged the event were being passed back; however, this approach loses the situational context of the detection which can provide the user more information about the scene. Primordial felt that in many cases the loss of such information was a major disadvantage and so it would be more useful to pass all of the entities that could be involved.

At the final stage of the process, Primordial focused on planning and creating real-world and artificial test cases to verify that the output of the alerts was valid and met expectations determined by the team.

3.8 Support Large-Scale Tentacle Server Deployment

In Phase II, Primordial made several modifications to the Phase I Tentacle system to facilitate large-scale deployment. Initially, Primordial examined the rate of processing for the video streams. Video processing had been rather slow and prone to delays. It became apparent that the cameras used by the Phase I system only supported low bandwidth streams and were only able to be accessed by one computer workstation at a time. Because of this, in order to simulate large-scale usage, it was necessary to find a way to rebroadcast the camera streams using some sort of repeater system to allow multiple workstations to access the video. The team investigated using third party tools such as VLC (VideoLan software application) video player to take in the initial camera feed on one machine and then rebroadcast it across the network to multiple clients in a more robust way. The team was able to successfully accomplish this rebroadcasting scheme but work was halted on this until there was a more robust design for the overall system. In order for it to be useful in a large scale system, there would need to be significant manual setup, which may or may not be acceptable.

Primordial decided that it would be useful for the user to be able to view the original camera feed being monitored for targets. In order to do this, Primordial upgraded the client GUI application to directly connect to the camera video and display it within the client application to the user. This was initially done by directly hard-coding the address to the camera within the application as a first-pass. Later, Primordial decided that marked up video was more useful to the operator and modified this structure to take in the processed video output frames coming from the Tentacle detection application for display to the user instead of the original raw video frames. In addition to the video connections, the Tentacle team spent significant time working on improving the networking code and making it more robust to packet loss and accidental server/client disconnections. There were several subtle, yet major, bugs discovered in the networking code during development which needed to be rectified in order for the system to be usable on a large-scale level.

Because large scale deployment of the Tentacle system was likely to encounter large quantities of target data, Primordial focused the bulk of its efforts on testing this scenario. For generating staged entities, the Primordial team used an in-house client application called TentacleSimulator. The team modified this application to have an option for reporting of mass flight record locations worldwide to simulate a large reporting load from Flight24 (15). This allowed the team to identify issues with ingestion of large quantities of points simultaneously, or within a very short time period. It also assisted in revealing the effects of large data quantities on the network packet processing code as well as on the Nesper alert processing system. Primordial discovered and rectified several major network code bugs that didn't occur until receiving thousands of network packets simultaneously.

During testing, it became apparent to the team that the broadcast approach to returning database data from the system used in Phase I was not appropriate for a large scale system. In Phase I, the server was broadcasting all return data to all clients. To improve this, the team modified the system to unicast information directly to individual "subscribed" clients based on stored session data. In addition, the team developed code to better provide internal maintenance of storage variables, stored geographic shapes and structures when clients were prematurely disconnected. These modifications created a more mature approach to sending client data and allowed the system to be more scalable in that only the essential amount of network traffic was being generated, rather than unnecessary broadcasting.

Another major concern with scalability was the number of supported simultaneous clients. Although, in Phase I, multiple connected clients were possible, the system was not set up to cater to large numbers of them. In order to test scenarios with multiple simultaneous clients, Primordial connected several physical machines fitted with the Tentacle GUI client and exhaustively tested subscribing to alerts, making requests, and following targets simultaneously while the system was processing large numbers of data points. The results of the test met initial expectations of the team.

Because the client application's settings were initially mostly hard-coded within the source code, Primordial added configuration file settings and code in the client application. This allowed the user to specify the two camera feeds in a configuration file rather than having the locations hardcoded in source as was done previously. This provided more flexibility and more customization for the end user as it no longer required manual changes to the codebase and recompilation, which would be impossible for an end user of the system.

Finally, the Primordial team spent significant time testing and making adjustments to the server and client code to maintain and repair connections between the client and server and from the client to video streams. This allowed temporary network disconnections to be rectified easily and make the system more robust in a large scale network topology. Additionally, code was added alert the user to problems encountered with these connections to provide more transparency.

3.9 Improve Tentacle Client GUI

The Primordial team made significant modifications to the client GUI from the version shown at the end of Phase I.

At the beginning of Phase II, Primordial updated the Phase I GUI to support overlaying text on the processed (post-tracking) video frames to give a description of the data inside the entity bounding boxes. This greatly assisted debugging and better identified for the user the targets being processed by the tracking system. Primordial also fixed issues with avatar display on the WorldWind map, where target locations were not being properly specified. Avatars were being displayed slightly offset due to inaccuracies in projection calculations.

During testing, Primordial encountered with problems with displaying map information on WorldWind on some new machines. The team spent some time investigating this issue and discovered that there is a known bug with graphical drivers on some systems, but that a graphics card driver update resolved it on all systems tested. This was noted in the documentation to assist the user in troubleshooting any problems while installing and configuring Tentacle at a later date. The team worked on refining the client GUI to be more user-friendly, including removing borders from video panes to maximize content space, equalizing spacing of elements on the page to better "flow" and other similar cosmetic details.

One major change to the client GUI was the addition of the archived alert system. A dialogue menu was added to allow the user to query the stored database of recorded entities. This initial menu was later expanded to support spatial queries. This entailed adding functionality to allow the user to specify geographic shapes via a file open dialogue and selection object. Primordial added additional functionality for the user to specify logical (Boolean) metadata criteria for determining which data would be returned. At later stages in the project, when live spatial queries were being integrated, the team redesigned this dialogue to provide the user with tools to subscribe to live (on-line) spatial and logical metadata alerts as well as the original archived query functionality.

At the end of the project, Primordial consolidated the elements in the query dialogue menus to be a single set of components, with archived and live alert subscriptions being treated the same. This created a more uniform experience for the user and a simpler interface. The user was able to toggle between live and archived functionality using a pulldown selector in the window. Additionally, toward the end of the project, the team made some final cosmetic adjustments to the client. This included visual notifications to the user within the video frame that the video streams had become disconnected, and notifications about server connection loss. Additionally, Primordial added auto-reconnect functionality to video streams when they again became available.

The team began an effort to develop a TentacleSDK as described in the Statement of Work to make the system more expandable for other GUI clients. An initial design and method stubs were created, but development was halted on this to focus on more pressing development needs for the system.

3.10 Test, Evaluate, Improve, and Demonstrate

During the entire period of development on the Tentacle system, Primordial and partners performed regular evaluations of the system as a whole and made modifications to the code and SDKs to improve performance and reliability. These adjustments covered all sections of the code, but especially focused on improving video processing time, entity detection accuracy and reliability, event processing and networking code robustness and performance.

In the beginning of Phase II, the Primordial team identified the video processing as weak point in addressing the Statement of Work's "real-time" requirement for the system. The team increased focus on trying to evaluate and improve the video processing code, which proved to be a challenging problem. The team performed time evaluations of video processing for the various individual segments of the processing pipeline, including the initial video streaming to the detection application, the internal detection processing, IntuVision SDK, and all relevant intermediate steps. The team made adjustments to the code to improve each of the various components to address defects. In the initial part of the pipeline, the video feed coming into the detection application, the Primordial team identified an issue where the video cameras providing the incoming video stream were not capable of handling large numbers of connected viewers. This would slow down the input into the system for tracking. To rectify the problem, Primordial set up a rebroadcasting system which took the load off of the cameras themselves and shifted it to more powered intermediate repeater nodes. Also during testing, the team found failures in the handoff of individual video frames between the IntuVision detection SDK and the Tentacle detection application. This was due to a memory concurrency error and produced skipping video or out of order processing of frames. With IntuVision's assistance, Primordial was able to rectify the issue by moving processing code from the callback method for the IntuVision SDK to later down the pipeline. This reduced lag time in the callback method and allowed the system to process frames much faster. Later in the project, specifically when high-resolution imagery was introduced, the system encountered out of memory errors after a short period of video processing. Even if an error wasn't thrown, the video processing was significantly slowed and became progressively more so as time went on. Primordial discovered a memory leak in the video processing code where references to frames were being inadvertently held on to after the frames were processed. This caused the frames to be retained and not disposed of by the .NET framework's garbage collection process, which eventually led to the system running out of memory. To fix this, Primordial changed the frame processing code to use a reference system with a centralized internal store in the form of a queue for frames. This allowed a single central location for frames to be created and disposed of within the server and ensure that frames were properly disposed of after processing. A similar system was set up in the Tentacle detection application to deal with frames there as well. This drastically increased the speed and reliability of the video processing in Tentacle.

Another major area of testing and improvement was the accuracy of entity reports. While in Phase I the focus was mostly on static detection and 3D projection, in Phase II, the majority of the effort was on entity tracking from a moving platform, especially UAVs. The team developed several algorithms for this which are described elsewhere in this document. However, during the development of the code to implement these algorithms, there were repeated iterations of

improvement to increase the accuracy of target projections on the ground plane. These included acquisition of real-world UAV telemetry data to use as input, as well as manual precise identification of UAV video corner points when the UAV telemetry proved inaccurate.
Live event processing was a significant portion of the work on the Tentacle project during Phase II. The live event detection underwent many different revisions and constant testing during the development process. In addition to in-house testing and refinement, Primordial acquired externally created data (from Flight24 (15)) to use as a real-world metric when evaluating the performance and accuracy of the live alert system. Primordial found this very effective in revealing problems with high-volume processing in the live alerts and in the server code itself.
In a related vein, Primordial focused a large amount of effort on improving the networking code for the system. In Phase I, the networking code was basically a prototype. It worked on a small scale, in its focused, low-volume domain. However, when the requirement for scalability was emphasized, Primordial refined the code to handle large amounts of data, large size data packets, packet loss and other related networking issues. Additionally, Primordial modified the server and client code to automatically reconnect if the connection was interrupted for some reason. This provided vastly improved performance and reliability over the Phase I system.

In addition to performance and reliability changes to the code, Primordial also instituted an automated test system to verify the output of critical code sections and prevent accidental reversions of bug fixes. To do this, Primordial installed, tested and integrated NUnit as an automated testing solution for the C# sections of the system code. Additionally, the team integrated JUnit as an automated testing solution for the Java-based GUI client application.
At the end of the project, the team conducted an extremely thorough end-to-end evaluation of the entire Tentacle system to determine which areas needed the most focus for development during the final phases. This helped to direct the team's efforts to make the system as solid as possible before the final demonstration and avoid focusing on less-important aspects with minimal remaining time.

Once new development was frozen and bugs had been fixed, Primordial conducted a thorough cleanup of the entire system's codebase, removing unused classes, code and comments which were part of alternate prototypes and functionality. This was done to help made the code more understandable for later (post-Phase II) improvements and integrations by other parties.
In October, 2014, the Primordial team demonstrated the final version of the system live at AFRL with the Tentacle partners assisting via telecon. Primordial gave a PowerPoint presentation with an overview of the entirety of the project, then demonstrated the system on a laptop for AFRL representatives.

4.0 RESULTS AND DISCUSSION

4.1 Improve Static Camera Tracking

Although the focus in Phase II in terms of tracking was more on moving target tracking from UAVs, the Tentacle team significantly improved on the static tracking shown in Phase I. Primordial and IntuVision conducted several rounds of video sample testing to identify issues. IntuVision added the Lab/CIEL model to provide robustness against illumination changes which were a roadblock for the Tentacle static tracking system in Phase I. They also provided extensive guidance and suggestions for fine tuning the implementation of the tracking code in the Tentacle system as well as camera settings. Primordial adjusted the original 3D projection algorithms to improve accuracy. IntuVision updated their SDK to help reduce “ghost objects”, another long-standing issue in the Phase I implementation of the static camera tracker. In addition to resolving weak points in the Phase I implementation, the team improved the camera resolution processing capability to get better results from the tracking system and also purchased a high-resolution camera for testing. IntuVision added the capability to reduce headlight glare from vehicles. This assisted in improving the range of time where the system could be used from daytime only to potentially in a nighttime scenario. These changes significantly improved the system over the version submitted at the end of Phase I.

4.2 Improve Entity Correlation

Primordial updated the Phase I client GUI application to only display entities tracked over multiple frames to avoid erroneous avatar display from artifacts or transient objects. IntuVision tested using a Local Binary Pattern model for extracting features, but this was found to be insufficient for Tentacle’s needs. IntuVision updated their color extraction algorithm to use advanced histograms which would make identifying objects in poor lighting/color conditions more robust and provide better correlation between camera views. Additionally, IntuVision improved on the Phase I vehicle classification models by generating multiple models for multiple views, thus removing the reliance on specific poses for classification. Finally, IntuVision experimented with cross-camera face matching using the LuxandSDK and developed a scheme for passing detected faces between cameras. However, Primordial was not able to replicate their results except on the single test video that IntuVision had provided. Additionally, the face had to be sufficiently close to the camera to provide enough pixels for recognition. Because of these issues, Primordial decided to refrain from integrating the facial detection/recognition functionality into the main Tentacle system and changed focus to other aspects of the project. The focus of phase I was on correlating the output of multiple static camera sensors to identify and geolocate real-world entities. In Phase II the focus shifted somewhat to developing similar technology for UAV sensors. The system works in a similar manner as with static cameras, but because of the unanticipated difficulty of processing UAV video and geolocating entities, the team was not able to make correlation of entities in this manner as robust as in phase I. Ideally, work on this would be part of future endeavors.

4.3 Improve Entity Geolocation

During Phase II, the focus for the team in terms of entity geolocation was shifted to UAV-based moving target detection; however, some changes to the Phase I client GUI were made to make reported entities smoother and more robust and the original 3D projection algorithm was updated to be more flexible and accept differing camera resolutions than what was available in Phase I. After these modifications, the team moved on to focus on UAV video geolocation. The team successfully implemented code which used the UAV video telemetry data in a similar way as the tie points in the original projection algorithm from Phase I. This provided reasonably accurate projections of entity locations on the map. More detail is given in the moving camera tracking section below.

4.4 Implement Moving Camera Tracking

IntuVision modified their SDK to take in bounding boxes and, later, pixel masks to identify targets for classification. They also updated their classification models to work better with UAV video sources by using the BIM (Biologically Inspired Model) approach.

Primordial modified the Tentacle system to take in UAV video metadata and use video corner and center point locations to create a homography matrix for projecting target pixel locations to real world locations. When initial provided UAV telemetry was found to be faulty, Primordial created a manual setup to generate test data for demonstrating the system's capabilities. Primordial create in-house approaches for moving target tracking to move the project forward and as a fall-back option to the tasks assigned to CMU. These provided reasonably successful results, but were limited by the fact that they required manual seeding of target locations to start the tracking process.

CMU provided a Linux-based SDK for performing tracking, as well as a sample tracking application. The CMU tracker supported template and color based approaches as well as simultaneous tracking of multiple entities in a UAV video as well as automatic video stabilization. The CMU visual tracking SDK had been extended to provide an interface for accessing target masks corresponding to the estimate of foreground and background pixels in each active target track sub-window. During the development process, CMU encountered significant difficulty as moving object detection on a moving background proved very problematic. Additionally track acquisition and maintenance is extremely difficult; often maintaining a track proved impossible if a target went out of sight for a period of time or the camera suffered extreme vibration and so on.

Primordial modified the sample application to use the SDK's target masks and output them along with their corresponding video frames to a network share location for ingestion into the main Tentacle tracking program.

Although CMU did work on implementing automatic detection of target entities for seeding into the various tracker modules, it was not completed by the end of the project. Primordial's in-house tracker modules also suffered from inability to solve this problem.

One other problem encountered by all of the members of the team was that video resolution had to be reasonably high to get acceptable results, especially in moving video. This especially affected the classification process. The team found the best results in mid-range high resolution video. Anything too far away (even if high resolution) made targets too small to reliably classify; anything too low in resolution would not provide enough information to properly classify, even if tracking was possible.

Wide area imagery was investigated as a possible source for the Tentacle system. However, the file size, the low frame rate and the very small target sizes in frames proved problematic for providing any sort of useful tracking. The team decided to focus on more standard UAV imagery for the final product which produced more promising results.

In addition to the tracking process, integrating CMU's Linux-based SDK into the rest of the system (Windows-based) proved difficult. The team tried several options, including IPC and Message-Passing software libraries, but eventually developed a software adapter on the Windows side and used a shared network folder to pass data between the two systems. It was not an ideal result, but provided enough functionality that the team was able to demonstrate the final product.

Primordial and the rest of the Tentacle team evaluated UAV video samples from AFRL and decided on the VIVID set to use for the final demonstration. Primordial then selected a subset of the VIVID video set which promised to display the best range of scenarios for demonstrating the system.

The final version of the tracking section of the Tentacle system took in UAV videos with associated metadata stored in XML files alongside. The videos were processed through the CMU SDK on a linux machine through the CMU application. For each video, initial target locations were specified in a text file for input into the CMU application. The CMU tracker would use the points to pre-seed its tracking modules. While the video ran, for each frame of the video, an output frame would be created with an added alpha-channel indicating the location of the target entities. Alongside, a bounding box file was appended to, including the bounding boxes for the tracked entities. This data was written to a shared network folder. On the Windows side of the system, the Tentacle detection application looped and checked for available frames from the network shared folder. These frames were fed into an adapter class which took the bounding box file and the video frames and fed the relevant data into the IntuVision SDK. The detected entities were then classified by IntuVision's SDK and the resulting entities were reported to the Tentacle server, which recorded their metadata, including appearance, classification and location.

4.5 Improve Metadata Extraction

IntuVision improved their Phase I classification by introducing higher-granularity color histograms as well as adding more color regions for associating colors with different parts of the detected entities. They also added the ability for their SDK to ingest target masks and frames from pre-extracted entities. To improve classification, IntuVision trained multiple classifiers for different views of vehicles and people, which provided more reliable classification of entities.

To support IntuVision's efforts, Primordial updated their video processing system to support higher-resolution video and purchased an HD video camera to provide better input. The higher-resolution video performs better in tracking than the very low-resolution video sources used in Phase I.

IntuVision explored performing gender and ethnicity detection in video as well as facial detection and recognition and had some success. However, these algorithms required the faces to be quite large on the screen and due to the nature of the video the system is likely to process, it would not have been effective. Additionally, Primordial was unable to replicate IntuVision's results with anything other than one of the provided test videos. Because of these reasons, the team decided to refrain from integrating these features into the final system.
IntuVision examined classifying vehicles and people from wide area UAV imagery. However, they found that because the images were taken from such a high altitude, even though the resolution of the video was high, there weren't enough pixels for a target to reliably classify it. IntuVision updated their SDK to support classification of UAV entities using the previously developed BIM models and local edge-based features. The BIM models emulate object recognition in the human visual cortex and use local edge-detectors' combined responses to create different elements of the model. These changes provided good results for classifying entities in the UAV test videos.

4.6 Implement TIPL Using STANAG 4559

Although the focus of the project drifted more toward the alert system and moving camera tracking tasks, the Tentacle team did make some progress on this task during the effort.
The team set up a SQL database to store incoming entity metadata such as physical characteristics and geographic location.

Primordial modified the tracking application to store the frames where an entity was detected as a video clip in a network location and associate the file path with the tracked entities in the database. This allowed the user to later retrieve the marked up video clips associated with an entity and would not need to sift through long sections of irrelevant video to find the video sections needed to analyze a target.

Primordial modified the Tentacle client to allow the user to query the server for any stored entities. Once the database received the query, it would return results to the user for display in the client. The user could see the entity's tracked location on the map, as well as load the associated video clip in an external video player through a link.

4.7 Implement Spatial Query Engine

One major focus of the effort was the implementation of the spatial query engine into the Tentacle system. This would allow the user to set geographically oriented queries and alerts on archived and live incoming target data.

Primordial chose to use Microsoft SQL as the core database for this effort as it provided out-of-the-box functionality for geospatial calculations and queries.

Primordial modified the client application to allow users to input geographic shapes in KML format for use in specifying regions associated with created queries. Primordial also updated the Tentacle networking code to support spatial query request and result processing.

The team contacted SRI TerraSight (20) as a potential Phase III/commercialization partner or integration target for the final system. They had expressed interest in spatial query functionality using Primordial's core product "Ground Guidance" in the past. Representatives from Primordial and SRI met to discuss future opportunities, but no definite plans were established.

Primordial implemented several archived spatial queries for the final product: Proximity, Enter/Exit, Cross, Containment/Count, Loiter, Cluster. To support combinations of these and more complex queries, Primordial implemented a hierarchical Boolean logic-based system to combine them. However, this proved to be too complex to support and so the system was developed to support only individualized query subscriptions.

Primordial moved on to implementing live versions of the previously constructed query types. These acted as subscriptions for the user, where any incoming data would be processed through them and targets matching the criteria would be reported immediately to the user. To facilitate this, Primordial integrated the Nesper event processing system into the Tentacle server to support real-time processing of data through the spatial query algorithms. While this performed well, the query language used by Nesper was complex and inflexible. Additionally, it was a third party package, and as such there were questions about licensing in relation to this product. To mitigate these issues, Primordial implemented an in-house alternative to Nesper which allowed more flexibility in modifying the processing code as needed and reduced the dependency on a third party plugin.

The team performed many rounds of adjustments to the implemented query algorithms to improve response and processing time for these calculations to support the "real-time" requirement. Primordial also tested the algorithms using both manufactured data for testing and live entity locations from registered commercial flights. This provided large-scale processing tests which would help ensure the algorithms would perform properly under load. After repeated improvements, the algorithms performed at an acceptable rate for the final product.

Primordial modified the client GUI to allow the user to subscribe to and unsubscribe from live alerts using the implemented queries. They also modified the server to provide functionality to maintain connections in the case of short interruptions as well as behind-the-scenes cleanup of shapes and stored data in the event a client disconnected (either voluntarily or involuntarily). The final version of the alerts were tested and performed satisfactorily under the team's expectations. The following section illustrates and describes the state of the final system including each of the query types supported.

The GUI client provides a subscription interface for the user to choose the parameters of their query and the source of the data (live or archived). **Figure 100** shows this interface.

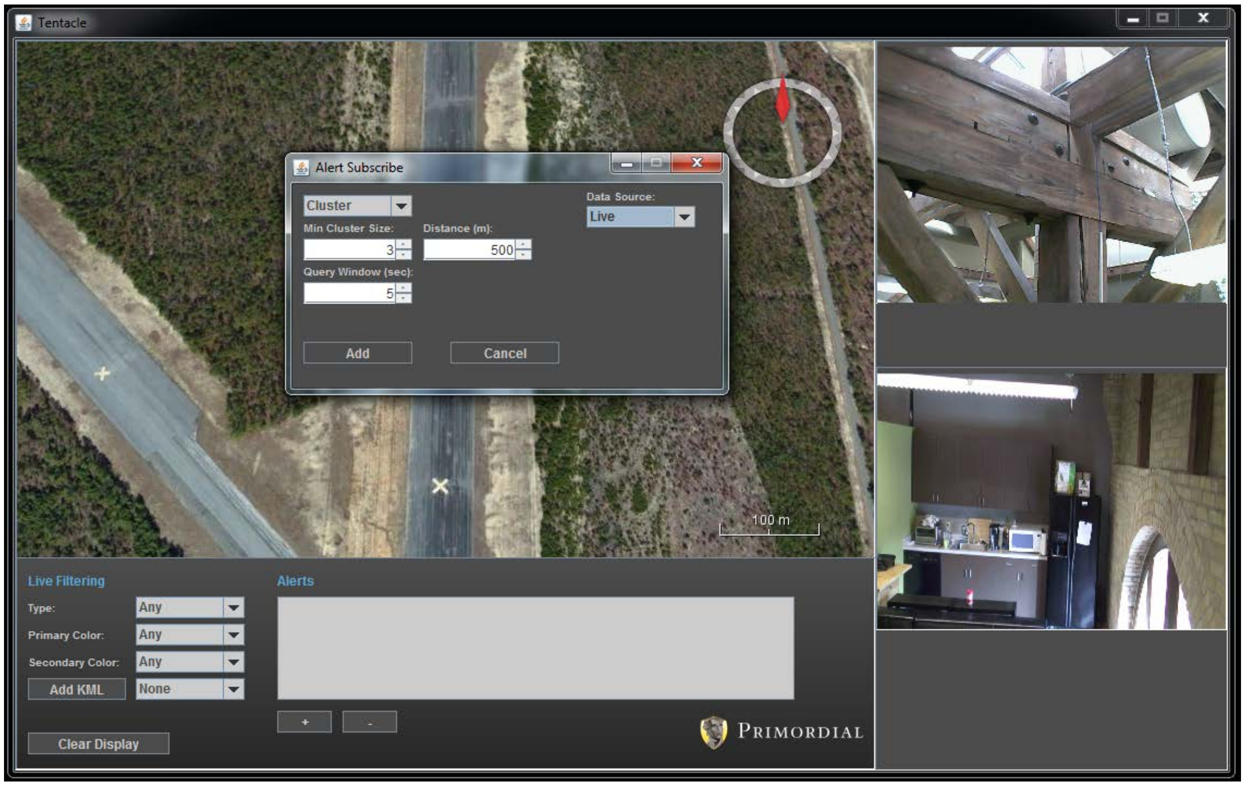
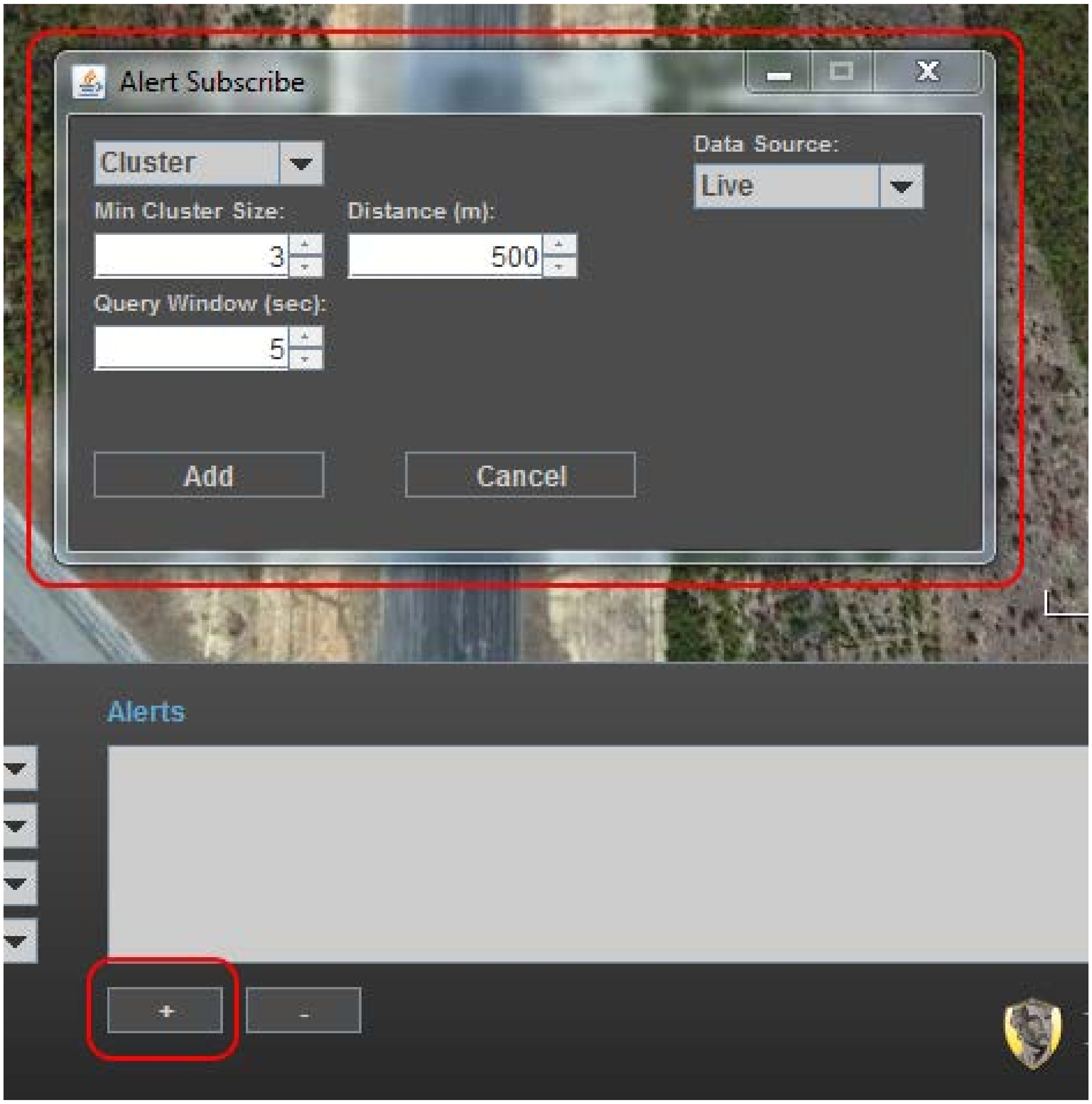


Figure 100. The Tentacle Alert Subscription Interface

To subscribe to an alert, the user clicks the "+" button on the main GUI screen (**Figure 101**). A dialogue window opens and the user specifies their parameters, then clicks the "add" button. The query is transmitted to the server which responds with a query ID. The ID is stored in the client and a message is given to the user that the subscription was successful (**Figure 102**).

To unsubscribe from the alert, the user clicks the "-" button on the main GUI screen. A dialogue window opens and lists the currently subscribed alerts. The user selects the alert they want to remove from the pulldown, then clicks the "Unsubscribe" button (**Figure 103**). The ID is sent back to the server with an unsubscribe request; the server then unsubscribes the client from alerts, removes the event and associated data and returns a response to the client application. The client then displays a message to the user that the unsubscribe request was successful (**Figure 104)**.



Alert Subscribe

Cluster

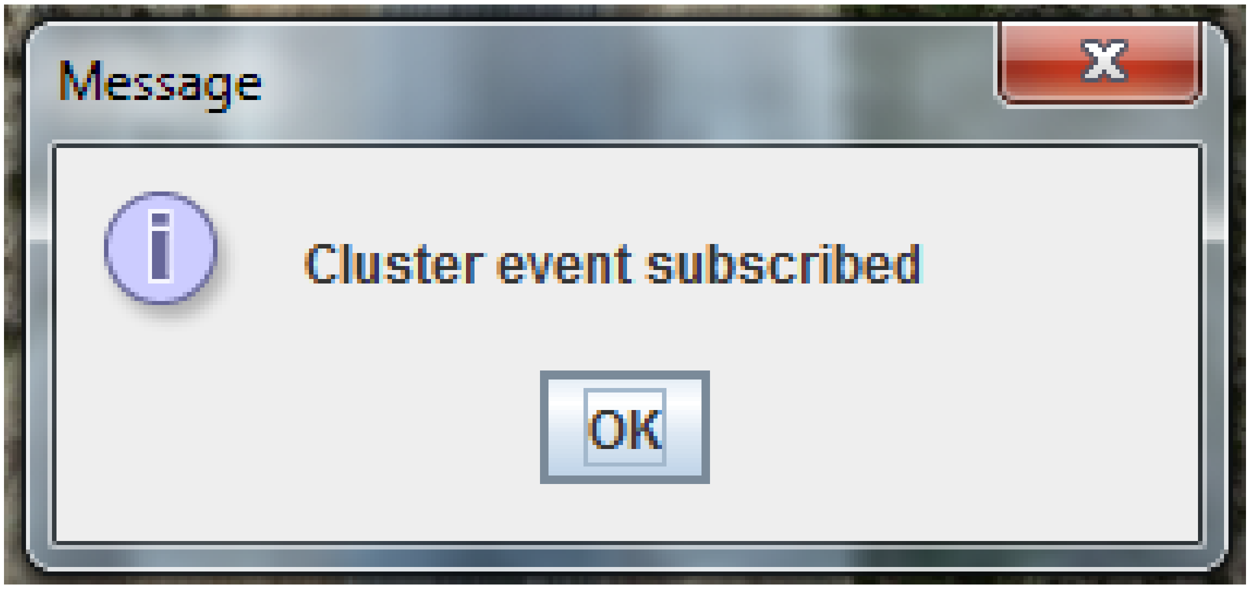
Min Cluster Size: 3

Distance (m): 500

Query Window (sec): 5

Data Source: Live

Figure 101. Alert Subscription Example



Cluster event subscribed

Figure 102. Event Subscribed Confirmation Dialogue

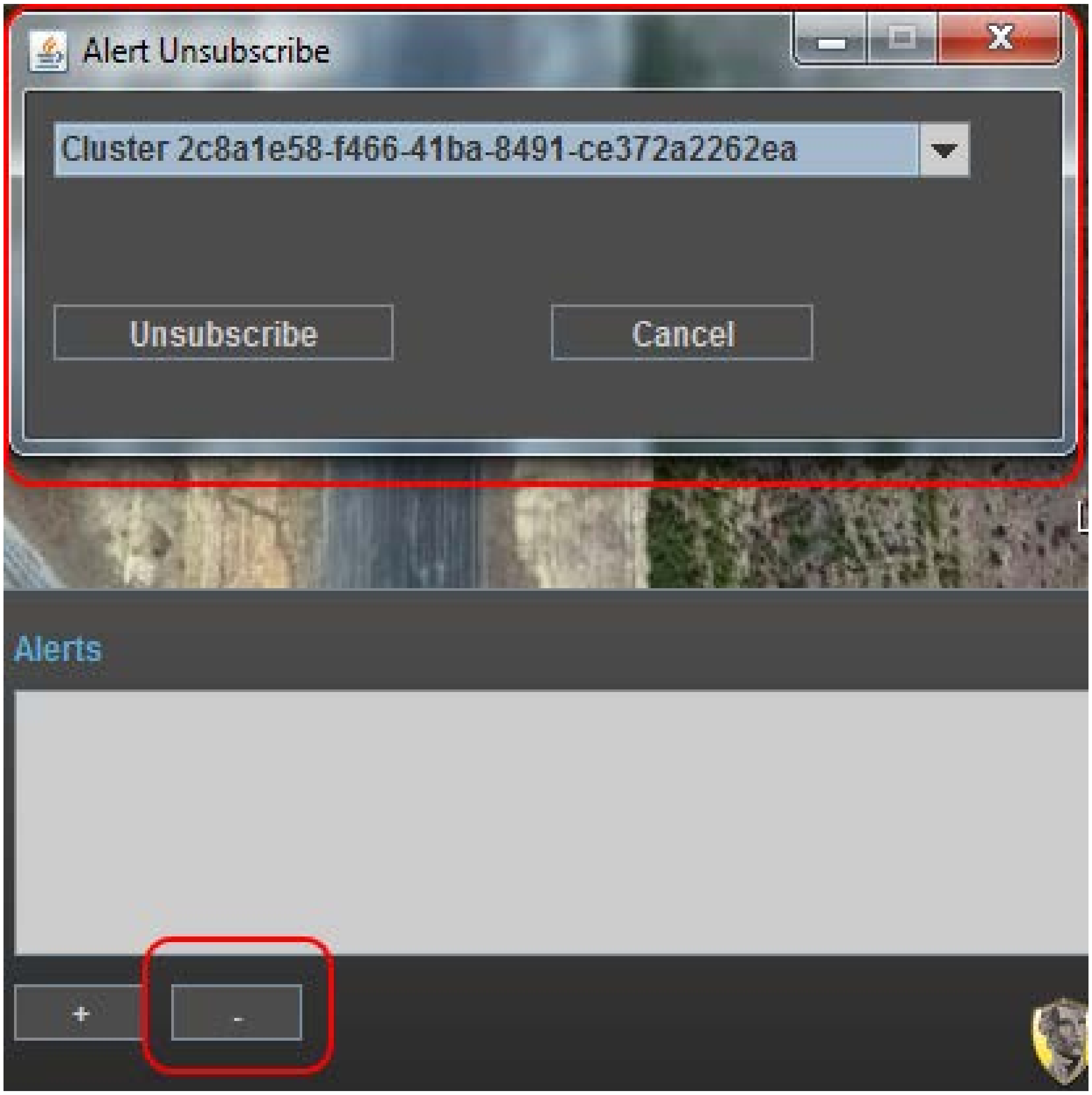


Figure 103. Alert Unsubscription Example

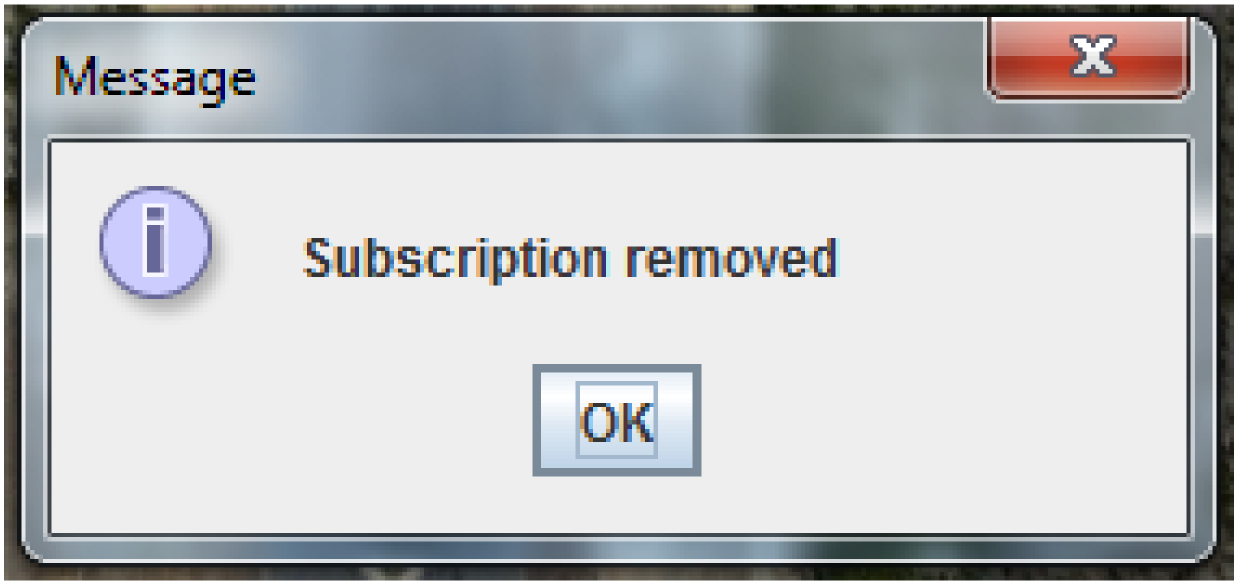


Figure 104. Event Subscribed Confirmation Dialogue

The following sections explain the behavior of the final query type implementations.

The Proximity query (**Figure 105**) detects two or more entities of interest that come within a specified distance of each other. Proximity alerts fire for each video frame or sensor input where the proximity evaluates to true. This provides a mechanism to perform state transition processing in the future, answering questions such as “is currently near” or “is moving away from”.

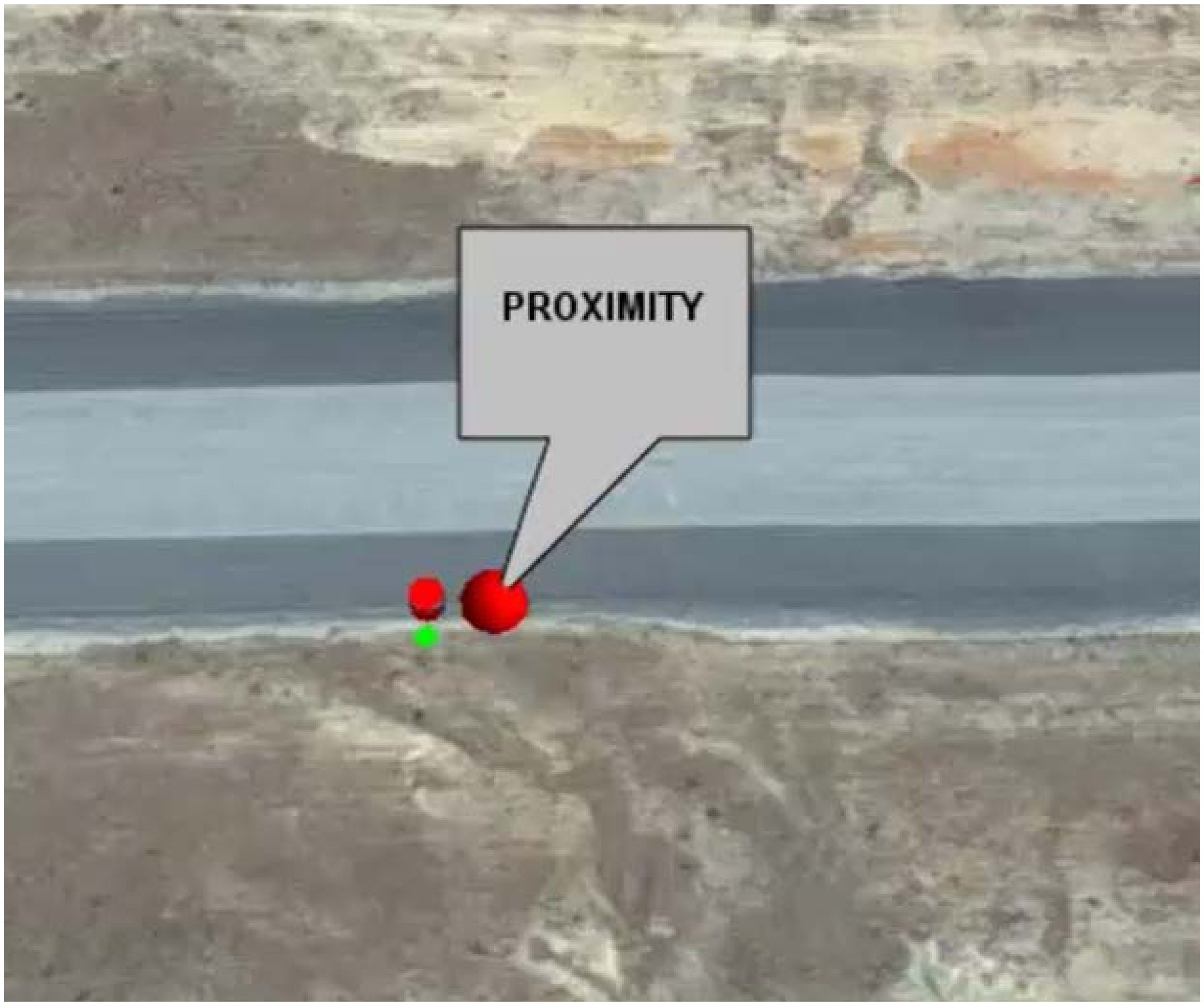


Figure 105. Detected Proximity Event

Containment (**Figure 106**) detects when a moving entity is within a specified area of interest. Containment is evaluated for each input (video frame or other sensor report).

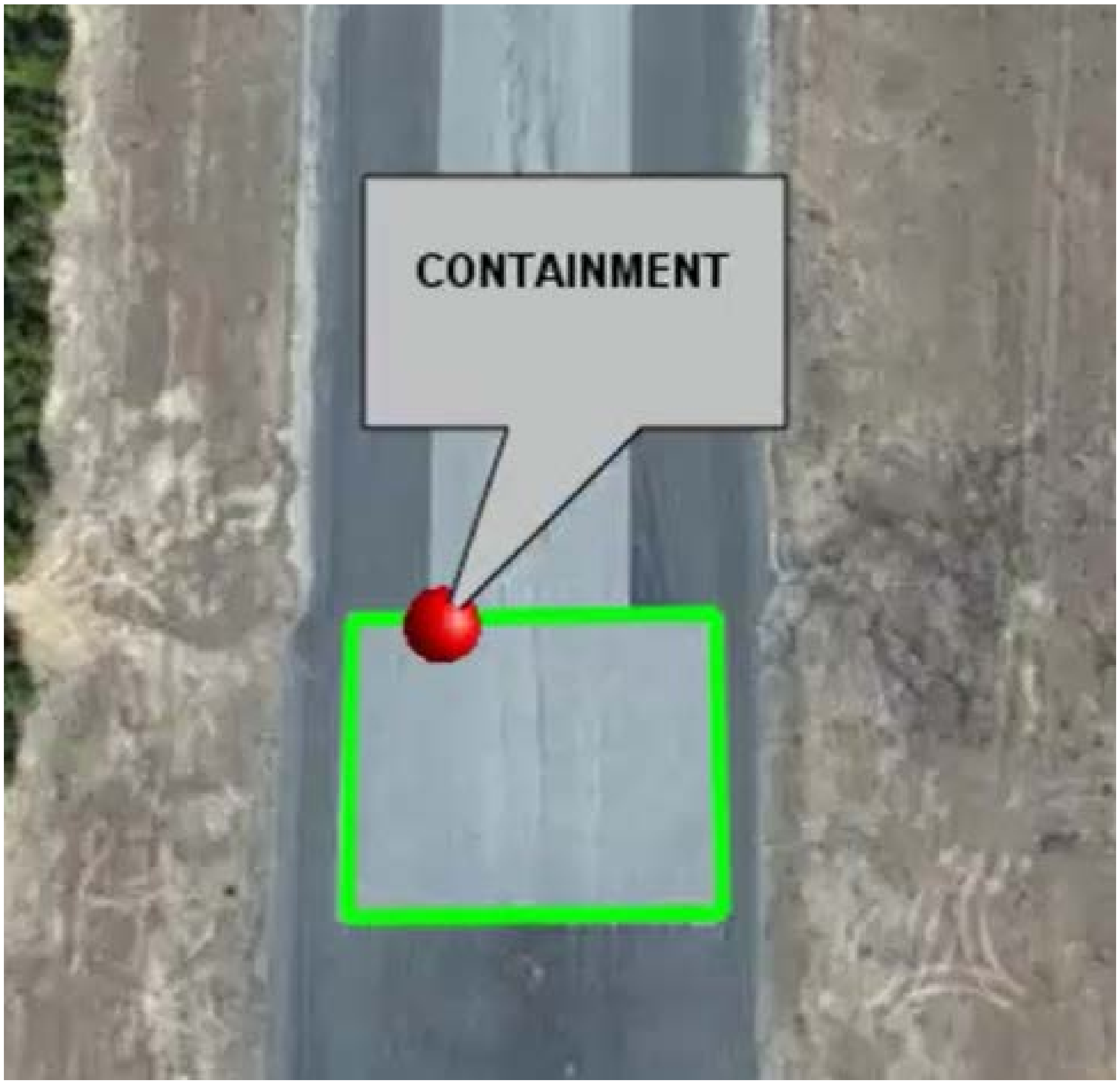
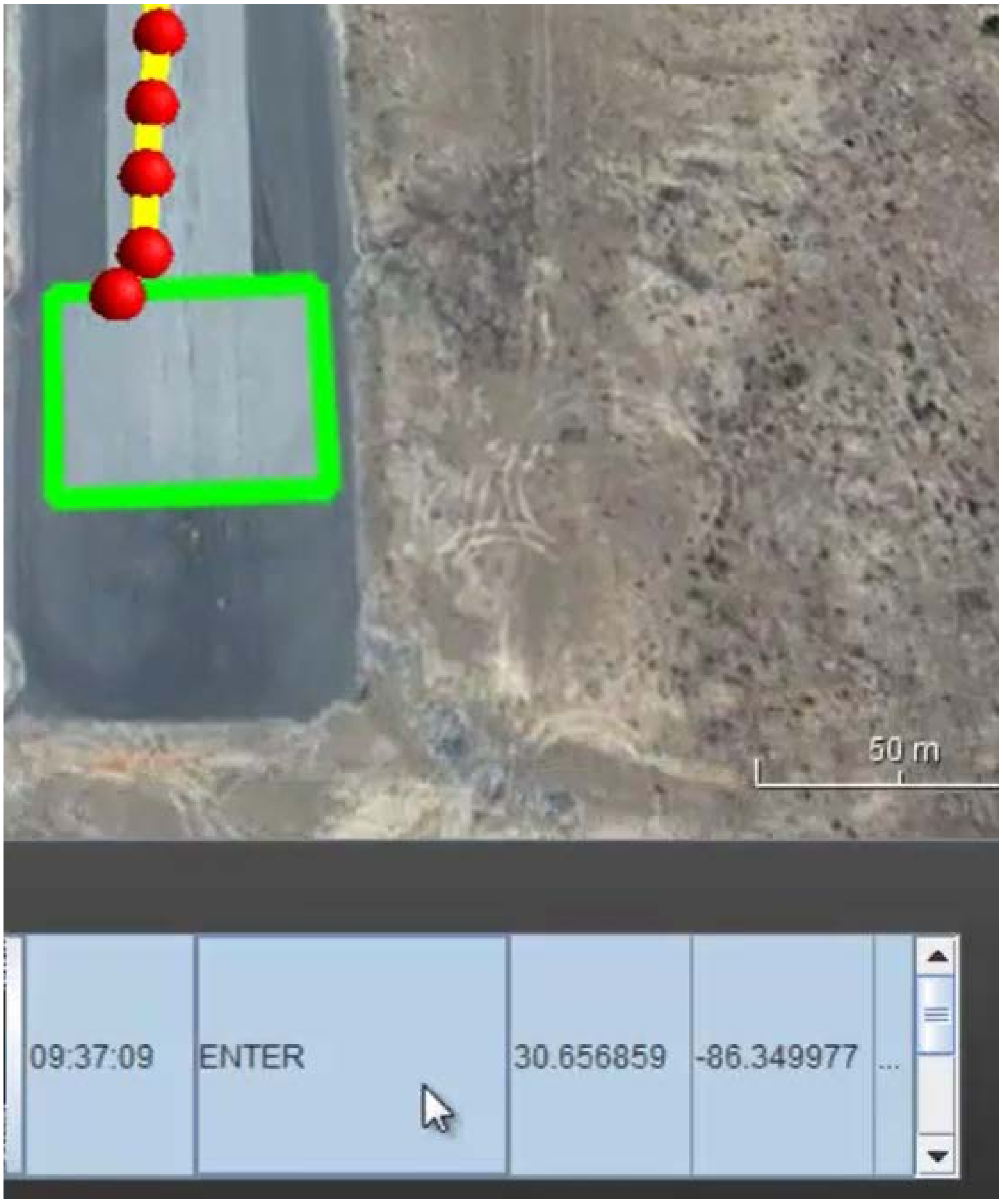


Figure 106. Detected Containment Event

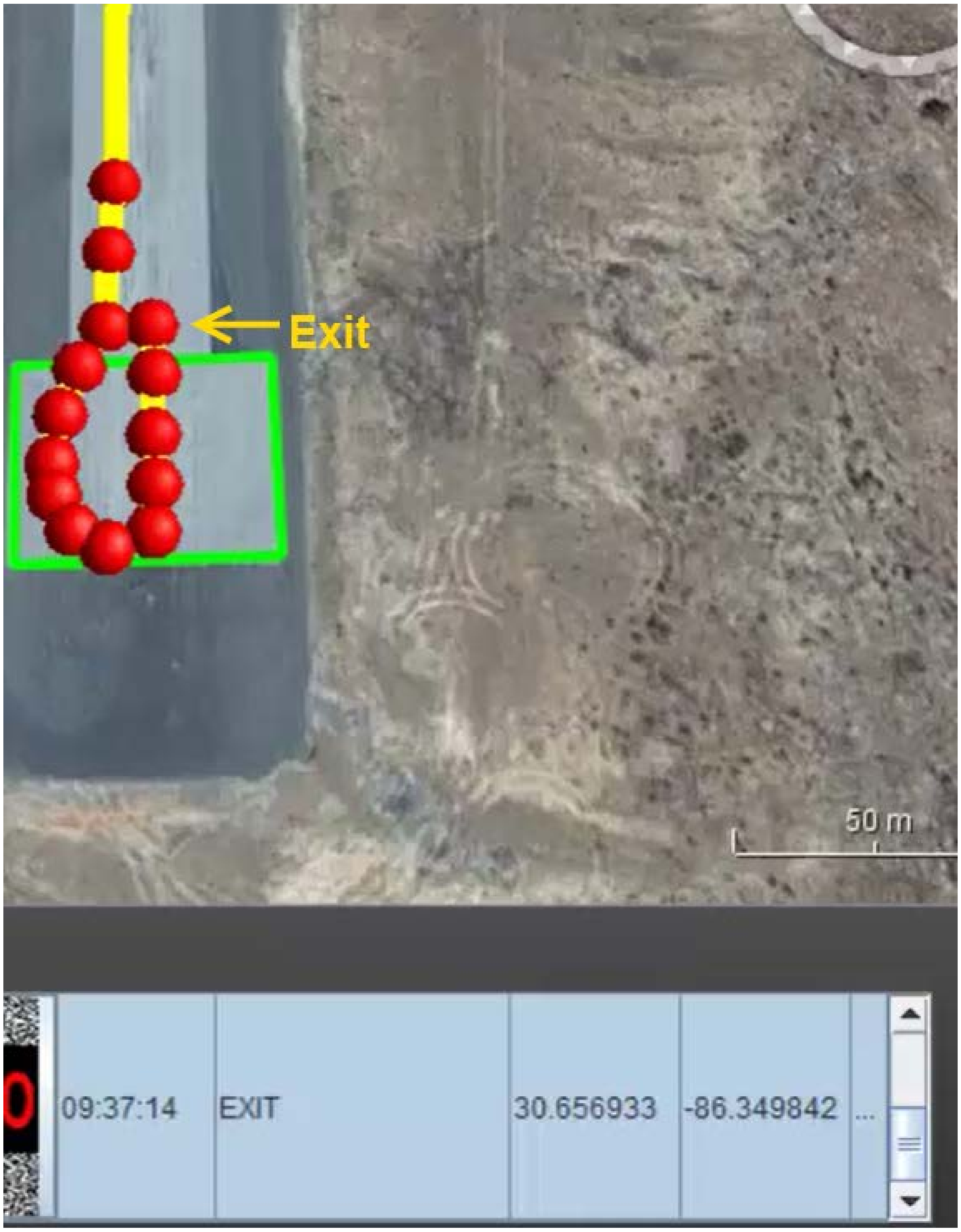
Enter (**Figure 107**) detects when a tracked entity moves into a specified area of interest. While enter is evaluated for each video frame and sensor input, this event only fires when a tracked entity transitions from outside to inside the AOI (Area of Interest).



09:37:09	ENTER	30.656859	-86.349977	...

Figure 107. Detected Enter Event

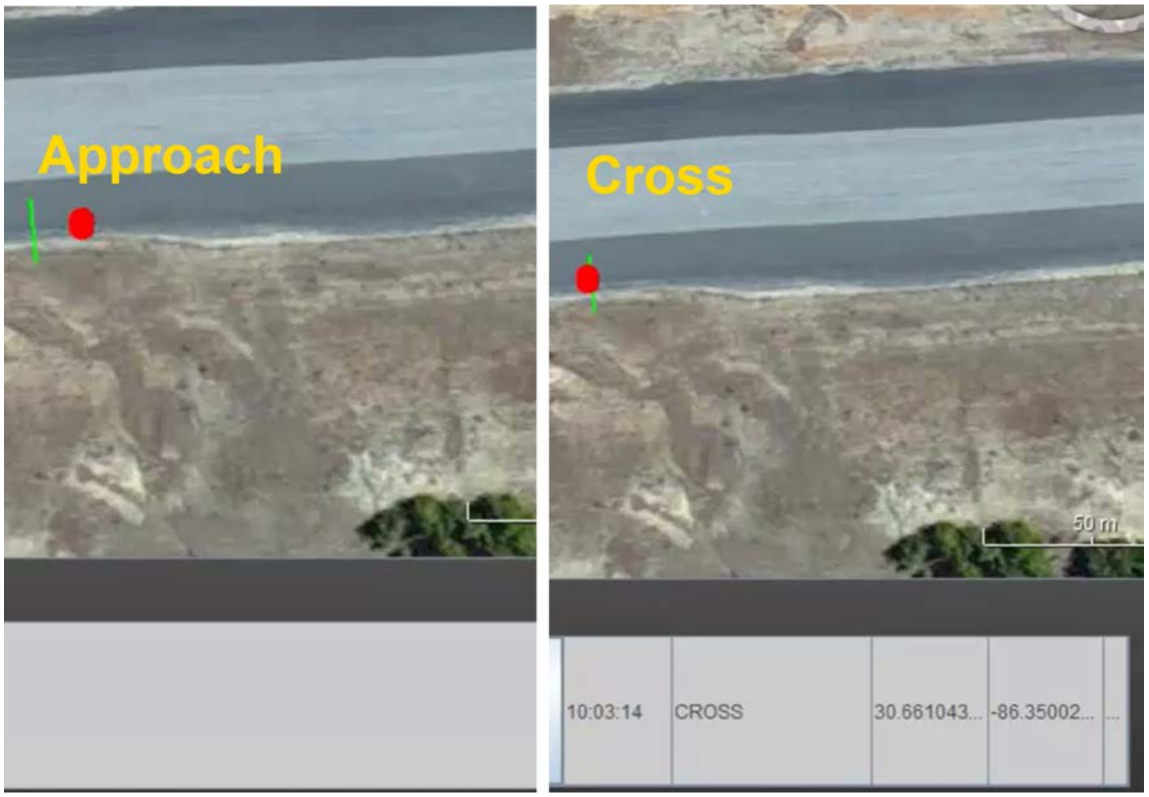
Exit (**Figure 108**) detects when a tracked entity moves out of a specified area of interest. While exit is evaluated for each video frame and sensor input, this event only fires when a tracked entity transitions from inside to outside the AOI.



09:37:14	EXIT	30.656933	-86.349842	...

Figure 108. Detected Exit Event

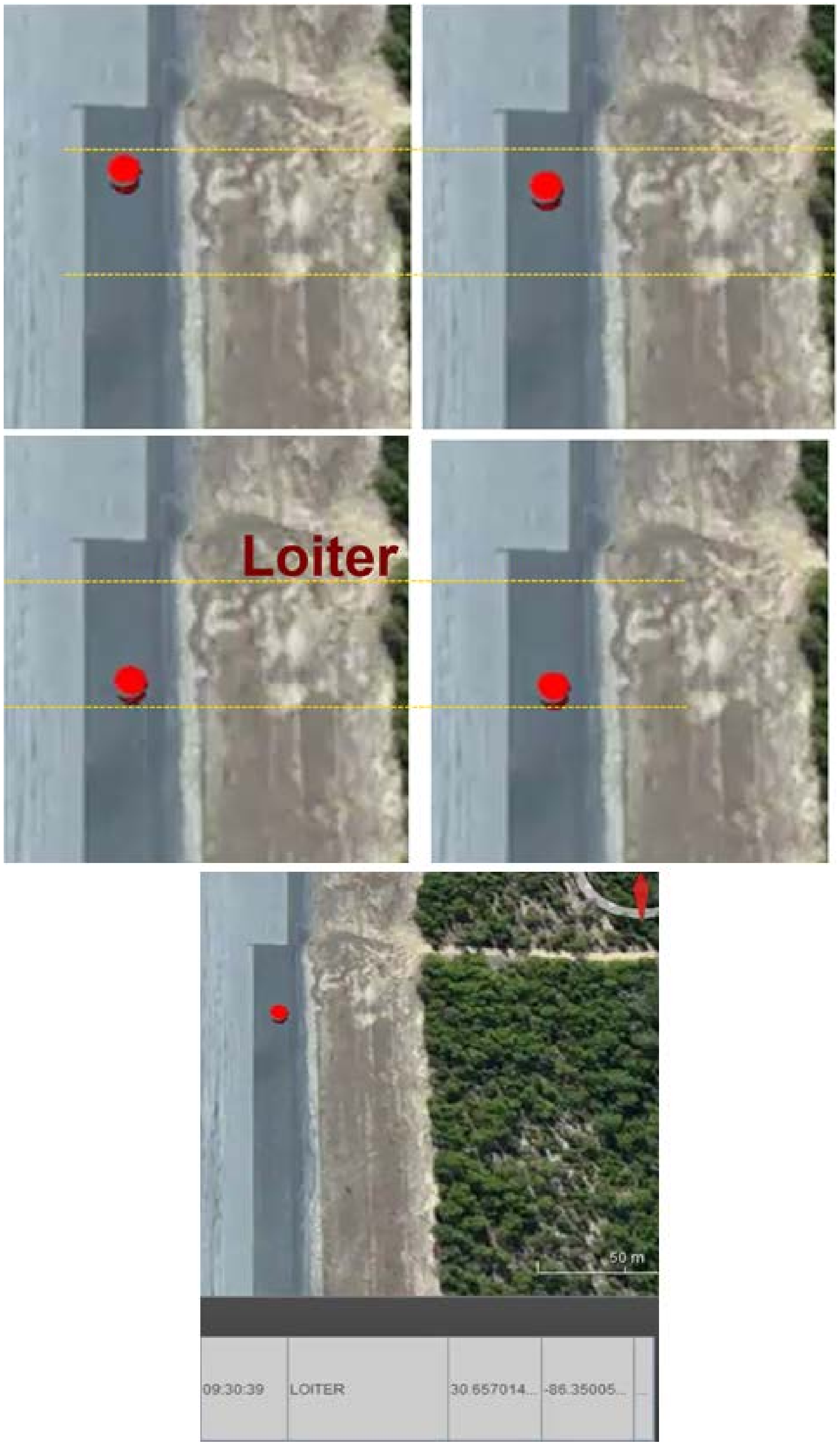
Cross (**Figure 109**) detects when a tracked entity moves across a specified line or boundary. Cross is evaluated for each video frame and sensor input, but only fires when a tracked entity transitions from one side of a line to the other.



10:03:14	CROSS	30.661043...	-86.35002...	...

Figure 109. Detected Cross Event

Loiter (**Figure 110**) detects limited movement of a tracked entity over a timeframe. A loiter event could be a stopped vehicle, a vehicle circling a block, an unusually slow moving vehicle. Because of speed and distance can vary for different tracked entities, Loiter is configurable for different modes of transportation. For example, a person exhibits different characteristics for Loitering than a car or a person on a bicycle.

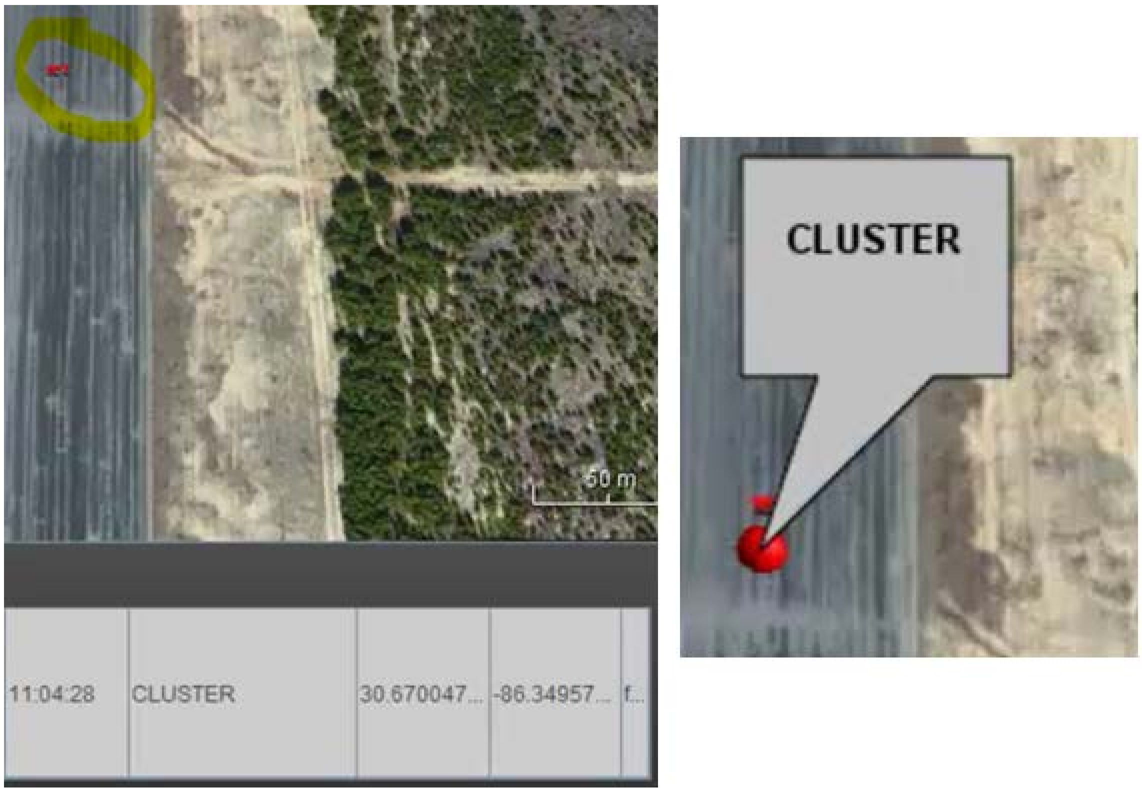


09:30:39	LOITER	30.657014...	-86.35005...	...

Figure 110. Detected Loiter Event

Cluster (**Figure 111**) detects when two or more tracked entities come within proximity to form a "group". In the figure: (Left) Entities moving in a cluster. (Right) The reported cluster event in the client. Cluster is different from simple proximity in that all objects do not need to be within a distance of each other. Rather, all objects in the cluster form a chained relationship where there is minimal distance, and density of entities, in a given geography (**Figure 112**). For example, Cluster could detect a convoy where Proximity would only detect a few vehicles within a distance of a reporting entity location.

Tentacle uses a clustering algorithm called Density-Based Spatial Clustering of Applications with Noise (DBSCAN). This takes as inputs the minimum cluster size (number of points) and a distance threshold for determining when points are close enough to be part of the same cluster.



11:04:28	CLUSTER	30.670047...	-86.34957...	f...

Figure 111. Detected Cluster Event

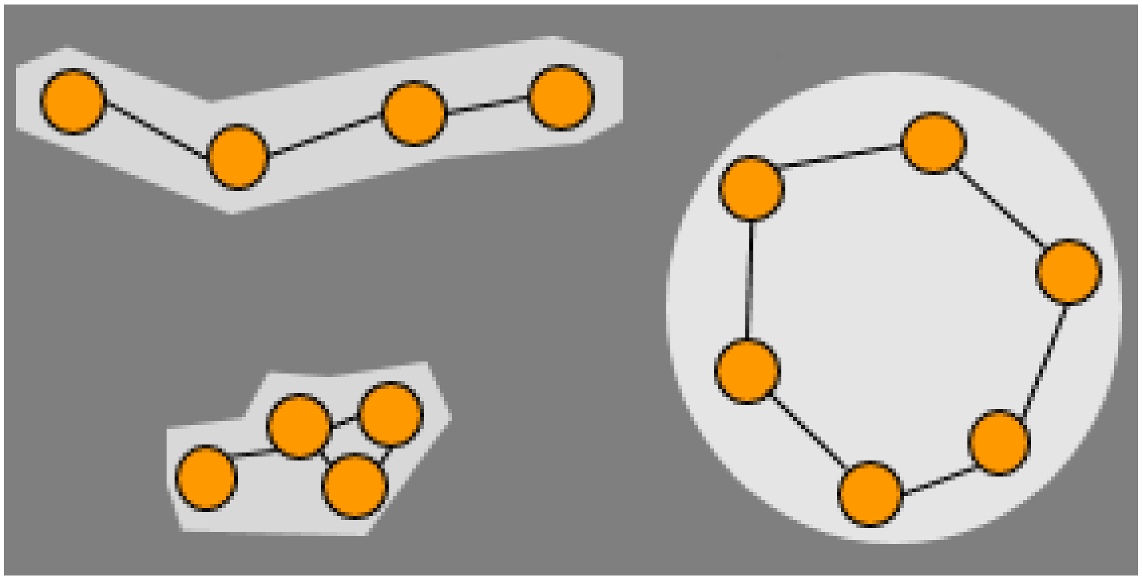


Figure 112. Conceptual Representation of Valid Cluster Events

In this example, an archived search for cluster events is performed. In this case three tracks were involved in the displayed cluster. For parameters, the distance threshold was 10m and the minimum clusters size was 5 points.

4.8 Support Large-Scale Tentacle Server Deployment

One of the goals of this effort was to improve on the Phase I system to make it more scalable. To accomplish this, Primordial developed a scheme for rebroadcasting video streams to clients to reduce load on individual camera sources. This would allow an infinite number of clients to view the video as long as enough layers of repeaters were used between the camera source and the client feed. The approach works by ingesting the source video stream into a set of nodes which then use a third party video application to rebroadcast the stream which can be ingested by other repeaters or by an end-user's client application. This can be repeated as often as needed to service more clients. Although this scheme was not used by the final demonstration setup, it was documented to allow for later use as needed.

Because the intent was for the Tentacle system to handle large quantities of data, the team tested all of its algorithms under heavy load. To do this, Primordial developed an in-house application to generate large numbers of entities to send to the server at once and sustain this rate. Additionally, the team modified this application to read real-world data from the Flight24 flight tracking website (15) to examine how real-world entity reports might behave within the system in very large quantities (usually around 8000 targets per position update). The system was tested numerous times using this setup, the networking code and the algorithms were all adjusted based on its feedback. To support faster processing of data, the Nesper library (described previously) was integrated and was later replaced by a similar in-house library. These libraries were optimized for faster data processing and allowed the system to handle more data more quickly, thus making the system more scalable as a whole.

In regard to the database query system (and the subsequent live alert system), originally it was a multicast system where all results were sent to all clients. Obviously this would not work in a scaled system, where so many requests would be impossible to transmit effectively. Primordial modified the system to register clients independently and transmit only their own requests back to them on an individual basis.

To verify that multiple physical clients connecting to a single server would not cause problems, Primordial tested that scenario in-house on several occasions. The system performed as expected every time.

In order to support flexibility and configurability, something essential if the system was used on a large scale basis, Primordial modified the client application to read settings from an external file rather than being hardcoded as in the Phase I system. This allowed the user to configure the server and camera feeds as desired, if the local Tentacle server changed or if they desired to view a different camera feed than what had been initially set up for them.
Finally, Primordial tested and improved the networking code to provide better support for high-volume network packet processing, robustness to lost connections and alerting the user to connectivity issues. This makes the system more resistant to issues it is likely to encounter in a large scale deployment scenario.

4.9 Improve Tentacle Client GUI

Primordial made significant changes to the GUI developed during Phase I. They updated the video display to give a description of the entities within detected bounding boxes. They also fixed the display of entities on the WorldWind map to be more accurately geolocated.
In addition to these cosmetic changes, Primordial added functionality for archived (and later, live) alerts. A dialogue menu was added to allow the user to configure the queries, including passing in geographic shapes in KML form. At one stage, the ability to provide complex Boolean queries in a tree form was added, but was later removed. This was due to the fact that such queries were very difficult to maintain, verify and test. The interface was simplified to allow individual queries. It was then modified to combine live and archived alerts into one "subscription" interface. This made the GUI much simpler and easier to use.

To keep the user more informed about the client status, diagnostic messages were added to the client to notify the user of connection problems with the server, the video feeds and to display the status of alerts. The final interface is shown in **Figure 113**.

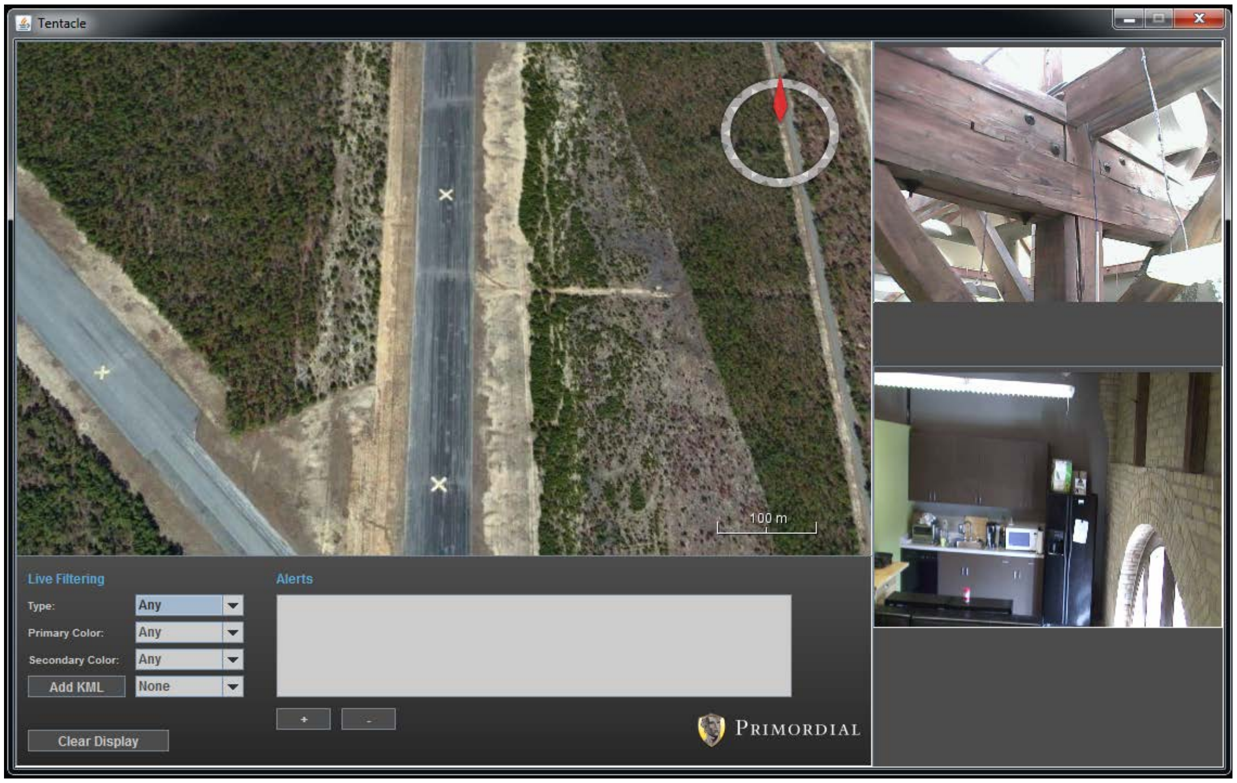
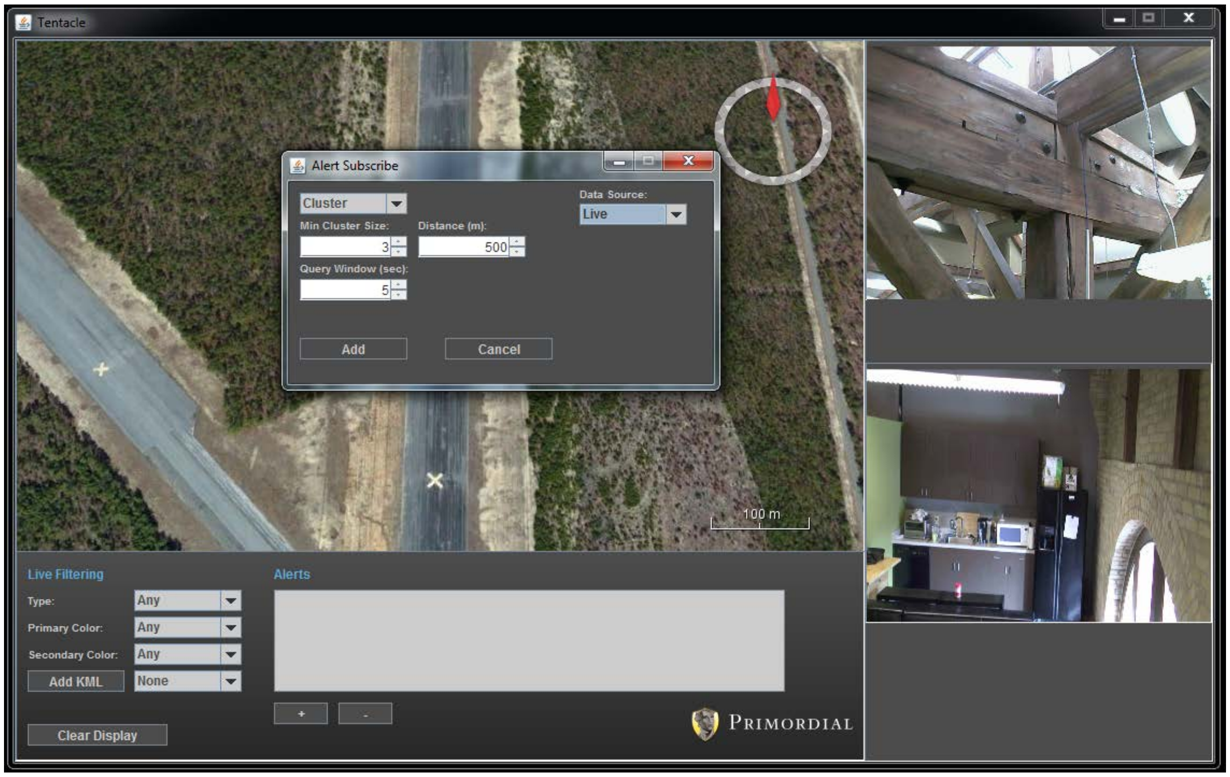


Figure 113. The Final Version of the Tentacle Client Interface

Figure 114 shows the final version of the alert subscription interface.


Alert Subscribe

Cluster

Min Cluster Size: 3

Distance (m): 500

Query Window (sec): 5

Data Source: Live

Live Filtering

Type: Any

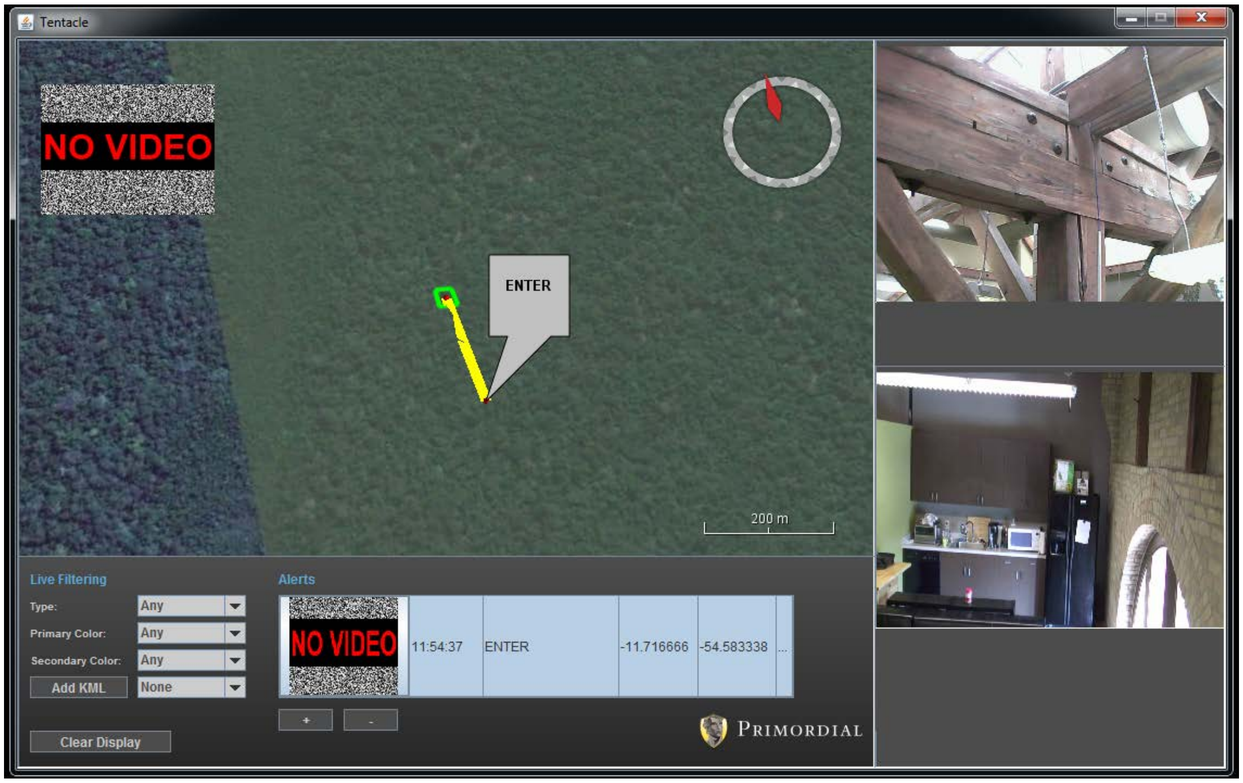
Primary Color: Any

Secondary Color: Any

Add KML: None

Figure 114. The Final Tentacle Client GUI Displaying the Alert-Subscribe Dialogue with Combined Archived/Live Data Functionality

Figure 115 shows the behavior of the system when an alert is being fired. In that figure, an entity entered a specified region and the user was notified in the alert notification table (center).



Type: Any
Primary Color: Any
Secondary Color: Any
None

11:54:37	ENTER	-11.716666	-54.583338	...

Figure 115. Example of the Completed Tentacle Client GUI Displaying a Live Event

4.10 Test, Evaluate, Improve, and Demonstrate

Primordial focused on testing throughout the entire development process.
During the testing process, numerous bugs were fixed, including several major ones. One serious bug was due to the way video frames were handled in memory; the application was not releasing the processed frames and would crash the system after a period of time. Testing and debugging by the Primordial team succeeded in identifying these issues and making the system more reliable.

Primordial and the team specifically focused on improving the system to behave in a more real-time manner. This involved testing and improving the video processing time and event processing time, as well as the database processing time.

Aside from speed and timing requirements, the team focused improving the accuracy of the system by performing numerous tests and adjustments to the entity tracking and geolocation code. The team tested and resolved issues in the static camera geolocation code; they also adapted UAV geolocation code over time by repeated testing and improvement cycles. Additionally, the team tested the system under higher loads. In Phase I, the system was merely a prototype and as such, had only processed small amounts of data. To support scalability in Phase II, the team tested the system under large numbers of reported entity points (8000+) which elicited several bugs and allowed the team to fix some important weaknesses. The team also tested the server using multiple clients and the system performed successfully.

The networking code was another major focus for testing during Phase II. Primordial fixed numerous bugs from the Phase I prototype version of the system and made it much more robust to large packet sizes, large volumes of data and connection loss.

To automate some of the testing process, Primordial integrated JUnit, a third party Java unit testing library, into the system for the client code. For the server code, Primordial added NUnit, the C# equivalent of JUnit.

At the end of the project, the team did a thorough end-to-end evaluation of the system and the source code to verify functionality, check for bugs and to make sure that later developers would be able to modify the code more easily.

During the effort, Primordial gave several demonstrations in the form of video clips to AFRL and partners. At the end of the effort, Primordial demonstrated the final version of the Tentacle system at AFRL with partners joining via conference call.

5.0 CONCLUSIONS AND RECOMMENDATIONS

5.1 Conclusions

Tentacle is capable of identifying and tracking entities in real-time video with operator assistance. The current algorithm cannot reliably detect entities in an automated way and often requires user assistance to start a track. Similarly, if an object moves out of frame, or goes under a bridge, the SDK does not always reacquire the track, thus requiring additional human intervention. However, once tracking is underway, the Tentacle flow produces valuable outputs in terms of entity classification and behavior tracking and recognition. While Tentacle has not achieved reliability for automatic track identification, the tool provides distinct advantages to battlefield operators working in human-in-the-loop contexts. In particular, Tentacle scales work capacity by allowing operators to offload tracking and pattern detection with large data sets that can easily overwhelm operations.

5.2 Recommendations

Although Tentacle shows promise in assisting operators scale workload and perform deeper intelligence on real-time and archived video, there are some improvements required to achieve the original intent of this effort. A summary of these items is provide below.

5.2.1 Scaling Video Processing Capability.

One problem facing real-time video processing is the network and server environment required to perform high-speed processing across a federation of incoming data feeds. While the Tentacle algorithms perform well against these data sets. The process required to perform entity extraction, tracking and classification is significant. This is an item that needs attention for reliably fielding Tentacle into real-world operational contexts as part of a commercial transition phase.

5.2.2 Support Open Architecture for Metadata.

The Tentacle architecture is designed to interact with open standards and DoD standards, such as Cursor on Target (CoT). However, the current implementation does not support this format. Adding CoT support would open options for alternative clients beyond the Tentacle client interface, for receiving alerts and notifications.

5.2.3 Improve Georeferencing from Sensors.

During test and evaluation phases, we continually experienced issues with low quality video resolution and missing or inaccurate georeferencing metadata. As video capture improves and georeferencing calibration improves, Tentacle will be better positioned to provide consistent and valuable output. However, in the absence of good input data, manual intervention and enhancement of the data is required.

5.2.4 Support Using Custom Imagery in Client.

NASA WorldWind's default aerial imagery for projection on its virtual globe is of low-resolution. There are built-in options for slightly better imagery, provided by Microsoft Bing; however even this is of poor quality compared to most military data available. Additional basemaps could be supported using open standards like Open Geospatial Consortium (OGC) Web Map Service (WMS).

5.2.5 Additional Use of Geo-Referenced Queries.

Combining Tentacle event detection with multi-sourced input from other sensors or map data provides a powerful framework for performing context-aware alerting. For example, detecting a loiter event may be of interest. However, determining if the loiter is next to a military supply route verses in the middle of the desert delivers significantly enhanced intelligence. Furthermore, incorporating feeds such as Blue Force Tracking (BFT) or Tactical Ground Reporting System (TIGR) would further inform operators if the loiter is friendly or hostile.

5.2.6 Support Additional Sensor Types.

Currently Tentacle only supports consumption of data from camera feeds; this will be expanded to cover other sensor types being used in the battlefield. Additional sensors might be: friendly C2 systems such as FBCB2 (Force XXI Battle Command Brigade and Below) and MTS (Movement Tracking System), ground vibration sensors, night and thermal imaging cameras, motion detection or audio sensors. Tentacle leverages the same architecture to process multiple video feeds to process and fuse multiple sensor feeds with video.

5.2.7 SRI Terrasight Integration.

The Tentacle team has a letter of support from SRI/Terrasight and there is interest in fielding Tentacle into programs of record. SRI has a proven track record for commercialization and Primordial feels they would be a good partner for Phase III or commercialization efforts.

6.0 REFERENCES

1. AFRL. *Air Force SBIR Topic "AF103-032 Multi-camera real-time Feature Recognition, Extraction, and Tagging Automation.*

2. Panoptic. C-Thru. [Online] http://panopticsystems.com.

3. IntuVision. Panoptes User Manual. [Online] 2010. http://www.IntuVisiontech.com/cutsheets/PanoptesUserBrief.pdf.

4. Video Recall User Manual. [Online] 2010. http://www.IntuVisiontech.com/cutsheets/VideoRecall%20Userbrief.pdf.

5. *COCOA - Tracking in Aerial Imagery.* Shah, Mubarak and Ali, Said. s.l. : SPIE (International Society for Optical Engineering) Airborne Intelligence, Surveillance, Reconnaissance (ISR) Systems and Applications, 2006.

6. Primordial. Demonstration Video of UAV Tracking 1. [Online] http://youtu.be/jHwxRUkI9YU.

7. Demonstration Video of UAV Tracking 2. [Online] http://youtu.be/z4VU3EpRJtY.

8. PAR Systems. PAR Government GV 3.0 Software. [Online] http://www.pargovernment.com/topic_details.asp?key=71.

9. NATO. STANAG 4559. [Online] http://www.nato.int/structur/ac/224/standard/4559/4559.htm.

10. Primordial. Demonstration Video of Archive Search Functionality 1. [Online] http://www.youtube.com/watch?v=VBNV4_5j_wk.

11. Demonstration Video of Proximity 1. [Online] http://youtu.be/UfNVRdjivgw.

12. Demonstration Video of Containment/Count Query Implementation 1. [Online] http://www.youtube.com/watch?v=fz7JdLSFrCs&feature=youtu.be.

13. Demonstration Video of Enter/Exit Queries 1. [Online] http://www.youtube.com/watch?v=x-nG6-YiOTQ.

14. Demonstration Video of Loitering 1. [Online] http://www.youtube.com/watch?v=niRSyYPWp6g&feature=youtu.be.

15. Demonstration Video of Boolean Combined Queries 1. [Online] http://www.youtube.com/watch?v=vWpsz0vKbSI&feature=youtu.be.

16. Demonstration Video of Cluster Query 1. [Online] http://www.youtube.com/watch?v=iRc44QhGyq4.

17. Nesper Home. [Online] http://docs.codehaus.org/display/ESPER/Home.

18. Airplane location feed. [Online] http://www.flightradar24.com/.

19. Quad trees background. [Online] http://en.wikipedia.org/wiki/Quadtree.

20. Haversine formula. [Online] http://en.wikipedia.org/wiki/Haversine_formula.

21. R-Trees. [Online] http://en.wikipedia.org/wiki/R-tree.

22. SRI. TerraSight. [Online] http://www.sri.com/engage/products-solutions/terrasight-software.

APPENDIX A – REVIEW OF TEST VIDEO

A.1 Small Datasets (on CD/DVD):

Backhoe data sample and visual-D challenge problem - like a 3D radar-based point cloud, 3D image of a tractor

Challenge problem SAR (Synthetic Aperture Radar) based GMTI in urban environments - radar spectrum data, not recognizable as objects (very different data that what we're used to using)

Gotcha Large Scene Data Example - Large area, SAR data, blurry shapes, only in Matlab data files

Gotcha Volumetric SAR dataset - xband data of overhead scene with various civilian vehicles and "calibration targets". All files are in Matlab format. Images of the data show fairly blurry examples of targets.

Wide Angle SAR Target Discrimination Challenge Problem - radar data samples representing different types of vehicles

CCD (Charge Coupled Device) challenge problem SPIE 2010 - Large area Xband SAR data, includes buildings and vehicles and trees, mostly matlab files, but some tifs of the general area (not big)

Greene 07 dataset – Looks like either a single huge image or a set of frames. The description indicates it's a single high-res "google earth" style image. Files are in NITF (National Imagery Transmission Format) format and there's a Derby Index database along of the files on the discs

Civilian vehicle data domes set - contains Xband scatter data for a set of civilian vehicle types in matlab format

MSTAR (Moving and Stationary Target Acquisition and Recognition) public targets - 1ft Xband scatter data for specific targets (tank, personnel carrier)

MSTAR predict lite software - "allows the use of scattering center models in the generation of grayscale signatures"

MSTAR public clutter - a set of "clutter" images from 1ft xband radar of 100 scenes. Images are in a proprietary format with embedded sensor info, although they give a link to a webpage to download tools for manipulating the data

MSTAR public mixed targets - 1ft STARLOS (SAR Target Recognition and Location System) xband sensor of various target types. Files are in proprietary format with embedded sensor info. JPG (Joint Photo Experts Group file format) preview files. Difficult to identify targets with the naked eye.

MSTAR public T-72 variants - 1ft STARLOS xband sensor of variants of T-72 tank. Files are in proprietary format with embedded sensor info. JPG preview files Difficult to identify targets with the naked eye.

A.2 Large Datasets (on Hard Drive):

COLUMBUS LARGE IMAGE FORMAT (CLIF) 2006 DATASET - Huge amount (245GB) of high res (~4000x2600px) 2fps video frames in RAW format (with previews) and accompanying TXT (Text) files with telemetry data. Covers multiple areas of a college campus, has people and vehicles moving.

COLUMBUS LARGE IMAGE FORMAT (CLIF) 2007 DATASET - Huge amount (966GB) of high res (~4000x2600px) 2fps video frames in RAW format (no previews) and accompanying TXT files with telemetry data. Has video frames from building mounts and aerial platforms of a college campus.

COLUMBUS SURROGATE UNMANNED AERIAL VEHICLE (CSUAV) DATASET - Has both EO (83GB) and FLIR (Forward Looking Infrared) (27GB) data. EO is 5fps, IR is 30fps. EO resolution is large (~4000x2600px), IR is small (~640x512px). Proprietary formats for both types. Has embedded sensor/telemetry data, along with a text file with the same data for each of the data files. Documentation indicates this includes some sample AVI (Audio Video Interleave) files as well but none were found in the directory on the hard drive. Readme has tips for opening/reading the files. Covers multiple straight-line runs over a large area.

MSTAR - Lots of data files. Look to be the same as those on the discs for the various MSTAR packages. Lots of images of various targets (tanks, etc) from different angles. Could be useful for training a classifier.

WPAFB-21Oct2009 - The large (254GB) dataset contains EO data from an aerial platform and includes the raw camera data, jpegs, and NITF Rset files of a flyover near Wright Patterson AFB. Has separate "pos" files with telemetry data (note: also seems to have some extra information in excel files). The EO files are in a resolution of 4872x3248px in 8bit grayscale.

VIVID datasets 3 and 4 – Excellent data set for Tentacle. Lots of the VIVID videos, in multiple sensor formats (EO, IR, etc). AVI video files, 30fps for all afaik. EO is 640x480px, IR is 320x256px Each have a metadata xml file accompanying it. Lots more scenes than what we originally had for VIVID.

A.3 Evaluation of the VIVID Dataset for Target Types

Because the VIVID dataset seemed the most promising, Primordial pushed further to identify the types of targets for each of the VIVID videos in the set to facilitate later testing selections. The evaluation created from that effort follows. A dash between the final video clip numbers indicates a range of videos, i.e. 001-005 indicates videos 001, 002, etc. 569 video files in total.

A.3.1 Scenes

-d4-apr20p1-1_1 Pass1 Clip_039-045
-d4-apr20p2-1_2 Pass2 Clip_001-005
-d4-apr20p3-4_6 Pass3 Clip_051
-d4-apr20p3-5_6 Pass3 Clip_052-054
-d4-apr20p3-6_6 Pass3 Clip_055-057

A.3.2 Vehicles

-d4-apr20p1-1_1 Pass1 Clip_046-050
-d4-apr20p2-1_2 Pass2 Clip_010-012
-d4-apr20p2-2_2 Pass2 Clip_013-019
-d4-apr20p3-1_6 Pass3 Clip_002-019
-d4-apr20p3-2_6 Pass3 Clip_020-030
-d4-apr20p3-3_6 Pass3 Clip_031-041
-d4-apr20p3-4_6 Pass3 Clip_042-050
-d4-apr20p4-1_3 Pass4 Clip_019-021
-d4-apr20p4-2_3 Pass4 Clip_022-025
-d4-apr20p4-3_3 Pass4 Clip_026-031
-d4-apr21p1-1_9 Pass1 Clip_003-006
-d4-apr21p1-2_9 Pass1 Clip_007-016
-d4-apr21p1-3_9 Pass1 Clip_017-021
-d4-apr21p1-4_9 Pass1 Clip_022-024
-d4-apr21p1-5_9 Pass1 Clip_025-028
-d4-apr21p1-6_9 Pass1 Clip_029-041
-d4-apr21p1-7_9 Pass1 Clip_045-052
-d4-apr21p1-8_9 Pass1 Clip_053-055
-d4-apr21p1-9_9 Pass1 Clip_057-061
-VIVID_DS3-D1-1_3\Day1Scenario1-1\Clip_010
-VIVID_DS3-D1-1_3\Day1Scenario1-2\Clip_004-015
-VIVID_DS3-D1-1_3\Day1Scenario9\Clip_006-008
-VIVID_DS3-D1-2_3\Day1Scenario4_5\Clip_004-008
-VIVID_DS3-D1-2_3\Day1Scenario6\Clip_003-009
-VIVID_DS3-D1-3_3\Day1Scenario7-1\Clip_008-010
-VIVID_DS3-D1-3_3\Day1Scenario7-2\Clip_002-004
-VIVID_DS3-D1-3_3\Day1Scenario8\Clip_013-018
-VIVID_DS3-D1-3_3\Day1Scenario10\Clip_009-011
-VIVID_DS3-D2-1_3\Day2Scenario1\Clip_001-014
-VIVID_DS3-D2-2_3\Day2Scenario1\Clip_015-027
-VIVID_DS3-D2-3_3\Day2Scenario4_5\Clip_009-016
-VIVID_DS3-D2-3_3\Day2Scenario6\Clip_010-012

A.3.3 People

-d4-apr20p3-1_6 Pass3 Clip_002-003

-d4-apr20p4-1_3 Pass4 Clip_021
-d4-apr21p1-1_9 Pass1 Clip_004-006
-d4-apr21p1-2_9 Pass1 Clip_007-016
-d4-apr21p1-3_9 Pass1 Clip_017
-d4-apr21p1-5_9 Pass1 Clip_025
-d4-apr21p1-6_9 Pass1 Clip_029
-d4-apr21p1-6_9 Pass1 Clip_041
-d4-apr21p1-7_9 Pass1 Clip_045-051
-d4-apr21p1-9_9 Pass1 Clip_061
-VIVID_DS3-D1-1_3\Day1Scenario1-1\Clip_010-012
-VIVID_DS3-D1-3_3\Day1Scenario7-1\Clip_008-010
-VIVID_DS3-D1-3_3\Day1Scenario7-2\Clip_002-004
-VIVID_DS3-D1-3_3\Day1Scenario8\Clip_013
-VIVID_DS3-D2-1_3\Day2Scenario1\Clip_001-007
-VIVID_DS3-D2-1_3\Day2Scenario1\Clip_011-013

A.3.4 Unrecognizable

-d4-apr20p3-1_6 Pass3 Clip_001

APPENDIX B – PRIMER ON BIOLOGICALLY INSPIRED MODELING (BIM)

B.1 Training Phase

1. **S1:** Apply a bank of Gabor filters to the input image. The filters come in 4 orientations θ and 16 scales s (see Table 1). Obtain 16×4 = 64 maps $(S1)_{s\ \theta}$ that are arranged in 8 *bands* (*e.g.*, band 1 contains filter outputs of size 7 and 9, in all four orientations, band 2 contains filter outputs of size 11 and 13, *etc*).
2. **C1:** For each *band*, take the max over scales and positions: each band member is sub-sampled by taking the max over a grid with cells of size $N\Sigma$ first and the max between the two scale members second, *e.g.*, for band 1, a spatial max is taken over an 8×8 grid first and then across the two scales (size 7 and 9). Note that we do not take a max over different orientations, hence, each band $(C1)_{\Sigma}$ contains 4 maps.
3. **During training only:** Extract K patches Pi=1,...K of various sizes $ni \times ni$ and all four orientations (thus containing $ni \times ni \times 4$ elements) at random from the $(C1)\Sigma$ maps from all training images.

B.2 Testing Phase

1.Compute S1 and C1

2.**S2:** For each C1 image $(C1)\Sigma$, use **Training Patches P to** compute: $Y = \exp(-\gamma||X - Pi||_2)$ for all image patches X (at all positions) and each patch P learned during training for each band independently. Obtain S2 maps $(S2)_{\Sigma i}$.

3.**C2:** Compute the max over all positions and scales for each S2 map type $(S2)i$ (*i.e.*, corresponding to a particular patch Pi) and obtain shift- and scale-invariant C2 features
$(C2)i$, for $i = 1. . .K$.
Note that the size of the features depends only on the number of patches extracted during learning and not on the input image size.

APPENDIX C – PROFESSIONAL PERSONNEL ASSOCIATED WITH PROGRAM

C.1 Primordial

Mr. Randy L. Milbert, Founder and President (Principal Investigator)

Education: BS, Electrical Engineering and Computer Science, MIT, 2000
Experience: Mr. Milbert founded Primordial in 2002 and has since secured more than $11M in US Army, USMC, USSOCOM, and USAF contracts. Mr. Milbert invented Ground Guidance in 2002 and has led its development since. He has served as the principal investigator on 12 successful R&D efforts including developing a geographically enabled augmented reality system for dismounted soldiers, real-time route planning software for tactical and logistical vehicles, and visual navigation algorithms for soldiers navigating in GPS-denied environments. Mr. Milbert has three issued and four pending patents for routing soldiers around combat obstacles, differentiating friend from foe in a head-mounted display, and providing turn-by-turn guidance during off-road navigation. In the last 15 years, he has helped to found three companies, win the MIT $50K Entrepreneurship Competition, and raise $6.5M in venture capital.
Mr. Steve Hersman, Vice President of Marketing
Education: BA, Business Administration, University of St. Thomas, 1988; Master of International Management, University of St. Thomas, 1994
Experience: For the last 13 years, Mr. Hersman has served in executive sales and marketing positions. The last 9 years have been at defense firms. Within the US Army, he has interfaced extensively with CECOM; PEO STRI; and TACOM. He has also secured defense contracts in Australia, Bahrain, India, Indonesia, Japan, Mexico, United Arab Emirates, United Kingdom, and Singapore. Prior to joining Primordial two years ago, Mr. Hersman was vice president of sales at Meggitt Defense Systems Caswell, director of international sales at Meggitt Training Systems, and director of carrier sales for Nexcom Telecommunications. At Nexcom, he expanded the business from 3 to 12 countries in 3 years, increased revenue 10X, and helped to sell the company for 30X its venture capital investments.

Mr. Benjamin L. Post, SME

Education: BS, Computer Science, University of Minnesota Duluth, 2001
Experience: Mr. Post served as a sergeant in the US Army. He entered the Army in January 2002. He attended basic combat training at Fort Jackson, advanced individual training at Fort Huachuca, and Arabic language training at at Presidio of Monterey. He deployed twice in support of the GWOT, once to Iraq where he conducted interrogations in Abu Ghraib and other locations. The second deployment was to Kuwait, where he led a small team that conducted intelligence collection activities. During his Army career, Mr. Post served as an interrogator, debriefer, team leader, and an observer/controller for field training exercises. He was assigned to the 202d military intelligence battalion in Fort Gordon from April 2004 to January 2007. His awards include the joint service achievement medal, combat action badge, and two GWOT expeditionary medals.

C.2 AHM/Panoptic

Mr. Philippe Van Nedervelde, CTO

Education: MS, Communications, Katholieke Universiteit Leuven; Post Graduate Degree in Media and Information Science, Katholieke Universiteit Leuven
Experience: Mr. Van Nedervelde is an award-winning entrepreneur-producer-designer in single- and multi-user online VR. As one of Europe's earliest VR practitioners, he has nearly two decades of experience in the design, sales, and production of world-class online VR projects for entertainment and business applications. Mr. Van Nedervelde's company, E-Spaces, has completed more than 50 online VR projects. Mr. Van Nedervelde has been an invited speaker at numerous VR conferences worldwide since the mid 1990s and received the prestigious VR World Congress presenter's award. Citizenship: Mr. Van Nedervelde is a Belgian citizen, but has held NATO security clearances and worked for the CIA. If his citizenship prevents him from working on the project, John Caulfield (see below) will replace him as the team's primary Panoptic consultant.

Dr. H. John Caulfield, Director, Scientific Board of Advisors Distinguished

Education: BA, Physics, Rice, 1958; Doctorate, Physics, Iowa State, 1962
Experience: Dr. Caulfield is a distinguished research professor at the University of Alabama and Fisk University. Mr. Caulfield has received numerous international awards and prizes and has authored over 200 refereed journal papers, 12 books, 13 patents, 26 book chapters, and several popular articles including one read by over 25 million people—a National Geographic cover story on holography. Business Week named him "One of America's 10 Top Scientists", Byte "One of the Most Influential People in the World in Minicomputers", and Fortune "A Pioneer on Optical Processing".

C.3 IntuVision

Dr. Sadiye Guler, Founder and CEO

Education: Doctorate, University of Massachusetts, Amherst
Experience: The founder and CEO of IntuVision, Dr. Guler has over 20 years of R&D and industry experience in computer vision technology and application development as well as successful transition of new technologies into operational products. She is the principal investigator for IntuVision's federally-funded video analysis and content extraction project, which is applying cognitive vision concepts to video tracking and feature extraction. Dr. Guler received Massachusetts High Technology Council's 2008 Women-to-Watch award for developing and commercializing industry-shaping video analysis products. She has published several articles in relevant publications and has pending patents in video content extraction. Prior to founding IntuVision, Dr. Guler served as Northrop Grumman's directing manager and chief scientist for advanced video technology. She led more than $9M in R&D projects and transitioned the resulting technology into a commercially-licensed Video Alert product installed on 300 camera systems.

C.4 Carnegie Mellon University (CMU)

Dr. Sanjiv Singh, Research Professor, Robotics Institute

Education: Doctorate, Robotics, CMU
Experience: Dr. Singh’s research relates to the operation of robots in natural and in some cases, extreme environments. His recent work has two main themes: perception in natural and dynamic environments and multi-agent coordination. He currently leads a program funded by TATRC to fully automate casualty evacuation using a helicopter. In the past, he has led the collision avoidance effort for DARPA’s OAV-II project. This project demonstrated an autonomous RMax that flew close to the ground, avoiding buildings, wires and trees at speeds of up to 8m/sec. He has also been involved in several efforts related to perception for micro air vehicles, including a recent DARPA seedling that developed a fully autonomous, self-contained quadrotor that is capable of flying very close to natural and manmade objects. He has also led projects to develop several sense- and- avoid capabilities for UAVs operating in the National Air Space.

Dr. Ben Grocholsky, Project Scientist, Robotics Institute

Education: Doctorate, Aeronautical Engineering, Sydney University, 2002
Experience: Dr. Grocholsky’s research develops sensor fusion, estimation and control technologies aimed at enhancing the operational capability and integrity of aerial and ground robot systems performing surveillance, mapping and navigation tasks. He has led two Army SBIR project sub-contracts to iRobot on air-ground collaborative ground target search, tracking and geolocation by man-portable robots. Before joining CMU, he led the UAV research effort on the University of Pennsylvania DARPA MARS 2020 multi-robot situational awareness project and was a researcher on the BAE Systems ANSER project that demonstrated multi-UAV simultaneous localization and mapping.

LIST OF SYMBOLS, ABBREVIATIONS, AND ACRONYMS

2D Two-dimensional
3D Three-dimensional
AF Air Force
AFB Air Force Base
AFRL Air Force Research Laboratory
AHM All Hazards Management
AIS Automatic Identification System
AOI Area of Interest
API Application Programming Interface
AUC Area Under Curve
AVI Audio Video Interleave
BFT Blue Force Tracking
BIM Biologically Inspired Model
CCD Charge Coupled Device
CLIF Columbus Large Image Format
CMU Carnegie Mellon University
COTS Commercial Off The Shelf
CPU Central Processing Unit
CRCV Center for Research in Computer Vision at University of Central Florida
CSUAV Columbus Surrogate Unmanned Aerial Vehicle
CTO Chief Technology Officer
DARPA Defense Advanced Research Projects Agency
DBSCAN Density-Based Spatial Clustering of Applications with Noise
DFARS Defense Federal Acquisition Regulation Supplement
DTIC Defense Technical Information Center
EO ElectroOptical
EPL Event Processing Language
FA False Alarm
FBCB2 Force XXI Battle Command Brigade and Below
FLIR Forward Looking Infrared
FPS Frames Per Second
GB Gigabyte
GMTI Ground Moving Target Indicator
GPU Graphical Processing Unit
GUI Graphical User Interface
GUID Globally Unique Identifier
HD High Definition
HMAX Hierarchical Model and X
HOG Histogram of Oriented Gaussians
HPW Human Performance Wing
ID Identification
IDEA Integrated Design Engineering Analysis
IP Internet Protocol
IPC Inter-Process Communication

IPL	Product Library Implementation
IR	Infrared
ISR	Intelligence Surveillance and Reconnaissance
JPG/JPEG	Joint Photo Experts Group file format
KLV	Key-Length Value format
KML	Keyhole Markup Language
LAB	Lightness and A and B color opponent dimension color space model
LBP	Local Binary Pattern
MCFRETA	Multi-camera real-time Feature Recognition, Extraction, and Tagging Automation
MD	Missed Detection
MHI	Motion History Image
MN	Minnesota
MSR	Main Supply Route
MSTAR	Moving and Stationary Target Acquisition and Recognition
MTS	Movement Tracking System
NASA	National Aeronautics and Space Administration
NIST	National Institute for Standards and Technology
NITF	National Imagery Transmission Format
NSILI	NATO Standard ISR Library Interface
OGC	Open Geospatial Consortium
OH	Ohio
OIRDS	Overhead Imagery Research Data Set
PNG	Portable Network Graphics
QVGA	Quarter-Video Graphics Array Resolution (320x240 pixels)
RAW	Raw image format containing minimally processed data
RGB	Red Green Blue
RH	Human Effectiveness Directorate
RHCV	Battlespace Visualization Branch
ROC	Receiver Operating Characteristic
SAR	Synthetic Aperture Radar
SBIR	Small Business Innovation Research
SDK	Software Development Kit
SDMS	Sensor Data Management System
SIFT	Scale-Invariant Feature Transform
SPIE	International Society for Optical Engineering
SQL	Structured Query Language
STANAG	NATO Standardization Agreement
STARLOS	SAR Target Recognition and Location System
SVM	Support Vector Machine
TIGR	Tactical Ground Reporting System
TIPL	Tentacle Product Library Implementation
TUM	Technical University Munich
TR	Technical Report
TXT	Text
UAV	Unmanned Aerial Vehicle
UCF	University of Central Florida

UGV	Unmanned Ground Vehicle
UI	User Interface
USGS	US Geological Survey
VGA	Video Graphics Array Resolution (640x480 pixels)
VIRAT	Video and Image Retrieval and Analysis Tool
VIVID	Video Verification of Identity
VLC	VideoLan media player software application
VM	Virtual Machine
WAMI	Wide Area Motion Imagery
WMS	Web Map Service
WP	Wright Patterson
XML	Extensible Markup Language