AFRL-RI-RS-TR-2015-215

VIDEO TO TEXT (V2T) IN WIDE AREA MOTION IMAGERY

INTELLIGENT FUSION TECHNOLOGY, INC.

SEPTEMBER 2015

FINAL TECHNICAL REPORT

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED

STINFO COPY

AIR FORCE RESEARCH LABORATORY
INFORMATION DIRECTORATE

■ **AIR FORCE MATERIEL COMMAND** ■ **UNITED STATES AIR FORCE** ■ **ROME, NY 13441**

NOTICE AND SIGNATURE PAGE

Using Government drawings, specifications, or other data included in this document for any purpose other than Government procurement does not in any way obligate the U.S. Government. The fact that the Government formulated or supplied the drawings, specifications, or other data does not license the holder or any other person or corporation; or convey any rights or permission to manufacture, use, or sell any patented invention that may relate to them.

This report was cleared for public release by the 88th ABW, Wright-Patterson AFB Public Affairs Office and is available to the general public, including foreign nationals. Copies may be obtained from the Defense Technical Information Center (DTIC) (http://www.dtic.mil).

AFRL-RI-RS-TR-2015-215 HAS BEEN REVIEWED AND IS APPROVED FOR PUBLICATION IN ACCORDANCE WITH ASSIGNED DISTRIBUTION STATEMENT.

FOR THE DIRECTOR:

/ S /
ERIC BLASCH
Work Unit Manager

/ S /
MICHAEL J. WESSING
Deputy Chief, Information Intelligence
Systems and Analysis Division
Information Directorate

This report is published in the interest of scientific and technical information exchange, and its publication does not constitute the Government's approval or disapproval of its ideas or findings.

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

The public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Department of Defense, Washington Headquarters Services, Directorate for Information Operations and Reports (0704-0188), 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302. Respondents should be aware that notwithstanding any other provision of law, no person shall be subject to any penalty for failing to comply with a collection of information if it does not display a currently valid OMB control number.
PLEASE DO NOT RETURN YOUR FORM TO THE ABOVE ADDRESS.

1. REPORT DATE *(DD-MM-YYYY)* SEPTEMBER 2015	**2. REPORT TYPE** FINAL TECHNICAL REPORT	**3. DATES COVERED** *(From - To)* JUN 2013 – JUL 2015

4. TITLE AND SUBTITLE

VIDEO TO TEXT (V2T) IN WIDE AREA MOTION IMAGERY

5a. CONTRACT NUMBER FA8750-13-C-0110

5b. GRANT NUMBER N/A

5c. PROGRAM ELEMENT NUMBER 63788F / 635322

6. AUTHOR(S)

Genshe Chen, Dan Shen, and Haibin Ling

5d. PROJECT NUMBER E3FM

5e. TASK NUMBER IF

5f. WORK UNIT NUMBER T1

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

Prime	Sub
Intelligent Fusion Technology, Inc.	Temple University
20271 Goldenrod Lane, Suite 2066	Dept of Computer & Info Science
Germantown, MD 20876	382 SERC Building, 1925 N. 12th St.
	Philadelphia, PA 19122

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

Air Force Research Laboratory/RIEA
525 Brooks Road
Rome NY 13441-4505

10. SPONSOR/MONITOR'S ACRONYM(S)

AFRL/RI

11. SPONSOR/MONITOR'S REPORT NUMBER

AFRL-RI-RS-TR-2015-215

12. DISTRIBUTION AVAILABILITY STATEMENT

Approved for Public Release; Distribution Unlimited. PA# 88ABW-2015-4224
Date Cleared: 4 SEP 2015

13. SUPPLEMENTARY NOTES

14. ABSTRACT

In this project, the Intelligent Fusion Technology, Inc. (IFT) team has developed a hard (video) and soft (text, voice chat) information fusion approach to automatically generate videos with annotation that can be easily used by future human or machine users. The tracking results following the standard format (.kw18) will also be output in a separated file for interfacing with other modules in the E2AT system integration. In the implementation, each entity corresponds to one tracklet with a unique track ID. Each entity consists of two sets of attributes: common attributes and uncommon attributes. Common attributes are those which will not change over the lifetime of a target track like type and color of the target. Uncommon attributes are those changing over time like target location, direction, and activity. The same sets of attribute definitions are used for entities constructed from both hard and soft data. The association, linkage, fusion, and concatenation can improve the visual tracking results through multi-intelligence information fusion.

15. SUBJECT TERMS

Visual tracking, Hard-soft fusion, automated video annotation, activity detection, event extraction, pattern of life, L1 tracker, cloud computing, deep learning.

16. SECURITY CLASSIFICATION OF:			**17. LIMITATION OF ABSTRACT**	**18. NUMBER OF PAGES**	**19a. NAME OF RESPONSIBLE PERSON** ERIK BLASCH
a. REPORT U	**b. ABSTRACT** U	**c. THIS PAGE** U	UU	108	**19b. TELEPHONE NUMBER** (*Include area code*) N/A

TABLE OF CONTENTS

LIST OF FIGURES

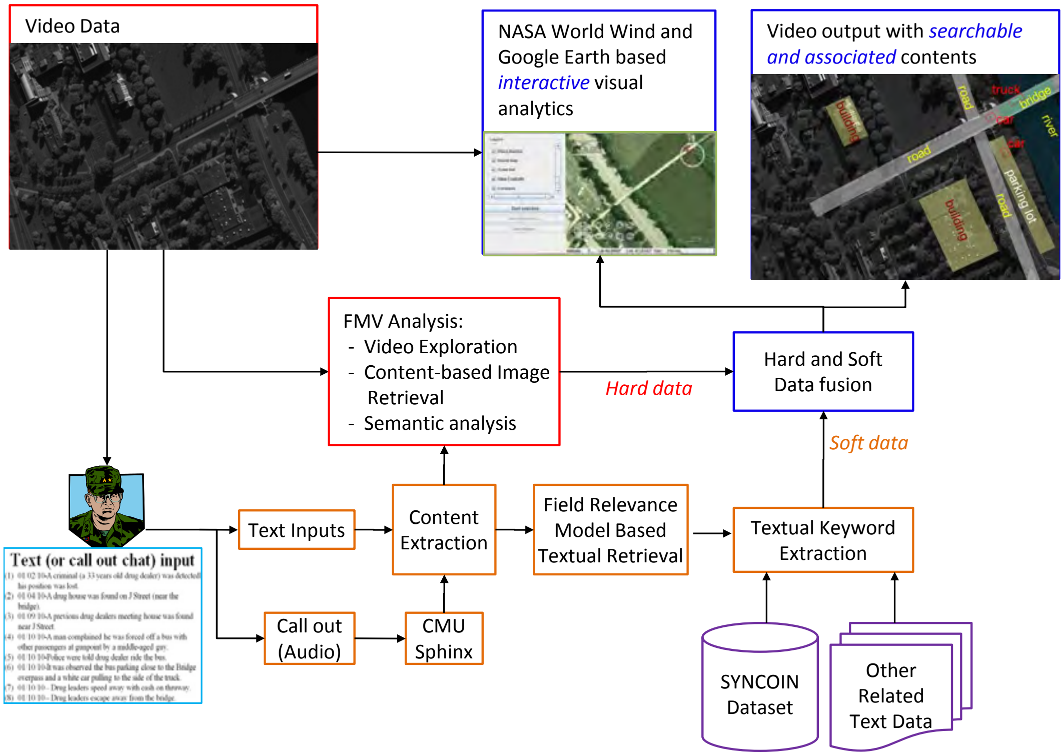
LIST OF TABLES

ACKNOWLEDGMENTS

The Authors thank Dr. Erik Blasch from Air Force Research Laboratory, Information Directorate (AFRL/RIEA) for his technical guidance during the effort.

1 SUMMARY

To provide video-to-text association to enable enhanced annotated video products, IFT team has developed a prototype (Figure 1), which fuses hard (video) and soft (text, voice chat) information to generate a video with annotation that can be easily used by future human or machine users. The tracking results following the required format (.kw18) will also be output in a separated file for interfacing with other modules in the E2AT system integration.


Video Data
NASA World Wind and Google Earth based *interactive* visual analytics
Video output with *searchable and associated* contents
road
truck
bridge
river
car
building
parking lot
FMV Analysis:
- Video Exploration
- Content-based Image Retrieval
- Semantic analysis
Hard data
Hard and Soft Data fusion
Soft data
Text Inputs
Content Extraction
Field Relevance Model Based Textual Retrieval
Textual Keyword Extraction
Call out (Audio)
CMU Sphinx
SYNCOIN Dataset
Other Related Text Data
Text (or call out chat) input

(1) 01 02 10-A criminal (a 33 years old drug dealer) was detected/ his position was lost.
(2) 01 04 10-A drug house was found on J Street (near the bridge).
(3) 01 09 10-A previous drug dealers meeting house was found near J Street.
(4) 01 10 10-A man complained he was forced off a bus with other passengers at gunpoint by a middle-aged guy.
(5) 01 10 10-Police were told drug dealer ride the bus.
(6) 01 10 10-It was observed the bus parking close to the Bridge overpass and a white car pulling to the side of the truck.
(7) 01 10 10 – Drug leaders speed away with cash on thruway.
(8) 01 10 10 – Drug leaders escape away from the bridge.

Figure 1: System Diagram

Specifically, IFT has:

1. Implemented various tracking algorithms such as L1 tracker, on-line adaboost (OAB), tracking-learning-detection tracker (TLD), online multiple instance learning tracker (MIL), visual tracking decomposition (VTD), likelihood of features tracker (Loft), compressive tracker (CT), and clustered set of structured uniformly dense robust features tracker (CSURF);
2. Compared these trackers on VIRAT dataset and a Skybox video;
3. Integrated tracking algorithms with the Kitware vsPlay software;
4. Investigated the motion detection, target classification and event detection;
5. Developed a cloud computing prototype for image registration and tracking;
6. Design and implement a hard-soft information fusion approach to improve the tracking accuracy.

2 INTRODUCTION

2.1 Problem Statement

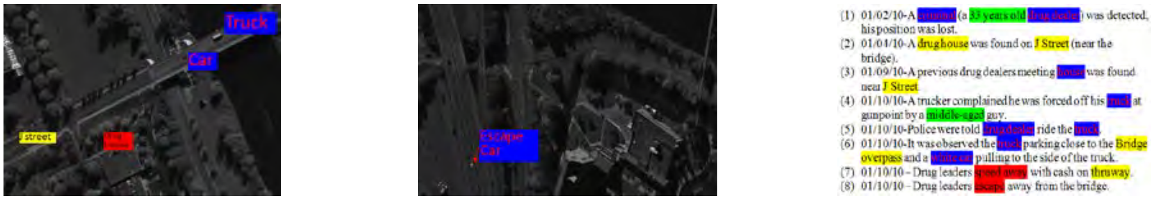
The primary focus E2T is to provide video-to-text association to enable enhanced annotated video products. Using current imagery exploitation tools such as FMV and Wide Area Motion Imagery (WAMI), the outputs (e.g., target features and track results) need to be presented in a format for textual association to enable hard (e.g., video) and soft (e.g., text) information fusion. As the operator calls out information in the video, the content must be transcribed in order to enable text extraction. Solutions proposed for the hard (video exploitation) and soft (text extraction) fusion require methods in metadata (e.g., time stamps) development for association, features (e.g., words and pixels) analysis for correlation, and contemporary processing techniques (e.g., track reports) for estimation. The secondary focus is to improve the persistent need to process and generate searchable content through the annotation, tagging, marking, and augmenting Image Intelligence (IMINT) data to better describe video products. Finally, the third focus is on automated interactive approaches between the operator, FMV data streams, and multi-media content to support FMV exploitation, access, annotation, indexing, storage, and linking of IMINT products (e.g. Geographical, moving, signals) to non-IMINT data products (e.g., open source, human, and communications) to enable reasoning (e.g., patterns of life). The three elements of FMVE include developments in (1) video-exploitation associated to text-extraction for annotated video outputs, (2) content-based image and textual retrieval, processing, and dissemination, and (3) interactive visual analytics for advanced operator reasoning.

The problems call for enhanced solutions to support operator video exploitation. For example, while the human viewer monitors the video feeds to recognize any significant content, their observations must be automatically translated into searchable display content as either textual and/or graphical products. The general FMVE developments would be physics-based (e.g., video) analysis to enable text-based (e.g., transcribed call-outs) hard-soft fusion. The FMVE operational advancements should leverage established technical tools and capabilities, build innovative image-to-analyst annotation functionality, and have a measureable evaluation of performance improvements.

2.2 Example Scenarios

To illustrate the objectives of the proposal, we give several application scenarios blow:

(1) Drug Dealer Scenario: In the scenario, there is a set of hard (a video from CLIF) and soft (Text in Figure 2 right) information covering a spatial and temporal window. From the text, we extract the key words (e.g., Criminal, Drug dealer, J Street, Truck, etc., highlighted in Figure 2, right) and match the keywords to the targets detected, identified and tracked in the video to generate the desired product – a video with annotation coming from the text file (Figure 2, left two).



(1) 01/02/10-A criminal (a 33 years old drug dealer) was detected, his position was lost.
(2) 01/04/10-A drughouse was found on J Street (near the bridge).
(3) 01/09/10-A previous drug dealers meeting house was found near J Street.
(4) 01/10/10-A trucker complained he was forced off his truck at gunpoint by a middle-aged guy.
(5) 01/10/10-Police were told drug dealer ride the truck.
(6) 01/10/10-It was observed the truck parking close to the Bridge overpass and a white car pulling to the side of the truck.
(7) 01/10/10 – Drug leaders speed away with cash on thruway.
(8) 01/10/10 – Drug leaders escape away from the bridge.

Figure 2: A Drug Dealer Tracking Scenario with Video Output and Text Input

(2) Joint Semantic and Appearance Retrieval: In this scenario, an operator aims to finding all images in a WAMI dataset containing environment images similar to an input one while having certain traffic pattern (e.g., two-way, heavy traffic). This will trigger the joint appearance-based (color and texture) and semantic-based retrieval. An illustration is shown in Figure 3. In the left of Figure 3, the operator provides an image query and the semantic query; in the right, the systems returns images resemble the query image and have the two-way heavy traffic as queried.

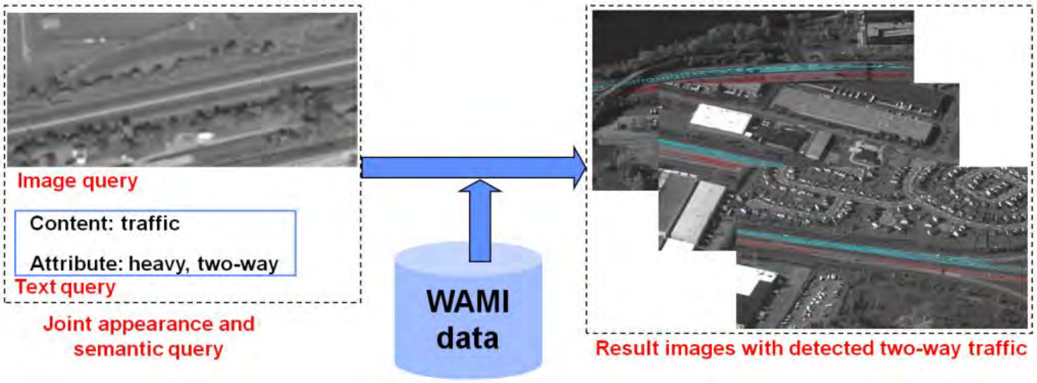


Image query
Content: traffic
Attribute: heavy, two-way
Text query
Joint appearance and semantic query
WAMI data
Result images with detected two-way traffic

Figure 3: Joint semantic and appearance query.

As shown in Figure 4, video tools include basic WAMI and FMV analysis (image registration and stitching, background modeling, cloud processing, target and context detection [1]) and situation awareness (target identification, L1-BPR tracking [2], [3], [8], simultaneous tracking and identification [4], Likelihood of Features Tracking (LOFT) using adaptive appearance models [18] - [22], multiple target tracking, scene parsing and activity recognition). For text exploration, we will use Penn. State U's SYNCOIN and IFT's experience on text processing to extract key words that will be used for video annotation from text or chat (use CMU Sphinx to convert chat to text). The key words include: What/Thing/Who, Where/Place and When/Time. For hard-soft information fusion, we will enhance the multi-parameter correlation/association in [14] to associate the hard (video with identified targets) and soft (key words extracted from text) information.

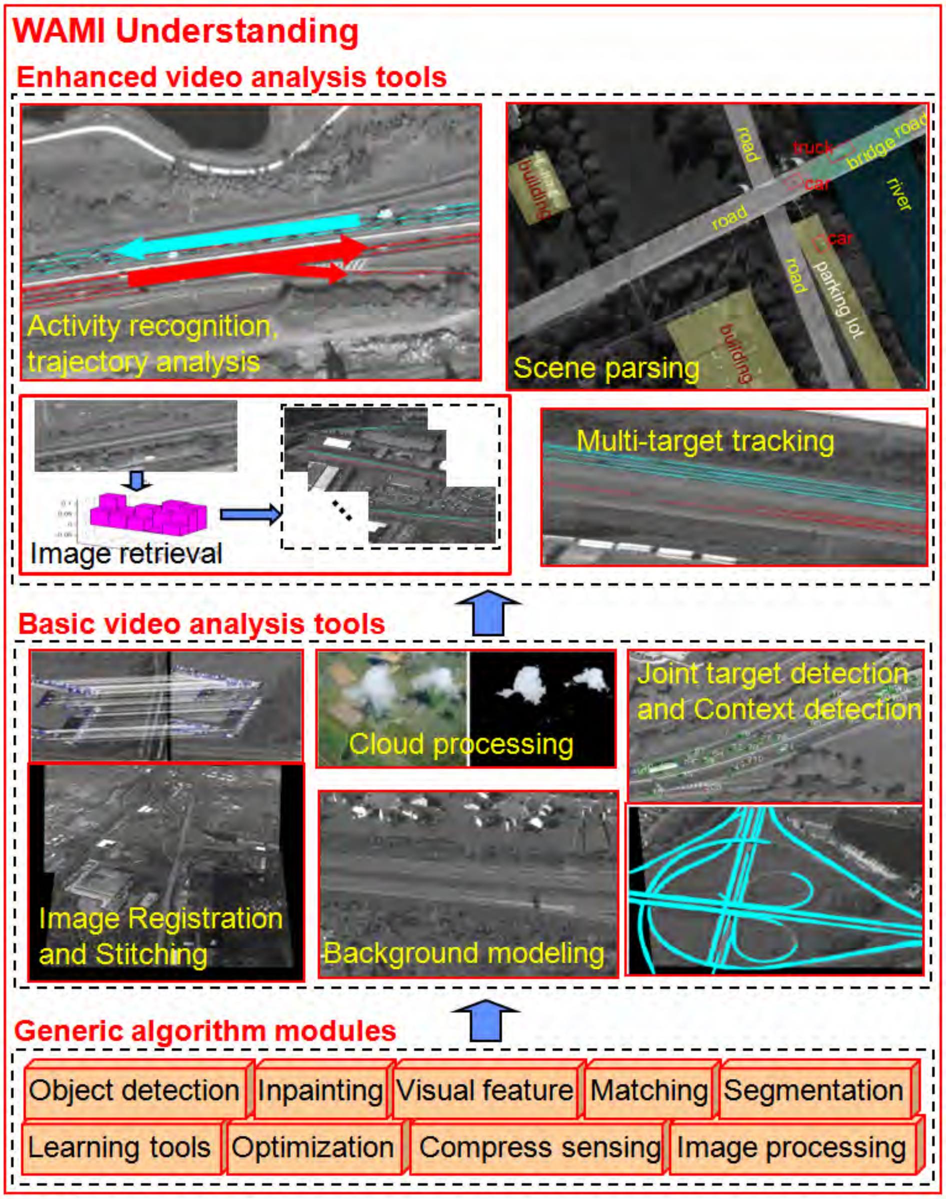


Figure 4: Video Tools

3 METHODS, ASSUMPTIONS, AND PROCEDURES

3.1 Visual Trackers

3.1.1 L1 tracker

L1 tracker is one of the state-of-the-art trackers that achieve good performance in tracking. It takes advantages of the sparse representation and compressive sensing techniques. L1 tracker also proves to be good in blur videos. L1 tracker used to be very slow due to the intensive computation of L1 solutions. But several techniques are applied to speed up the process, it now can be solved in real time.

3.1.1.1 L1 tracker framework

The particle filter provides an estimate of posterior distribution of random variables related to Markov chain. In visual tracking, it gives an important tool for estimating the target of next frame without knowing the concrete observation probability. It consists of two steps: prediction and update. Specially, at the frame *t*, denote x_t which describes the location and the shape of the target, $y_{1:t-1}=\{y_1,y_2,\cdots,y_{t-1}\}$ denotes the observation of the target from the first frame to the frame $t-1$. Particle filter precedes two steps with following two probabilities:

$$p(x_t|y_{1:t-1}) = \int p(x_t|x_{t-1})p(x_{t-1}|y_{1:t-1})\,dx_{t-1}, \tag{1}$$

$$p(x_t|y_{1:t}) = \frac{p(y_t|x_t)p(x_t|y_{1:t-1})}{p(y_t|y_{1:t-1})}. \tag{2}$$

The optimal state for the frame t is obtained according to the maximal approximate posterior probability: $x^*_t = \arg\max_x p(x|y_{1:t})$.

The posterior probability is approximated by using finite samples $S_t =\{x^1{}_t,x^2{}_t,\cdots,x^N{}_t\}$ with different weights $W=\{w^1{}_t,w^2{}_t,\cdots,w^N{}_t\}$ where N is the number of samples. The samples are generated by sequential importance distribution $\Pi(x_t|y_{1:t},x_{1:t-1})$ and weights are updated by:

$$w_t^i \propto w_{t-1}^i \frac{p(y_t|x_t^i)p(x_t^i|x_{t-1}^i)}{\Pi(x_t|y_{1:t},x_{1:t-1})}. \tag{3}$$

In the case of $\Pi(x_t|y_{1:t},x_{1:t-1})=p(x_t|x_{t-1})$, the above equation has a simple form $w_t^i \propto w_{t-1}^i p(y_t|x_t^i)$. Then, the weights of some particles maybe keep increasing and fall into the degeneracy case. To avoid such a case, in each step, samples are re-sampled to generate new sample set with equal weights according to their weights distribution.

The sparse representation model aims at calculating the observation likelihood for sample state x_t, i.e. $p(z_t|x_t)$. At the frame *t*, given the target template set $T_t=[t^1{}_t,t^2{}_t,\cdots,t^n{}_t]$, let $S_t =\{x^1{}_t,x^2{}_t,\cdots,x^N{}_t\}$ denote the sampled states and let $O_t =\{y^1{}_t,y^2{}_t,\cdots,y^N{}_t\}$ denote the corresponding candidate target patch in target template space. The sparse representation model is then:

$$y_t^i = T_t a_T^i + I a_I^i, \forall y_t^i \in O_t, \tag{4}$$

where *I* is the trivial template set (identity matrix) and $a^i_t=[a^i_T;a^i_I]$ is sparse. Additionally, nonnegative constraints are imposed on a^i_T for the robustness of the L1 tracker. Consequently, for each candidate target patch y^i_t, the sparse representation of y^i_t can be found via solving the following L1-norm related minimization with nonnegative constraints:

$$\min_a \frac{1}{2}\left\|y_t^i - Aa\right\|_2^2 + \lambda\|a\|_1, a \succcurlyeq 0, \tag{5}$$

where $A=[T_t,I,-I]$.

Finally, the observation likelihood of state x^i_t is given as:

$$p\left(z_t \middle| x_t^i\right) = \frac{1}{\Gamma}\exp\left\{-a\left\|y_t^i - T_t c_T^i\right\|_2^2\right\}, \tag{6}$$

where α is a constant controlling the shape of the Gaussian kernel, Γ is a normal factor and c^i_T is the minimizer of the L1-norm minimization restricted to T_t. Then, the optimal state x^*_t of frame t is obtained by:

$$x_t^* = arg\max_{x_t^i \in S_t} p\left(z_t \middle| x_t^i\right). \tag{7}$$

In addition, a template update scheme is adopted to overcome pose and illumination changes.

There are two types of templates in the template dictionary: target templates and trivial templates. The target templates are updated dynamically for representing target objects during the tracking process. The trivial templates (identity matrix I) is for representing occlusions, background and noise. However, since parts of objects may also be represented by the trivial templates, the region detected by the original tracker sometimes does not fit the target very accurately.

We take a modified version for improving tracking accuracy. The new model is based on the following observation. When there are no occlusions, the target in the next frame should be well approximated by a sparse linear combination of target templates with a small residual. Thus, the energy of the coefficients in *a* associate with trivial templates, named trivial coefficients, should be small. On the other hand, when there exist noticeable occlusions, the target in the next frame cannot be well approximation by any sparse linear combination of target templates, the large residual (corresponding to occlusions, background and noise in an ideal situation) will be compensated by the part from the trivial templates, which leads to a large energy of the trivial coefficients. The minimization is obviously not optimal since it does not differentiate these two cases.

In other words, to optimize the usage of the trivial templates in the tracking, we need to adaptively control the energy of the trivial coefficients. That is, when occlusions are negligible, the energy associated with trivial templates should be small. When there are noticeable occlusions, the energy should be allowed to be large. This motivation leads to the following minimization model for L1 tracker:

$$\min_a \frac{1}{2}\|y - A'a\|_2^2 + \lambda\|a\|_1 + \frac{\mu_t}{2}\|a_I\|_2^2,\ s.t. a_T \geqslant 0, \tag{8}$$

where *A'=[T*$_t$*,I],a=[a*$_T$*;a*$_I$*]* are the coefficients associated with target templates and trivial templates respectively, and the parameter μ_t is a parameter to control the energy in trivial templates. In our implementation, the value of μ_t for each state is automatically adjusted using the occlusion detection method. That is, if occlusions are detected, $\mu_t = 0$; otherwise μ_t is set as some pre-defined constant. The benefit of the additional L2 norm regularization term is that the trivial templates coefficients from minimization are small and lead to better tracking results.

A minimal error bounding method is proposed to reduce the number of needed L1 minimizations. Actually, the method is based on the following observation:

$$\|T_t a - y\|_2^2 \geq \|T_t \hat{a} - y\|_2^2, \forall a \in \mathbb{R}^N, \tag{9}$$

where

$$\hat{a} = arg \min_a \|T_t a - y\|_2^2. \tag{10}$$

Consequently, for any samples x^i_t, its observation likelihood has the following upper bound:

$$p(z_t|x_t^i) = \frac{1}{\Gamma}\exp\left\{-a\left\|y_t^i - T_t c_T^i\right\|_2^2\right\} \triangleq q(z_t|x_t^i), \tag{11}$$

where $q(y^i_t|x^i_t)$ is the probability upper bound for state x^i_t. It is seen that if $q(z_t|x_t) < \frac{1}{2N}\sum_{j=1}^{i-1} p(z_t|x_t^j)$, then the sample x^i_t will not appear in the resample set. In other words, x^i_t can be discarded without being processed. Thus, a two stage resample method is used to significantly reduce the number of samples needed in tracking.

The APG method is originally designed for solving the unconstrained minimization. Thus, we need to convert the constrained minimization model into an unconstrained problem. Let $1 \in R^N$ denote the vector with all entries are equal to 1 and let 1^{R}_{N+}(a) denote the indicator function defined by:

$$1_{\mathbb{R}_+^N}(a) = \begin{cases} 0,\ a \geqslant 0; \\ +\infty, otherwise. \end{cases} \tag{12}$$

So, the minimization equation in the modified version is equivalent to the following minimization problem:

$$\arg\min_a \frac{1}{2}\|y - A'a\|_2^2 + \lambda\|a\|_1 + \frac{\mu_t}{2}\|a_I\|_2^2 + 1_{\mathbb{R}_+^N}(a_T). \tag{13}$$

Then, the APG model will be:

$$\min F(a) + G(a), \tag{14}$$

$$F(a) = \frac{1}{2}\|y - A'a\|_2^2 + \lambda 1_T^T a_T + \frac{\mu_t}{2}\|a_I\|_2^2, \tag{15}$$

$$G(a) = \|a_I\|_1 + 1_{\mathbb{R}_+^N}(a_T). \tag{16}$$

To solve the above optimization problem, we could use the following algorithm:

```
Algorithm 1: Real Time Numerical algorithm for solving the minimization
(i) Set $a_0 = a_{-1} = 0 \in \mathbb{R}^N$ $and\ set$ $t_0 = t_{-1} = 1.$
(ii) For k=0,1..., iterate until convergence
        $\beta_{k+1} \coloneqq \alpha_k + \frac{t_{k-1}-1}{t_k}(\alpha_k - \alpha_{k-1});$
        $g_{k+1|T} \coloneqq \beta_{k+1|T} - \left(A'^T(A'\beta_{k+1} - y)\right)|_T/L \ - \lambda 1_T;$
        $g_{k+1|I} \coloneqq \beta_{k+1|I} - \left(A'^T(A\beta_{k+1} - y)\right)|_I/L \ - \mu\beta_{k+1|I}/L;$
        $\alpha_{k+1|T} \coloneqq \max(0, g_{k+1|T});$
        $\alpha_{k+1|I} \coloneqq \ \textstyle\sum_{\lambda/L}(g_{k+1|I});$
        $t_{k+1} \coloneqq (1 + \sqrt{1 + 4t_k^2})/2.$
```

Then, our final APG-L1 tracker algorithm will be as follows:

```
Algorithm 2:  APG-L1 Tracker
Input:
Current frame $F_t$;
Sample Set $\boldsymbol{S}_{t-1} = \{\boldsymbol{x}_{t-1}^{i}\}_{i=1}^{N}$;
Template set $\boldsymbol{T} = \{\boldsymbol{t}_i\}_{i=1}^{n}$;
for $i=1$ to $N$ do
    Drawing the new sample $\boldsymbol{x}_t^i$ from $\boldsymbol{x}_{t-1}^i$;
    Preparing the candidate patch $\boldsymbol{y}_t^i$ in template space;
    Solving the least square problem;
    Computing $q_i$ ;
end for
Sorting the samples in descent order according to $q$;
Setting $i=1$ and $\tau=0$;
while $i<N$ and $q_i \geq \tau$ do
    Solving the minimization via Algorithm 1;
    Computing the observation likelihood $p_i$;
    $\tau = \tau + \frac{1}{2N} p_i$;
    $i = i + 1$;
end while
Set $p_j = 0, \forall j \geq i$.
Output:
Finding the $\boldsymbol{x}_t^*$;
Detecting the occlusion and update μ;
Updating the template set $\boldsymbol{T}_{t-1}$ ;
Updating the sample set $\boldsymbol{S}_{t-1}$ with $p$.
```

3.1.2 Other Trackers --- Review

3.1.2.1 Compressive Tracker (CT)

Compressive tracking is a low computational complexity model based on features extracted in the compressed domain. By applying these feature extracted in preprocessing, the surrounding background is separated from the target object via a naive Bayes classifier. In the appearance model, features are selected by an information-preserving and non-adaptive dimensionality reduction from the multi-scale image feature space based on compressive sensing theories. The framework of compressive tracker is presented in table below.

```
Algorithm 3. Compressive Tracking
Input: video frames

1.Sample a set of image patches, $D^{\gamma} = \{\mathbf{z} | \|\mathbf{l}(\mathbf{z}) - \mathbf{l}_{t-1}\| < \gamma\}$ where $\mathbf{l}_{t-1}$is the
  tracking location at the ($t$-1)-th frame, and extract the features with low
```

dimensionality.

2.Use classifier B to each feature vector $\mathbf{v}(\mathbf{z})$ and find the tracking location $\mathbf{l}_t$ with the maximal classifier response.

3.Sample two sets of image patches $D^{\alpha} = \{\mathbf{z}|\|\mathbf{l}(\mathbf{z}) - \mathbf{l}_t\| < \alpha\}$ and $D^{\zeta,\beta} = \{\mathbf{z}|\zeta < \|\mathbf{l}(\mathbf{z}) - \mathbf{l}_t\| < \beta\}$ with $\alpha < \zeta < \beta$. $(\gamma, \alpha, \zeta$ and β are search radius of the set to detect the object location).

4.Extract the features with these two sets of samples and update the classifier parameters.

Output: Tracking location $\mathbf{l}_t$ and classifier parameters.

A random matrix R projects data from high dimensional image space $\mathbf{x} \in \mathbb{R}^m$ to a low dimensional space $\mathbf{v} \in \mathbb{R}^n$: $\mathbf{v} = R\mathbf{x}$, where n<<m. For each sample $\mathbf{z} \in \mathbb{R}^m$, its low-dimensional representation is $\mathbf{v} = (v_1, \cdots, v_n)^{\top} \in \mathbb{R}^n$. All elements in $\mathbf{v}$ are independently distributed and a naive Bayes classifier is modeled:

$$B(\mathbf{v}) = \log\left(\frac{\prod_{i=1}^{n} p(v_i|y=1)p(y=1)}{\prod_{i=1}^{n} p(v_i|y=0)p(y=0)}\right) = \sum_{i=1}^{n} \log(\frac{p(v_i|y=1)}{p(v_i|y=0)}) \tag{17}$$

where the uniform prior us assumed $p(y = 1) = p(y = 0)$, and $y \in \{0,1\}$ is a binary variable which represents the labels of the samples. The conditional distribution $p(v_i|y = 1)$ and $p(v_i|y = 0)$ in the classifier $B(\mathbf{v})$ are assumed to be Gaussian distributed with four parameters $\mu_i^1, \sigma_i^1, \mu_i^0, \sigma_i^0$ where

$$p(v_i|y = 1) \sim N(\mu_i^1, \sigma_i^1), p(v_i|y = 0) \sim N(\mu_i^0, \sigma_i^0) \tag{18}$$

The scalar parameter above are incrementally updated

$$\mu_i^1 \leftarrow \lambda\mu_i^1 + (1-\lambda)\mu^1$$

$$\sigma_i^1 \leftarrow \sqrt{\lambda(\sigma_i^1)^2 + (1-\lambda)(\sigma^1)^2 + \lambda(1-\lambda)(\mu_i^1 - \mu^1)^2}, \tag{19}$$

where $\lambda > 0$ is a learning parameter, $\sigma^1 = \sqrt{\frac{1}{n}\sum_{k=0|y=1}^{n-1}(v_i(k) - u^1)^2}$ and $\mu^1 = \frac{1}{n}\sum_{k=0|y=1}^{n-1} v_i(k)$. The above equations can be easily derived by maximal likelihood estimation.

3.1.2.2 Tracking-Learning-Detection Tracker (TLD)

TLD tracker is a framework designed for long-term tracking of an unknown object in a video stream. Its block diagram is shown in Figure 5. The components of the framework are characterized as follows: *Tracker* estimates the object's motion between consecutive frames under the assumption that the frame-to-frame motion is limited and the object is visible. The tracker is likely to fail and never recover is the object moves out of the camera view. *Detectors* treat every frame as independent and perform full scanning of the image to localize all appearances that have been observed and learned in the past. As any other detector, the detector makes two types of errors: false positive and false negative. *Learning* observes performance of both, tracker and detector, estimates detector's errors and generates training examples to avoid these errors in the future. The learning component assumes that both the tracker and the detector can fail. By virtue of the learning, the detector generalizes to more object appearances and discriminates against background.

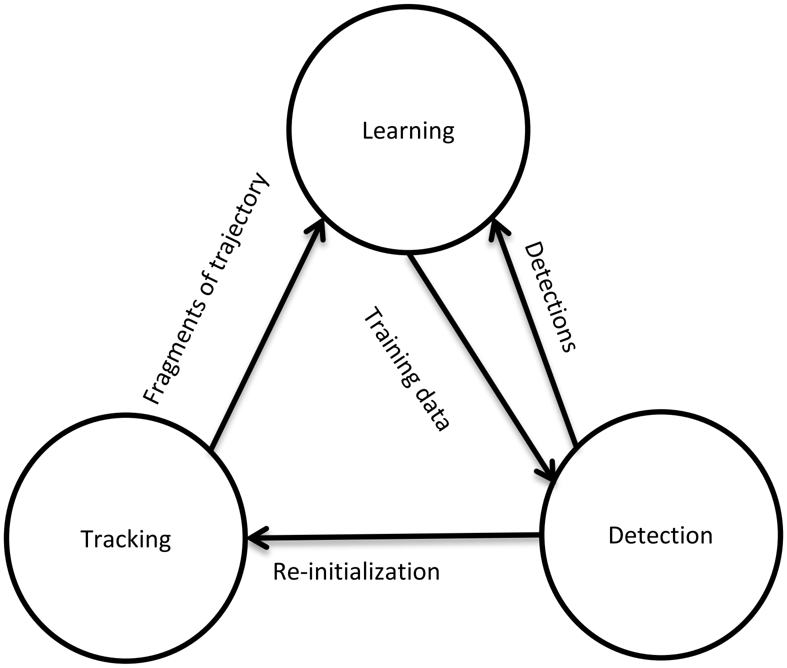


Figure 5: The block diagram of the TLD framework

3.1.2.3 Online Multiple Instance Learning Tracker (MIL)

MIL framework allows users to update the appearance model with a set of image patches, even though it is not known which image patch precisely captures the object of interest. This leads to more robust tracking results with fewer parameter tweaks. Weak classifiers are chosen sequentially to optimize the following criteria: $(\mathbf{h}_k, \alpha_k) = \text{argmax}_{\mathbf{h}\in\mathcal{H},\alpha} J(\mathbf{H}_{k-1} + \alpha\mathbf{h})$ where

$\mathbf{H}_{k-1}$ is the strong classifier made up of the first (k-1) weak classifiers, and $\boldsymbol{\mathcal{H}}$ is the set of all possible weak classifiers. In batch boosting algorithms, the objective function J is computed over the entire training dataset.

Algorithm 4. On-line MILBoost

```
Input: Dataset $\{X_i, y_i\}_{i=1}^N$, where $X_i = \{x_{i1}, x_{i2}, \cdots\}, y_i \in \{0,1\}$
Update all $M$ weak classifier in the pool with data $\{x_{ij}, y_i\}$.
Initialize $H_{ij} = 0$ for all $i, j$
for $k$ = 1to $K$ do
    for m=1 to $M$ do
        $p_{ij}^m = \sigma(H_{ij} + h_m(x_{ij}))$
        $p_i^m = 1 - \prod_j (1 - p_{ij}^m)$
        $\mathcal{L}^m = \sum_i (y_i \log(p_i^m) + (1 - y_i)\log(1 - p_i^m))$
    end for
    $m^* = \mathrm{argmax}_m \mathcal{L}^m$
    $\mathbf{h}_k(x) \leftarrow h_{m^*}(x)$
    $H_{ij} = H_{ij} + \mathbf{h}_k(x)$
end for
Output: Classifier $\mathbf{H}(x) = \sum_k \mathbf{h}_k(x)$, where $p(y|x) = \sigma(\mathbf{H}(x))$
```

For the current video frame, a training dataset $\{(X_1, y_1), ((X_2, y_2) \cdots\}$, where $X_i = \{x_{i1}, x_{i2}, \cdots\}$. The estimate of $p(y|x)$ is updated to maximize the log likelihood of the data. Thus the instance probability as $p(y|x) = \sigma(\mathbf{H}(x))$ where $\sigma(x) = \frac{1}{1+e^{-x}}$ is the sigmoid function: the bag probabilities $p(y|X)$ are modeled using the NOR model.

3.1.2.4 Online AdaBoost (OAB)

Boost tracker is a novel on-line AdaBoost feature selection algorithm for tracking. The distinct advantage of the method is its capability of on-line training. This allows adapt the classifier while tracking the object. Therefore appearance changes of the object (e.g. out of plane rotation, illumination changes) are handled quite naturally. Moreover, depending on the background the algorithm selects the most discriminating features for tracking resulting in stable tracking results. By using fast computable features (e.g. Haar-like wavelets, orientation histograms, local binary patterns) the algorithm runs in real-time.

The main idea of on-line boosting is the introduction of the *selectors*. They are randomly initialized and each of them holds a separate feature pool of weak classifiers. When a new training sample arrives the weak classifiers of each selector are updated. The best weak classifier (having the lowest error) is selected by the selector where the error of the weak classifier is estimated from samples seen so far. The complexity is determined by the number of selectors.

The part which requires most of the processing time is the updating of weak classifiers. In order to speed up this process, we propose as a modification to use a single "global weak classifier" pool (see Figure 6) which is shared by all selectors instead of single pools for each of them. The advantage of this modification is that now for each sample that arrives, all weak classifiers need to be updated only once. Then the selectors sequentially switch to the best weak classifiers need to be updated only once. Then the selectors sequentially switch to the best weak classifier with respect to the current estimated λ and the importance weight is passed on to the next selector. This procedure is repeated until all selectors are updated. Finally, at each time step an updated strong classifier is available. In order to increase the diversity of the weak classifiers and to allow changes in the environment, the worst weak classifier of the shared feature pool is replaced with a new randomly chosen one.

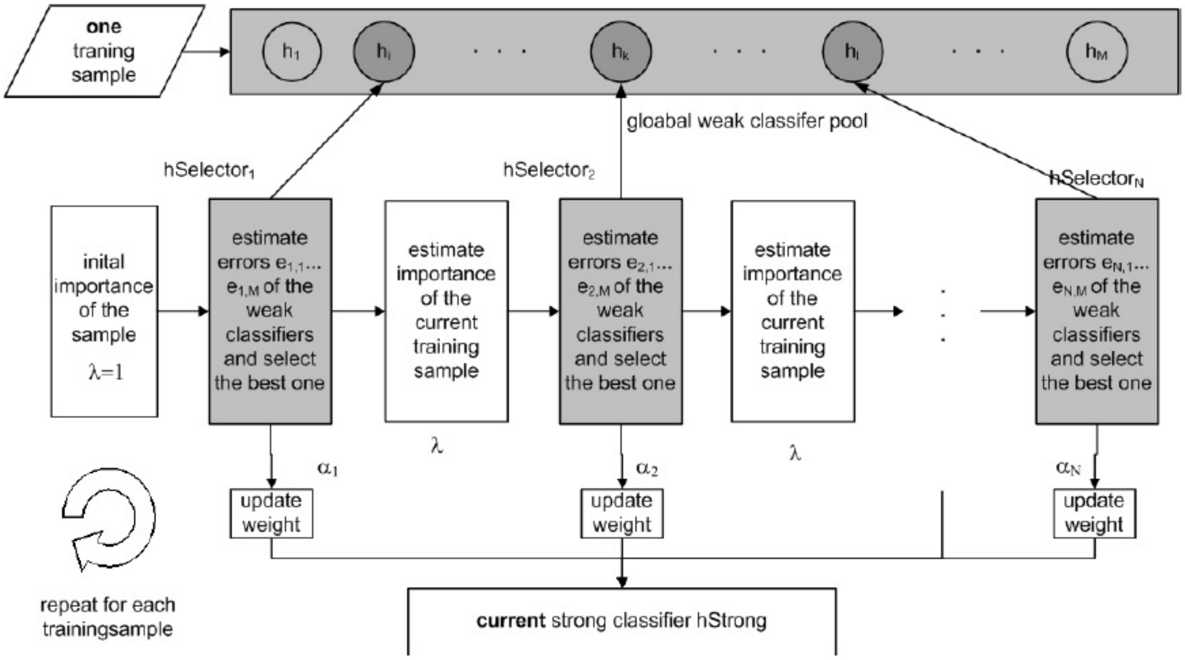

one traning sample
h1
hi
hk
hl
hM
gloabal weak classifer pool
hSelector1
hSelector2
hSelectorN
inital importance of the sample
λ=1
estimate errors e1,1... e1,M of the weak classifiers and select the best one
estimate importance of the current training sample
estimate errors e2,1... e2,M of the weak classifiers and select the best one
estimate importance of the current training sample
estimate errors eN,1... eN,M of the weak classifiers and select the best one
α1
λ
α2
λ
αN
update weight
update weight
update weight
repeat for each trainingsample
current strong classifier hStrong

Figure 6: Principle of on-line boosting for feature selection

3.1.2.5 Visual Tracking Decomposition (VTD)

VTD is a novel tracking algorithm that can work robustly in a challenging scenario such that several kinds of appearance and motion changes of an object occur at the same time. The algorithm is based on a visual tracking decomposition scheme for the efficient design of observation and motion models as well as trackers. The observation model is decomposed into multiple basic observation models that are constructed by sparse principal component analysis (SPCA) of a set of feature templates. Each basic observation model covers a specific appearance of the object. The motion model is also represented by the combination of multiple basic motion models, each of which covers a different type of motion. Then the multiple basic trackers are designed by associating the basic observation models and the basic motion models, so that each specific tracker takes charge of a certain change in the object. All basic trackers are then integrated into one compound tracker through an interactive Markov Chain Monte Carlo (IMCMC) framework in which the basic trackers communicate with one another interactively while run in parallel. By exchanging information with others, each tracker further improves its performance, which results in increasing the whole performance of tracking. Figure 7 shows the processes of observation model and tracking model decomposition.

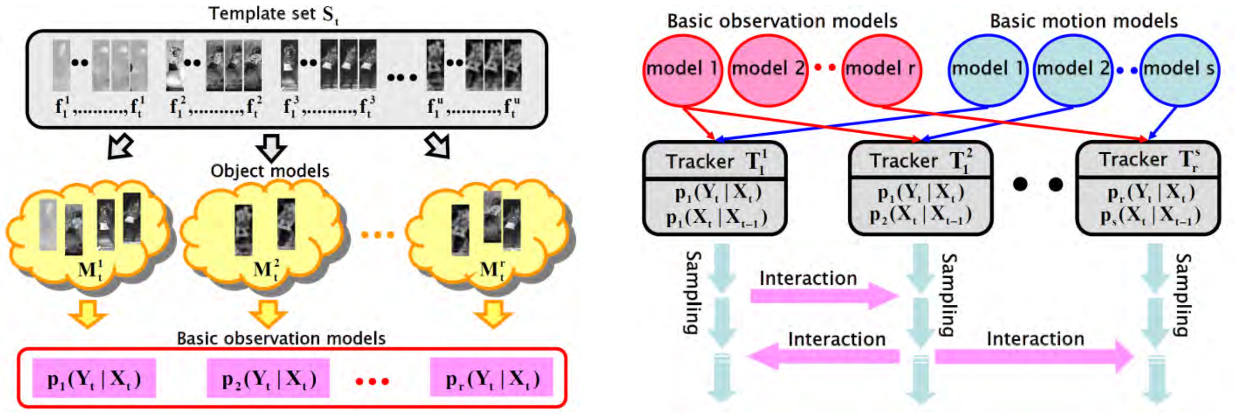


Figure 7: The process of observation model decomposition and the process of tracker decomposition [81].

3.1.2.6 Likelihood of Features Tracker (LoFT)

The likelihood of features tracking (LoFT) system is based on fusing multiple sources of information about the target and its environment akin to a track-before-detect approach. LoFT uses image-based feature likelihood maps derived from a template-based target model, object and motion saliency, track prediction and management, combined with a novel adaptive appearance target update model.

LoFT uses a recognition-based target localization approach using the maximum likelihood of the target being within the search region conditioned on a feature. A likelihood map is estimated for each feature by comparing feature histograms of the target within the search region using a sliding window based approach (see Fig. 8). Each pixel in the likelihood map indicates the posterior probability of that pixel belonging to the target. Fusing features enables adaptation of the tracker to dynamic environment changes and target appearance variabilities. Using a track-before-detect approach provides more robust localization especially for cluttered dense environments. Feature adaptation uses a weighted sum Bayes fusion rule that tends to perform better than other methods such as the product rule. The critical aspect in weighted sum fusion is the relative importance of feature maps. Each feature performs differently depending on the target characteristic and environmental situations during tracking. Equally weighted fusion of likelihood maps can decrease performance, when some of the features are not informative in that environment. The importance assigned to each feature can be adapted to the changes in target pose and the surrounding background. Temporal feature weight adaptation can improve performance under changes that are not explicitly modeled by the tracker.

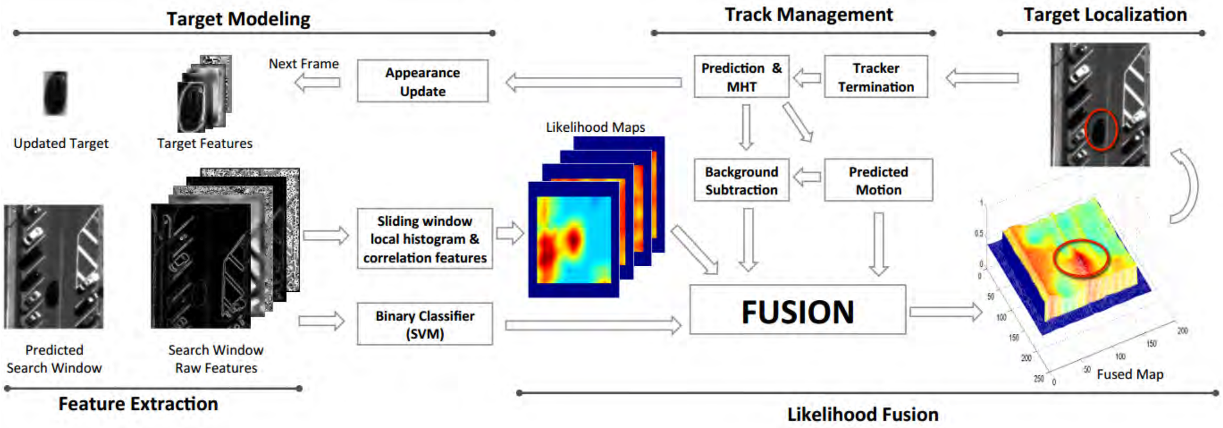


Figure 8: Likelihood of Features Tracking (LoFT) processing pipeline.

Fig. 8 shows major components including feature extraction, feature likelihood map estimation by combining with the template, vehicle detection using support vector machine (SVM) classification, fusion module that also incorporates prediction based motion and background subtraction based motion, to produce a fused likelihood for target localization after track extension. The track management includes termination module, prediction with or without multiple hypothesis tracking (MHT) and object appearance updating for adaptive target modeling [82].

3.1.2.7 Likelihood of Features Tracker (LoFT) Clustered Set of Structured Uniformly Dense Robust features tracker (CSURF)

Visual object tracking for surveillance applications poses challenges due to many factors such as the distractor objects in the scene, changing imaging conditions (e.g. illumination, viewpoint), scale, blur and appearance change. Many trackers in the literature utilize adaptive models to keep up with the dynamic appearance of objects. While some trackers utilize adaptive templates, others utilize keypoint based models (e.g. visual bags of words). Keypoint based tracking methods usually rely on a keypoint detector and descriptor in order to detect points on an object that can be robustly and repeatably detected in the subsequent frames and describe the regions around them with a robust descriptor. For objects with enough support (large scale) this approach works well, but it suffers from the lack of good feature points in aerial surveillance videos. CSURF [34] opts for detection based tracking, but propose a different approach using a clustered set of structured uniformly dense robust features (CSURF) to describe regions rather than finding interest points. The choice of descriptors is the Speeded-Up Robust Feature (SURF) descriptors but other robust descriptors of several features can be also employed. Based on these considerations, CSURF model consists of a collection of 64 dimensional SURF descriptors with structural information that represent the local image patches around regularly spaced points on the support of the object at a fixed scale.

3.2 Tracking System and Integration

3.2.1 Video Frame Registration

In this section, we present the video registration step. For the camera motion, we need to stabilize the video frames for moving target detection by frames registration. We stabilizes the video with respect a base frame in the video sequence over a short time period of about one second or 30 frames. This registration is accomplished by identifying prominent Beltrami color metric tensor features, which are a metric extension of structure tensor features that are matched between frames using a block matching approach. We call the combination of feature extraction with local region/block matching hybrid prominent feature-block matching. Once the features are available, a RANSAC approach is used to remap one frame into the coordinate system of the base video frame within the chunk of 30 frames. The homography model assumes that a single plane is sufficient to model the 3D scene which may not be valid for complex video sequences. Once the frame homographies are available, the flux tensor motion detection can be applied to detect moving target.

3.2.1.1 Necessity of video registration

Applications of unmanned aerial vehicles (UAVs) for surveillance, monitoring, situation awareness and resource management have steadily increased in recent years. Most UAVs have an onboard vision sub-system designed to acquire, preprocess, and transmit video images as they y over an area of interest. The large amount of video data and effects due to camera motion make it less suitable for direct analysis by human operators. There are a number of interesting challenges and opportunities to automate certain subtasks and assist the human analysts to improve the overall performance of such systems. One of the key challenges is due to the inherent camera motion - inevitable since the UAV is a moving platform. It is often impractical to assume that geo-location and orientation of the camera will be available at a resolution and robustness required by other algorithms. Effects due to occlusion and atmospheric conditions such as illumination, cloud motion, rain, etc, compound the challenge. As a result, detecting, locating, monitoring and tracking of objects in videos become severely hampered, and sometimes impossible. A number of interesting papers have appeared recently designed to address specific bottlenecks in the video analysis chain. The scope of these papers varies vastly with regard to assumptions on the range of motion and complexity of objects and their relative geospatial manifests.

Video registration is an essential task designed to deal with the effects caused by camera motion [83, 84] egomotion estimation is an alternative approach which we do not discuss in this paper. In this context, registration refers to the process of determining corresponding points or regions between two frames of a potentially dynamic scene taken at different times from different viewpoints by a mobile camera or other sensors. Both the vantage and time vary between the two images. Although image registration has been extensively addressed in the literature over the last three decades [85,86], a typical image pair that forms the basis of analysis has changed significantly over the years, including sensor configuration and scene dynamics that constitute the central disparity between frames. The set of assumptions used to simplify tractability issues and meet required (onboard) performance constraints are also forced to change with sensor technology. Although many methods have been proposed for airborne video registration [83, 84, 87, 88], it is still an open problem.

Our method uses edges and corners in each block of the image with certain confidence to extract the control points. The proposed technique shows good results and low error for image registration. Unmanned aerial vehicle have become important for gathering information and assessing a remote situation in many different applications. Although visual information is rich and discriminative it is usually difficult to analyze by computer. Another component of complexity consists in the unstable camera is due to the UAV motion. This generates many difficulties for motion analysis and object tracking. Therefore the registration becomes essential preliminary task to eliminate the errors caused by the camera displacement.

3.2.1.2 Prominent Feature Block-based Region Selection

The first step in registration is to detect salient or prominent features (PF) that are preserved under geometric image transformations. Spatial manifests such as corners, edges, contours, and regions prove to be effective in grouping pixels within an image, and establishing a basis for registration across a pair of images. These features are generally represented by points (corner, center of gravity and line intersection), lines (Hough transform) or areas (window), and facilitate registration when the correspondence between such features (drawn from two images) is established by some means. A set of corresponding PF block regions is the most desired baseline since it can be directly used to determine the parameters of the transformation function. Complex techniques do exist, which could be suitably adopted to exploit oriented properties of the spatial features, e.g. slope of line-segments, or inclusive angle associated to a corner etc. Such techniques would invariably involve fusing limited knowledge about the scene in terms of the parametric models used relating the oriented attributes of a selected feature across two instances. Depending on the context, the analysis may seek a balance between constancy and saliency of such attributes. We look at the distribution of feature point attributes within a PF block and use the cancroids for displacement vectors. The tensor representation of edges and corners provide consistent characteristics since they are related to the image structure.

3.2.1.3 Beltrami Color Metric Tensor

Many first and second derivative feature detectors and descriptors are available in the literature.7 We use the 2D color structure tensor defined in terms of the outer product of spatial gradients in each channel is given in Eq.1 with C_i representing image channels ($i = 3$ for RGB color), and further described in [90, 91]. The 2D grayscale structure tensor matrix is also referred to as the second moment or autocorrelation matrix [89].

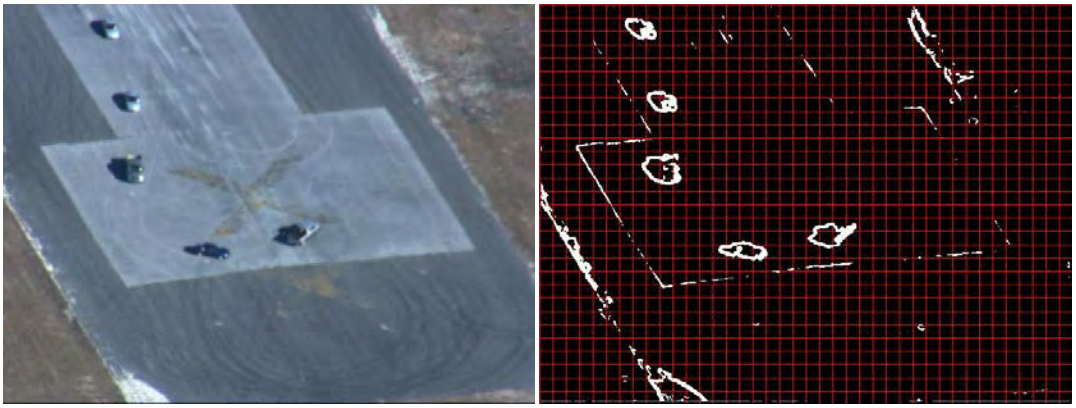
Local descriptors based on the two eigenvalues of the structure tensor provide information about the signal in orthogonal directions. Small eigenvalues are indicative of noise so the trace of $\mathbf{J}_C$ can filter these locations.

$$\mathbf{J_C} = \begin{bmatrix} \sum_{i=C_1,C_2,\ldots} \int_\Omega \left(\frac{\partial \mathbf{I}_i}{\partial x}\right)^2 d\mathbf{y} & \sum_{i=C_1,C_2,\ldots} \int_\Omega \frac{\partial \mathbf{I}_i}{\partial x}\frac{\partial \mathbf{I}_i}{\partial y} d\mathbf{y} \\ \sum_{i=C_1,C_2,\ldots} \int_\Omega \frac{\partial \mathbf{I}_i}{\partial x}\frac{\partial \mathbf{I}_i}{\partial y} d\mathbf{y} & \sum_{i=C_1,C_2,\ldots} \int_\Omega \left(\frac{\partial \mathbf{I}_i}{\partial y}\right)^2 d\mathbf{y} \end{bmatrix} \tag{20}$$

The eigenvalues of $\mathrm{J_C}$ are correlated with the local image properties of edgeness and cornerness, defined as $\lambda_1 \gg 0$, $\lambda_2 \approx 0$ and $\lambda_1 \approx \lambda_2 \gg 0$, respectively. For a 2D multi-spectral image, the Beltrami operator defines a metric on a two-dimensional manifold $(x, y, C_1(x, y), C_2(x, y), C_3(x, y))$, in the five-dimensional spatial-color space (x, y, C_1, C_2, C_3):

$$\begin{aligned} Beltrami(\mathbf{I_{RGB}}) &= \mathbf{det}(\mathcal{I} + \mathbf{J_C}) \\ &= 1 + \mathbf{trace}(\mathbf{J_C}) + \mathbf{det}(\mathbf{J_C}) \\ &= 1 + (\lambda_1 + \lambda_2) + \lambda_1\lambda_2 \end{aligned} \tag{21}$$

The determinant is the appropriate generalization of the gradient magnitude of intensity images to multispectral image gradients. In order to evaluate the color tensor matrix $\mathrm{J_C}$ two (convolution) scale factors are required - one for the spatial derivative (gradient) filters and one for integration (summation) filters. Fig. 9 shows an example of the 2D color structure (or Beltrami) tensor response. Each block containing high values of tensor magnitude responses is considered to be a PF macroblock and selected for the matching process described next. Reducing the total number of PF macroblocks reduces computational cost.



(a) Original frame

(b) Beltrami tensor response

Figure 9: Thresholded output of the 2D Beltrami tensor applied to the image shown.

The gird shows correspondence between location of salient features and non-overlapping macroblocks.

3.2.1.4 PF Block Region-Correspondences

Once the prominent feature (PF) macroblocks are selected based on an efficient evaluation and thresholding of Eq. 21, the next step is displacement or region correspondence matching. The matching process uses a PF block in the source image and searches for the best matching or most similar overlapping block contained with a search zone/window in the target image; each PF block is compared to all shifted overlapped areas by sliding the source PF block by one pixel. Two standard measures of similarity are the Euclidean distance and Normalized Cross Correlation (NCC). The L_1 approximation (referred to as sum of absolute differences (SAD)) of the L_2 Euclidean metric is used to reduce computation complexity. The NCC measure is less sensitive to absolute intensity changes between the source and target images due to the normalization terms in the denominator but is much more expensive to compute than SAD. Both were considered. The minimum of the SAD measure can be defined as,

$$\Delta X_{opt} = \arg\min_{\Delta X} \sum_{X \in \Omega} |\mathbf{I}(X + \Delta X, t - k) - \mathbf{I}(X, t)| \tag{22}$$

The NCC between target (or reference) image $\mathbf{I}(\mathrm{X}, \mathrm{t} - \mathrm{k})$ and source (or template) image $\mathbf{I}(\mathrm{X}, \mathrm{t})$ is defined as,

$$\gamma = \frac{\sum_{X \in \Omega}[\mathbf{I}(X + \Delta X, t - k) - \mu_{t-k}][\mathbf{I}(X, t) - \mu_t]}{\sqrt{\sum_{X \in \Omega}[\mathbf{I}(X + \Delta X, t - k) - \mu_{t-k}]^2 \sum_{X \in \Omega}[\mathbf{I}(X, t) - \mu_t]^2}} \tag{23}$$

where $\mu_{t-k} = \langle \mathbf{I}(X + \Delta X, t - k) \rangle$ and $\mu_t = \langle \mathbf{I}(X, t) \rangle$ are the local intensity means (averages) in the target and template image regions respectively and the denominator is the product of the local variances. The NCC for vector images (RGB color) can be appropriately extended.

We want to find the translation or displacement ΔX that maximizes the NCC measure,

$$\Delta X_{opt} = \arg\max_{\Delta X} |\gamma(\Delta X)| \tag{24}$$

The NCC can also be interpreted as the cosine of the angle between the two mean corrected region blocks. If we represent the mean subtracted pixels in the target and source windows as the vectors $\overrightarrow{W_T}, \overrightarrow{W_S}$, respectively, then, $NCC \equiv \gamma(\Delta X) = \left(\overrightarrow{W_T} \cdot \overrightarrow{W_S}\right) / \left(\|\overrightarrow{W_T}\| \cdot \|\overrightarrow{W_S}\|\right)$.

3.2.1.5 Filtering Motion Blocks

Figure 10 shows an example of the region-correspondences between two frames from a video sequence. The matching process is performed in the (grayscale or color) intensity space. There are two types of control points: those belonging to moving objects and those from the background. PF blocks containing moving objects are not suitable for homography since they introduce error due to the object motion between the two timesteps. Therefore it is more robust to use only points of the background, and discard those from the moving foreground objects using some type of motion filtering. The motion filtering consists of separating the displacement caused by camera motion from the displacement due to the moving objects in the scene. For this purpose we use the motion field statistics to determine which points belong to the background or foreground motion. Figure 11(a) shows an example of the displacement of the points of interest obtained by matching prominent blocks as described in previous section. We notice that the background has a dominant motion direction different from the foreground objects (cars). In general, in UAV-video, the number of prominent features due to moving objects in the scene will be small compared to the number of PF macroblocks in the background. Under this assumption the direction histogram of the motion field vectors can be used to detect the dominant displacement direction due to camera motion. We choose the maximum population of directions. Points that do not belong to this population are discarded from the set of control point macroblocks as shown in Figure 11 (b).

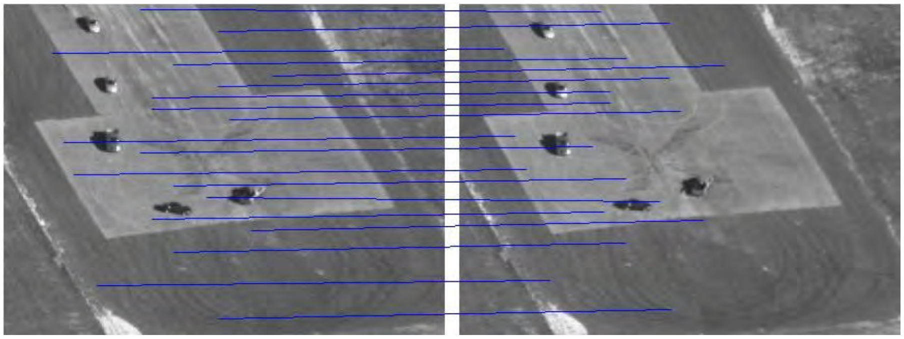
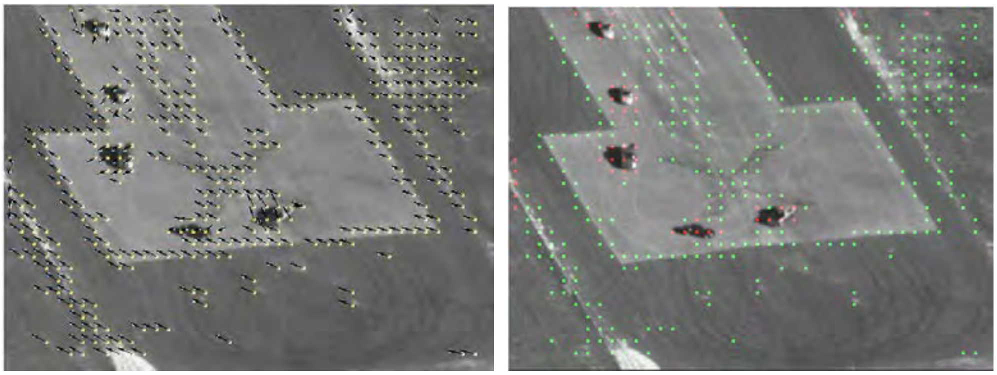


Figure 10: Region correspondences based on SAD matching.



(a) Displacement vectors

(b) After motion filtering

Figure 11: (a) PF block-based region correspondence vectors show the high quality of the matches. (b) Object motion blocks marked by red points are discarded, green ones kept.

3.2.1.6 Projective Transformation Estimation

Once region-based block correspondences are established, we need to compute the homography relating the two coordinate systems. This enables image $\mathbf{I}(\mathrm{X},\mathrm{t})$ to be mapped into the coordinate system of the base frame for a given video segment $\mathrm{I}(\mathrm{X},\mathrm{t}-\mathrm{k})$. Note that we are interested in finding a good solution for the homography, and not on finding the unique solution for the true 3D camera motion, as our goal is mainly to compensate for and remove the effects of the background or (dominant) ground plane motion. Since UAV imagery can have significant perspective effects a projective mapping is more accurate than a single global affine transformation. Other approaches include multiple local affine projections [92] and non-rigid transformations [93]. The projective mapping function or homography uses the coordinates of the corresponding PF block centroids (control points) to find a weighted least squares solution for the transformation matrix coefficients. The homography is used to warp the image at time t into the coordinate system of the base frame at time $(t-k)$. The two images, $\mathbf{I}(x,y,t)$ and $\mathbf{I}(x,y,t-k)$ can be related by a projective transformation (or homography) when the scene points are approximately planar. Let the image coordinates of the same scene point lying on the plane π be $P(x,y)$ and $P'(x',y')$, in the view at time t and (t - k) respectively. The two views can be related by the following homogeneous relationships:

$$x' = \frac{ax+by+c}{gx+hy+w} \tag{25}$$

$$y' = \frac{dx+ey+f}{gx+hy+w} \tag{26}$$

The homography can be written in matrix notation as:

$$\begin{bmatrix} x' \\ y' \\ w' \end{bmatrix} = \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & w \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \tag{27}$$

$$P' = \mathbf{A}_{(t-k,t)}P \tag{28}$$

This transforms position P observed at time t, to position P0 in the coordinate system at time $(t\ -\ k)$ via the projective transformation matrix (a backward transformation from time t to time $(t\ -\ k)$). Usually we assume $w\ =\ 1$ in matrix A.

Suppose we are given three images, $\mathbf{I}(x,y,t-2)$, $\mathbf{I}(x,y,t-1)$, $\mathbf{I}(x,y,t)$ with corresponding planar points, $P"$, P', P and homography transformation matrices $\mathbf{A}(t-1,t)$ and $\mathbf{A}(t-2,t-1)$ that projectively maps t to (t - 1) (i.e., Frame 2 to Frame 1) and (t-1) to (t-2) (i.e., Frame 1 to Frame 0), respectively. Without loss of generality we assume, for simplicity of notation, that the images are sequentially sampled at one unit time intervals, t, (t - 1), and (t - 2). We can then write the two respective projective transformations as,

$$P' = \mathbf{A}_{(t-1,t)}P \quad and \quad P'' = \mathbf{A}_{(t-2,t-1)}P' \tag{29}$$

and the composite or cumulative projective transformation relating pixels in frame t to pixels in frame (t-2) (i.e., pixels in Frame 2 to pixels in Frame 0), as the product of two homographies or projective maps/transformations:

$$P'' = \mathbf{A}_{(t-2,t-1)}\mathbf{A}_{(t-1,t)}P \tag{30}$$

In the general case, mapping pixel positions from frame t to corresponding pixel positions in the coordinate system of frame (t - k), we have

$$P(t-k,t) = \mathbf{A}_{(t-k,t)}P(t,t) \tag{31}$$

$$\mathbf{A}_{(t-k,t)} = \mathbf{A}_{(t-k,t-k+1)}\mathbf{A}_{(t-k+1,t-k+2)}\ldots.\mathbf{A}_{(t-2,t-1)}A_{(t-1,t)} \tag{32}$$

We also need to specify the coordinate system in which we reference or measure a pixel's position. Since the prime notation is limited, $P(t-k,t)$ denotes pixel position/geometry from image $\mathbf{I}(x,y,t)$ mapped to the coordinate system of image frame $\mathbf{I}(x,y,t-k)$ and $P(t,t)$ is the pixel position measured in its original coordinate system $\mathbf{I}(x,y,t)$. The elements of matrix **A** in Eq. 27 and 28 can be solved using weighted least squares, robust statistics such as LMedS or combinatorial methods such as RANSAC. Each pair of corresponding points provides three linear constraints that can be written in a matrix form $\mathbf{B}_i\mathbf{a} = 0$ as shown below,

$$\mathbf{B}_i\mathbf{a} = \begin{bmatrix} \mathbf{0}^{\mathrm{T}} & -w_i'\mathbf{x}_i^{\mathrm{T}} & -y_i'\mathbf{x}_i^{\mathrm{T}} \\ w_i'\mathbf{x}_i^{\mathrm{T}} & \mathbf{0}^{\mathrm{T}} & -x_i'\mathbf{x}_i^{\mathrm{T}} \\ y_i'\mathbf{x}_i^{\mathrm{T}} & x_i'\mathbf{x}_i^{\mathrm{T}} & \mathbf{0}^{\mathrm{T}} \end{bmatrix} \begin{bmatrix} \mathbf{a_1} \\ \mathbf{a_2} \\ \mathbf{a_3} \end{bmatrix} = \mathbf{0} \tag{33}$$

where $\mathbf{a}_i^T$ is the i^{th} row of **A** in Eq. 27. This above equation, $\mathbf{B}_i\mathbf{a} = 0$, is an equation *linear* in the unknown vector a. The matrix $\mathbf{B}_i$ is a 3×9 matrix, and a is a 9×1-vector made up of the entries of the matrix A,

$$\mathbf{a} = \begin{bmatrix} \mathbf{a}_1 \\ \mathbf{a}_2 \\ \mathbf{a}_3 \end{bmatrix}, \; \mathbf{a}_i = \begin{bmatrix} A(i,1) \\ A(i,2) \\ A(i,3) \end{bmatrix} \tag{34}$$

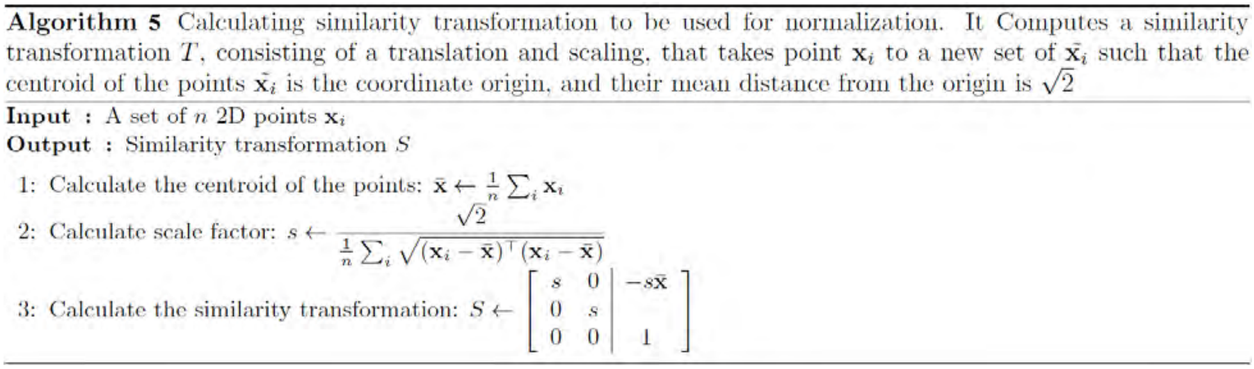
Notice that there are three equations in (33), however just two of them are linearly independent since the third row is obtained up to scale. Therefore each point correspondence provides two equations in the entries of A.

Based on this, (33) can be written as,

$$\mathbf{B}_i\mathbf{a} = \begin{bmatrix} \mathbf{0}^{\mathrm{T}} & -w'_i\mathbf{x}_i^{\mathrm{T}} & -y'_i\mathbf{x}_i^{\mathrm{T}} \\ w'_i\mathbf{x}_i^{\mathrm{T}} & \mathbf{0}^{\mathrm{T}} & -x'_i\mathbf{x}_i^{\mathrm{T}} \end{bmatrix} \begin{bmatrix} \mathbf{a}_1 \\ \mathbf{a}_2 \\ \mathbf{a}_3 \end{bmatrix} = \mathbf{0} \qquad (35)$$

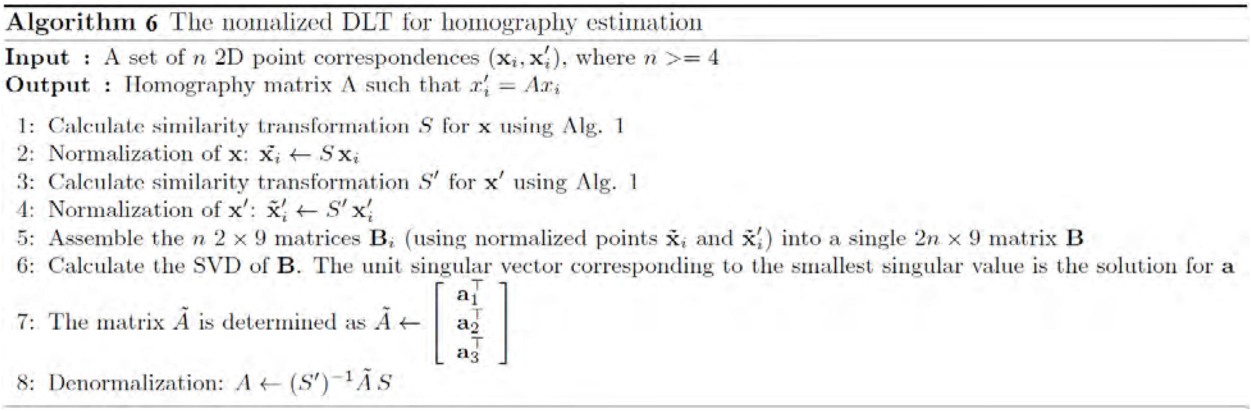
where $\mathbf{B}_i$ is the 2 × 9 matrix shown in Eq. (35). We solve Eq. (35) for **a** with 9 unknown elements using the normalized Direct Linear Transformation (DLT) approach given in Alg. 6. DLT provides for improved numerical stability and accuracy when solving for **A**.

Notice that for improving the accuracy of the results in the DLT algorithm, a normalization process (Alg. 5) has to be applied beforehand. This step is very important for less well conditioned problems such as DLT. Apart from improved accuracy of results, normalizing data has one more advantage, namely that an algorithm which incorporates an initial data normalization step will be invariant with respect to arbitrary choices of the scale and coordinate origin. As mentioned in [94] this is because the normalization step cancels out the effect of reference frame changes, by effectively choosing a canonical coordinate system for the measurement data. Therefore, algebraic minimization is carried out in a fixed canonical frame, and the DLT algorithm practically becomes invariant to similarity transformations. In order to give an idea of the importance of the normalization step in homography estimation using DLT, we performed a simulation using perturbed synthetic feature points.



Algorithm 5 Calculating similarity transformation to be used for normalization. It Computes a similarity transformation T, consisting of a translation and scaling, that takes point $\mathbf{x}_i$ to a new set of $\tilde{\mathbf{x}}_i$ such that the centroid of the points $\tilde{\mathbf{x}}_i$ is the coordinate origin, and their mean distance from the origin is $\sqrt{2}$

```
Input : A set of $n$ 2D points $\mathbf{x}_i$
Output : Similarity transformation $S$
Calculate the centroid of the points: $\bar{\mathbf{x}} \leftarrow \frac{1}{n}\sum_i \mathbf{x}_i$
Calculate scale factor: $s \leftarrow \dfrac{\sqrt{2}}{\frac{1}{n}\sum_i \sqrt{(\mathbf{x}_i - \bar{\mathbf{x}})^{\top}(\mathbf{x}_i - \bar{\mathbf{x}})}}$
Calculate the similarity transformation: $S \leftarrow \left[\begin{array}{cc|c} s & 0 & -s\bar{\mathbf{x}} \\ 0 & s & \\ 0 & 0 & 1 \end{array}\right]$
```



Algorithm 6 The nomalized DLT for homography estimation

```
Input : A set of $n$ 2D point correspondences $(\mathbf{x}_i, \mathbf{x}'_i)$, where $n >= 4$
Output : Homography matrix A such that $x'_i = Ax_i$
Calculate similarity transformation $S$ for $\mathbf{x}$ using Alg. 1
Normalization of $\mathbf{x}$: $\tilde{\mathbf{x}}_i \leftarrow S\,\mathbf{x}_i$
Calculate similarity transformation $S'$ for $\mathbf{x}'$ using Alg. 1
Normalization of $\mathbf{x}'$: $\tilde{\mathbf{x}}'_i \leftarrow S'\,\mathbf{x}'_i$
Assemble the $n$ $2 \times 9$ matrices $\mathbf{B}_i$ (using normalized points $\tilde{\mathbf{x}}_i$ and $\tilde{\mathbf{x}}'_i$) into a single $2n \times 9$ matrix $\mathbf{B}$
Calculate the SVD of $\mathbf{B}$. The unit singular vector corresponding to the smallest singular value is the solution for $\mathbf{a}$
The matrix $\tilde{A}$ is determined as $\tilde{A} \leftarrow \begin{bmatrix} \mathbf{a}_1^{\top} \\ \mathbf{a}_2^{\top} \\ \mathbf{a}_3^{\top} \end{bmatrix}$
Denormalization: $A \leftarrow (S')^{-1}\tilde{A}\,S$
```

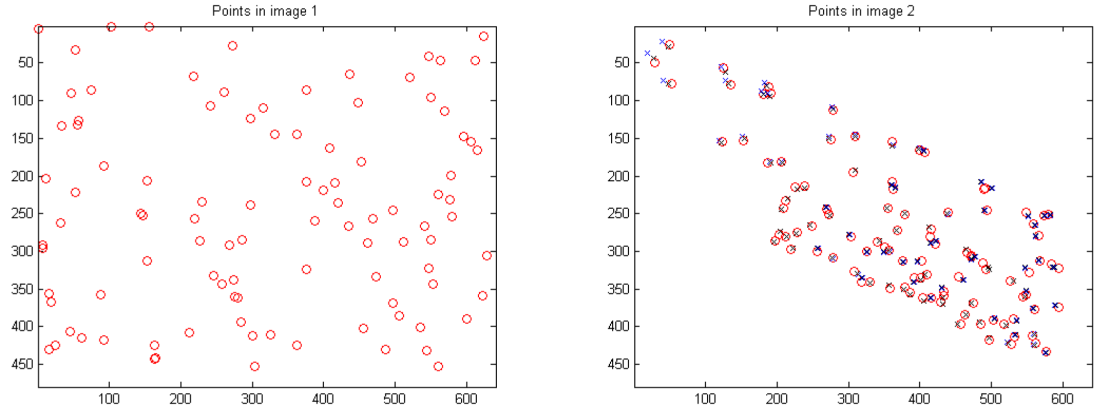


Figure 12: Simulated results to demonstrate the importance of using normalization step in DLT homography estimation algorithm.

Figure 12 shows a set of 100 points x that were randomly generated representing some feature points in the first image (**I**1). The blue-cross and black-cross marks in the right figure indicate the transformed points from I1 to I2 using non-normalized and normalized cases, respectively. The actual feature points in I2 are drawn in red-circles. For this experiment, we measured 25:75 average pixels error for the non-normalized case and 16:53 for the normalized case. The assumption is that there are no matching errors and the noise is equal additive Gaussian added to both I1 and I2.

Then a homography matrix A,

$$A = \begin{bmatrix} 0.5187 & 1.5228 & -25.3423 \\ 0.7852 & 0.6910 & -33.1947 \\ 0.0010 & 0.0002 & 1.0000 \end{bmatrix} \qquad (36)$$

is randomly generated which maps x in I1 to x0 = Ax in the second image I2. The image size is considered to be standard definition size of 640×480 pixels. Some noise (white Gaussian noise with zero mean and standard deviation two) are added to the feature points in both images as shown in Fig. 12. Then the homography transformation between the perturbed points in I1 and I2 has been estimated using the DLT algorithm, once using normalized points and the other without normalization. The geometric errors are computed using the estimated homography for both cases. For this experiment, we got 25:75 average pixels error for the non-normalized one and 16:53 for the normalized one. The estimated homography matrices in two cases of directly applying DLT or using a normalization method before DLT are as following, respectively:

$$A_{direct} = \begin{bmatrix} 0.5576 & 1.6262 & -40.0205 \\ 0.8366 & 0.7512 & -46.9605 \\ 0.0011 & 0.0003 & 1.0000 \end{bmatrix} \tag{37}$$

$$A_{normalized} = \begin{bmatrix} 0.5211 & 1.5410 & -26.5591 \\ 0.7938 & 0.6997 & -34.9253 \\ 0.0010 & 0.0002 & 1.0000 \end{bmatrix} \tag{38}$$

As can be seen, the estimated homography after applying a normalization step numerically is very close to the original used homography and moreover gives a better result. A similar experiment has been done for a case where just the points in I1 are perturbed and the points and homography is the same as the previous experiment. For this case we got 7:60 average pixels error for the non-normalized one and 6:40 for the normalized one. The estimated homography matrices in two cases of directly applying DLT or using a normalization method before DLT are as following, respectively:

$$A_{direct} = \begin{bmatrix} 0.5359 & 1.5653 & -31.6377 \\ 0.8064 & 0.7158 & -38.7444 \\ 0.0010 & 0.0002 & 1.0000 \end{bmatrix} \tag{39}$$

$$A_{normalized} = \begin{bmatrix} 0.5226 & 1.5343 & -26.7555 \\ 0.7908 & 0.6969 & -34.3263 \\ 0.0010 & 0.0002 & 1.0000 \end{bmatrix} \tag{40}$$

To demonstrate the necessity of taking the normalization step before applying DLT, another homography matrix

$$A = \begin{bmatrix} 1.9344 & 1.0464 & -84.3444 \\ 0.0704 & 1.7104 & -20.2447 \\ 0.0016 & 0.0012 & 1.0000 \end{bmatrix} \tag{41}$$

was randomly generated which maps x in **I**1 to $\mathrm{x}' = A\mathrm{x}$ in the second image **I**2 and then Gaussian noise with zero mean and standard deviation of two are added to the feature points in both images. Like the previous example, the homography transformation between the perturbed points in **I**1 and **I**2 has been estimated using the DLT algorithm, once using normalized points and the other without normalization. The geometric errors are computed using the estimated homography for both cases. For this experiment, we got 17:14 average pixels error for the non-normalized one and 12:26 for the normalized one. The estimated homography matrices in two cases of directly applying DLT or using a normalization method before DLT were tested for the second example. Figure 13 shows another simulation where the corresponding homography matrices were as follows:

$$A_{direct} = \begin{bmatrix} 1.9971 & 1.1315 & -101.8787 \\ 0.0748 & 1.8003 & -29.0347 \\ 0.0016 & 0.0014 & 1.0000 \end{bmatrix} \tag{42}$$

$$A_{normalized} = \begin{bmatrix} 1.9084 & 1.0463 & -83.0005 \\ 0.0620 & 1.7130 & -20.1356 \\ 0.0015 & 0.0013 & 1.0000 \end{bmatrix} \tag{43}$$

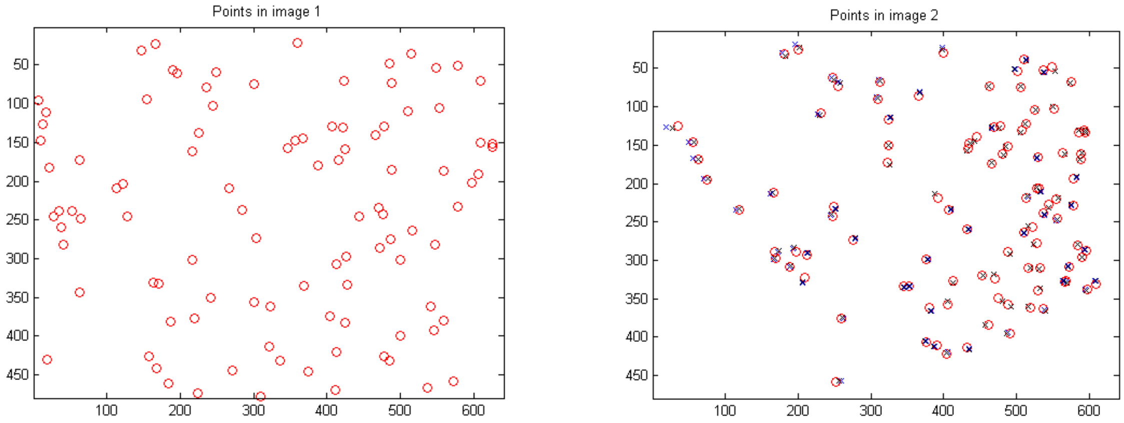
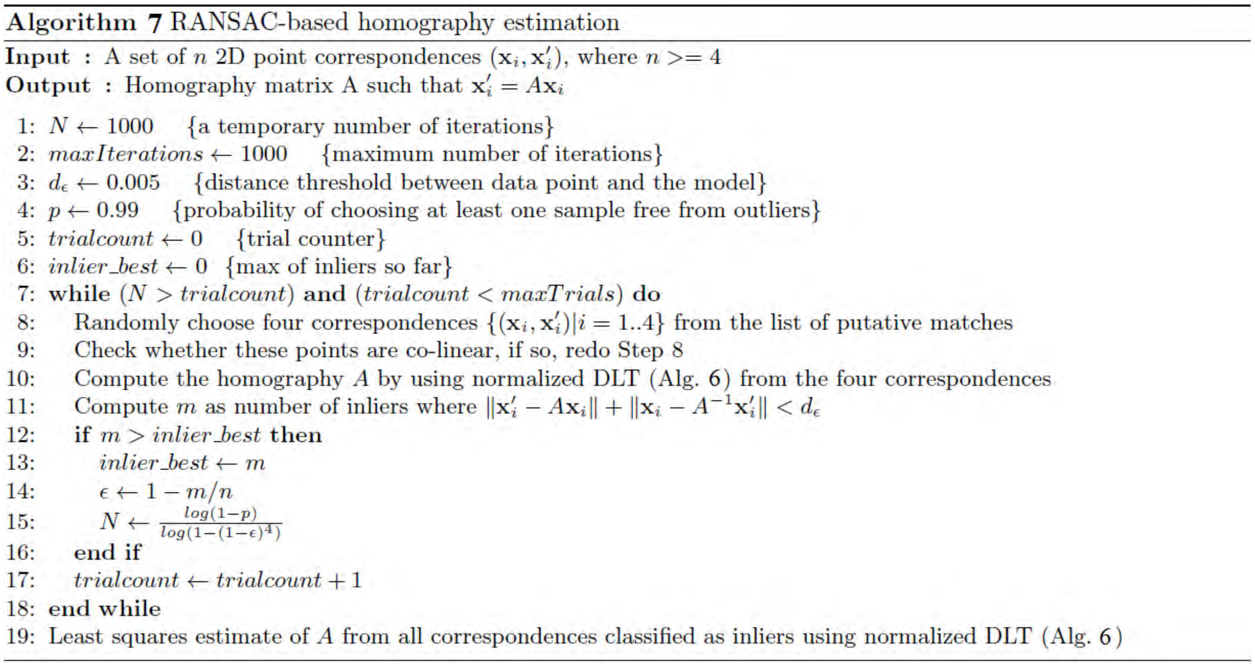


Figure 13: Simulated results to demonstrate the importance of using normalization step in DLT homography estimation algorithm.

The blue-cross and black-cross marks in the right figure indicate the transformed points from I1 to I2 using non-normalized and normalized cases, respectively. The actual feature points in I2 are drawn in red-circles. For this experiment, we measured 17:14 average pixels error for the non-normalized one and 12:26 for the normalized one. The assumption is that there are no matching errors and the noise is equal additive Gaussian added to both **I**1 and **I**2.

The DLT method, described in Algorithm 6, is used to solve Eq. 33. The DLT method is robust only if the dominant source of the noise is in the location measurement of corresponding feature points. The DLT method is not appropriate when there are mismatches corresponding to two putative features. For this purpose we use a method, based on RANSAC (Random Sample Consensus), to robustify the estimate with respect to false matches. The RANSAC-based homography estimation incorporating normalized DLT is described in Algorithm 7.



Algorithm 7 RANSAC-based homography estimation

```
Input : A set of $n$ 2D point correspondences $(\mathbf{x}_i, \mathbf{x}'_i)$, where $n >= 4$
Output : Homography matrix A such that $\mathbf{x}'_i = A\mathbf{x}_i$

1: $N \leftarrow 1000$   {a temporary number of iterations}
2: $maxIterations \leftarrow 1000$   {maximum number of iterations}
3: $d_\epsilon \leftarrow 0.005$   {distance threshold between data point and the model}
4: $p \leftarrow 0.99$   {probability of choosing at least one sample free from outliers}
5: $trialcount \leftarrow 0$   {trial counter}
6: $inlier\_best \leftarrow 0$   {max of inliers so far}
7: while $(N > trialcount)$ and $(trialcount < maxTrials)$ do
8:     Randomly choose four correspondences $\{(\mathbf{x}_i, \mathbf{x}'_i) | i = 1..4\}$ from the list of putative matches
9:     Check whether these points are co-linear, if so, redo Step 8
10:    Compute the homography $A$ by using normalized DLT (Alg. 6) from the four correspondences
11:    Compute $m$ as number of inliers where $\|\mathbf{x}'_i - A\mathbf{x}_i\| + \|\mathbf{x}_i - A^{-1}\mathbf{x}'_i\| < d_\epsilon$
12:    if $m > inlier\_best$ then
13:        $inlier\_best \leftarrow m$
14:        $\epsilon \leftarrow 1 - m/n$
15:        $N \leftarrow \frac{log(1-p)}{log(1-(1-\epsilon)^4)}$
16:    end if
17:    $trialcount \leftarrow trialcount + 1$
18: end while
19: Least squares estimate of $A$ from all correspondences classified as inliers using normalized DLT (Alg. 6)
```

The performance of RANSAC-based homography estimation using Algorithm 8 depends on the proportion of inliers and the number of iterations. The probability that after N iterations of RANSAC we have not picked a set of inliers is given by $(1 - g^4)^N$, with g = m/n being the proportion of the inliers; the behavior of this curve is shown in Figure 14 for three values of N (N = 10, 100, 1000). For the RANSAC iteration, the initial value of N in Algorithm 7 is initialized with a large value that is then adaptively updated in iteration using the Equation in Step 15 as shown in Figure 15.

An alternative to RANSAC is Least Median of Squares (LMedS) estimation, in which the model is selected using the median of the distances of all points in the dataset (whereas in RANSAC a minimum sized set of samples are randomly selected). As indicated in [94] LMS has the advantage of not requiring any threshold; however it fails if more than 50% of the data are outliers. There are other variations of RANSAC such as Adaptive-Scale Kernel Consensus (ASKC) which can be used as alternative robust estimators [95]. Alternatively some other methods which consider the hohomraphy estimated by Alg. 7 as an initialization and then iteratively try to minimize the error using Levenberg-Marquardt method in order to optimize the initial estimation [94].

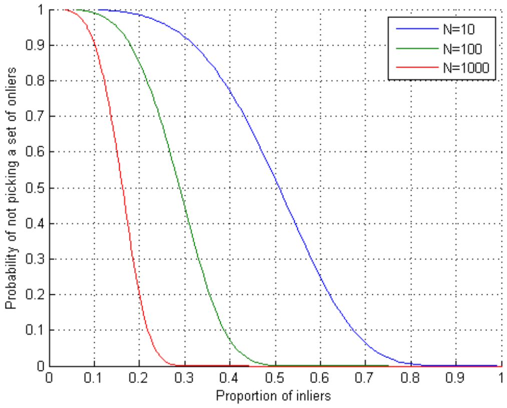


Figure 14: Evaluation of the robustness of the proposed RANSAC-based homography estimation in Alg. 7.

The horizontal axis indicates the proportion of inliers (g = m/n) and the vertical axis shows the probability that after N RANSAC iterations we have not picked a sufficient set of inliers based on the function $(1 - g^4)^N$ for three values of N.

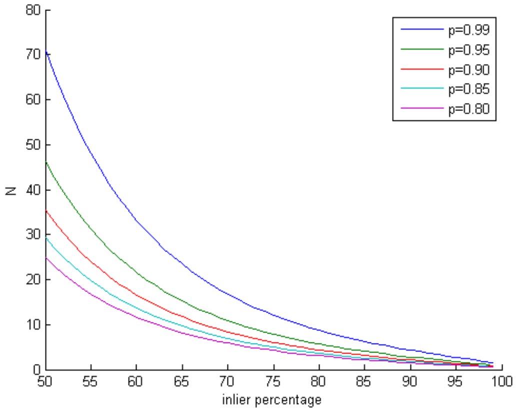


Figure 15: Plot of $N = \frac{\log(1-p)}{\log(a-(1-g)^4)}$,

Figure 15 gives an update for the number of iterations in Algorithm 7. N is adaptively determined in Step 15 of Algorithm 7. The horizontal axis indicates the inlier percentage. Plots for different values of p (see Step 4) that is related to the quality of the estimate is shown in different colors. For higher values of p more iteration are needed.

3.2.2 Motion Detection

In this section, we present how to detect moving target using flux tensor motion detection method. Motion blob detection is performed using our novel flux tensor method which is an extension to 3D grayscale structure tensor. Both the grayscale structure tensor and the proposed flux tensor use spatial-temporal consistency more efficiently, thus produce less noisy and more spatially coherent motion segmentation results compared to classical optical flow methods [96]. The flux tensor is more efficient in comparison to the 3D grayscale structure tensor since motion information is more directly incorporated in the flux calculation which is less expensive than computing eigenvalue decompositions as with the 3D grayscale structure tensor.

3.2.2.1 3D Structure Tensors

Structure tensors are a matrix representation of partial derivative information. As they allow both orientation estimation and image structure analysis they have many applications in image processing and computer vision. 2D structure tensors have been widely used in edge/corner detection and texture analysis, 3D structure tensors have been used in low-level motion estimation and segmentation [96, 97].

Under the constant illumination model, the optic-flow (OF) equation of a spatiotemporal image volume $\mathbf{I}$(x) centered at location $x = [x, y, t]$ is given by Eq. 44 [98] where, v(x) $= [v_x, v_y, v_t]$ is the optic-flow vector at **x**, **I** doing like this.

$$\begin{aligned}\frac{d\mathbf{I}(\mathbf{x})}{dt} &= \frac{\partial\mathbf{I}(\mathbf{x})}{\partial x}\,v_x + \frac{\partial\mathbf{I}(\mathbf{x})}{\partial y}\,v_y + \frac{\partial\mathbf{I}(\mathbf{x})}{\partial t}\,v_t \\ &= \nabla\mathbf{I}^T(\mathbf{x})\;\mathbf{v}(\mathbf{x}) = 0\end{aligned} \tag{44}$$

$\mathbf{v}(\mathbf{x})$ is estimated by minimizing Eq. 44 over a local 3D image patch $\mathbf{\Omega}(x, y)$, centered at x. Note that v_t is not 1 since spatial-temporal orientation vectors will be computed. Using Lagrange multipliers, a corresponding error functional e_{ls}(x) to minimize Eq. 44 using a least-squares error measure can be written as Eq. 45 where W(x, y) is a spatially invariant weighting function (e.g., Gaussian) that emphasizes the image gradients near the central pixel [97].

$$\begin{aligned}e_{ls}(\mathbf{x}) = \int_{\mathbf{\Omega}(\mathbf{x},\mathbf{y})} &\left(\nabla\mathbf{I}^T(\mathbf{y})\;\mathbf{v}(\mathbf{x})\right)^2\,W(\mathbf{x},\mathbf{y})\,d\mathbf{y} \\ &+\lambda\left(1 - \mathbf{v}(\mathbf{x})^T\mathbf{v}(\mathbf{x})\right)\end{aligned} \tag{45}$$

Assuming a constant **v(x)** within the neighborhood $\Omega(\mathrm{x}, \mathrm{y})$ and differentiating e_{ls}(x) to find the minimum, leads to the standard eigenvalue problem (Eq. 46) for solving $\hat{\mathrm{v}}$(x) the best estimate of **v(x)**.

$$\mathbf{J}(\mathbf{x},\mathbf{W})\;\hat{\mathbf{v}}(\mathbf{x}) = \lambda\;\hat{\mathbf{v}}(\mathbf{x}) \tag{46}$$

The 3D structure tensor matrix $\mathbf{J}(\mathbf{x}, \mathbf{W})$ for the spatiotemporal volume centered at x can be written in expanded matrix form, without the spatial filter W(x, y) and the positional terms shown for clarity, as Eq. 47.

$$\mathbf{J} = \begin{bmatrix} \int_{\mathbf{\Omega}} \frac{\partial \mathbf{I}}{\partial x}\frac{\partial \mathbf{I}}{\partial x} d\mathbf{y} & \int_{\mathbf{\Omega}} \frac{\partial \mathbf{I}}{\partial x}\frac{\partial \mathbf{I}}{\partial y} d\mathbf{y} & \int_{\mathbf{\Omega}} \frac{\partial \mathbf{I}}{\partial x}\frac{\partial \mathbf{I}}{\partial t} d\mathbf{y} \\ \int_{\mathbf{\Omega}} \frac{\partial \mathbf{I}}{\partial y}\frac{\partial \mathbf{I}}{\partial x} d\mathbf{y} & \int_{\mathbf{\Omega}} \frac{\partial \mathbf{I}}{\partial y}\frac{\partial \mathbf{I}}{\partial y} d\mathbf{y} & \int_{\mathbf{\Omega}} \frac{\partial \mathbf{I}}{\partial y}\frac{\partial \mathbf{I}}{\partial t} d\mathbf{y} \\ \int_{\mathbf{\Omega}} \frac{\partial \mathbf{I}}{\partial t}\frac{\partial \mathbf{I}}{\partial x} d\mathbf{y} & \int_{\mathbf{\Omega}} \frac{\partial \mathbf{I}}{\partial t}\frac{\partial \mathbf{I}}{\partial y} d\mathbf{y} & \int_{\mathbf{\Omega}} \frac{\partial \mathbf{I}}{\partial t}\frac{\partial \mathbf{I}}{\partial t} d\mathbf{y} \end{bmatrix} \tag{47}$$

A typical approach in motion detection is to threshold $\mathbf{trace}(\mathbf{J})$ (Eq. 48); but this results in ambiguities in distinguishing responses arising from stationary versus moving features (e.g., edges and junctions with and without motion), since $\mathbf{trace}(\mathbf{J})$ incorporates total gradient change information but fails to capture the nature of these gradient changes (i.e., spatial only versus temporal).

$$\mathbf{trace}(\mathbf{J}) = \int_{\mathbf{\Omega}} ||\nabla I||^2 d\mathbf{y} \tag{48}$$

To resolve this ambiguity and to classify the video regions experiencing motion, the eigenvalues and the associated eigenvectors of J are usually analyzed [99, 100]. However eigenvalue decomposition at every pixel is computationally expensive especially if real time performance is required.

3.2.2.2 Flux Tensors

In order to reliably detect only the moving structures without performing expensive eigenvalue decompositions, the concept of the ***flux tensor*** is proposed. Flux tensor is the temporal variations of the optical flow field within the local 3D spatiotemporal volume. Computing the second derivative of Eq. 44 with respect to t, Eq. 49 is obtained where, $\mathbf{a}(\mathbf{x}) = [ax, ay, at]$ is the acceleration of the image brightness located at x.

$$\begin{aligned} \frac{\partial}{\partial t}\left(\frac{d\mathbf{I}(\mathbf{x})}{dt}\right) &= \frac{\partial^2 \mathbf{I}(\mathbf{x})}{\partial x \partial t} v_x + \frac{\partial^2 \mathbf{I}(\mathbf{x})}{\partial y \partial t} v_y + \frac{\partial^2 \mathbf{I}(\mathbf{x})}{\partial t^2} v_t \\ &\quad + \frac{\partial \mathbf{I}(\mathbf{x})}{\partial x} a_x + \frac{\partial \mathbf{I}(\mathbf{x})}{\partial y} a_y + \frac{\partial \mathbf{I}(\mathbf{x})}{\partial t} a_t \end{aligned} \tag{49}$$

which can be written in vector notation as,

$$\frac{\partial}{\partial t}(\nabla \mathbf{I}^T(x)\mathbf{v}(\mathbf{x})) = \frac{\partial \nabla \mathbf{I}^T(\mathbf{x})}{\partial t}\, \mathbf{v}(\mathbf{x}) + \nabla \mathbf{I}^T(\mathbf{x})\, \mathbf{a}(\mathbf{x}) \tag{50}$$

Using the same approach for deriving the classic 3D structure, minimizing Eq. 49 assuming a constant velocity model and subject to the normalization constraint $\|\mathbf{v}(\mathbf{x})\| = 1$ leads to Eq. 51,

$$e^{F}_{ls}(\mathbf{x}) = \int_{\mathbf{\Omega}(\mathbf{x},\mathbf{y})} \left(\frac{\partial(\nabla \mathbf{I}^{T}(\mathbf{y}))}{\partial t}\, \mathbf{v}(\mathbf{x}) \right)^{2} W(\mathbf{x},\mathbf{y})\, d\mathbf{y} \\ + \lambda \left(1 - \mathbf{v}(\mathbf{x})^{T} \mathbf{v}(\mathbf{x}) \right) \tag{51}$$

Assuming a constant velocity model in the neighborhood $\Omega(x, y)$, results in the acceleration experienced by the brightness pattern in the neighborhood (x, y) to be zero at every pixel. As with its 3D structure tensor counterpart J in Eq. 47, the 3D flux tensor JF using Eq. 51 can be written as

$$\mathbf{J_F}(\mathbf{x}, \mathbf{W}) = \int_{\mathbf{\Omega}} W(\mathbf{x}, \mathbf{y}) \frac{\partial}{\partial t} \nabla \mathbf{I}(\mathbf{x}) \cdot \frac{\partial}{\partial t} \nabla \mathbf{I}^{\mathbf{T}}(\mathbf{x}) d\mathbf{y} \tag{52}$$

and in expanded matrix form as Eq. 53.

$$\mathbf{J_F} = \begin{bmatrix} \int_{\mathbf{\Omega}} \left\{ \frac{\partial^2 \mathbf{I}}{\partial x \partial t} \right\}^2 d\mathbf{y} & \int_{\mathbf{\Omega}} \frac{\partial^2 \mathbf{I}}{\partial x \partial t} \frac{\partial^2 \mathbf{I}}{\partial y \partial t} d\mathbf{y} & \int_{\mathbf{\Omega}} \frac{\partial^2 \mathbf{I}}{\partial x \partial t} \frac{\partial^2 \mathbf{I}}{\partial t^2} d\mathbf{y} \\ \int_{\mathbf{\Omega}} \frac{\partial^2 \mathbf{I}}{\partial y \partial t} \frac{\partial^2 \mathbf{I}}{\partial x \partial t}\, d\mathbf{y} & \int_{\mathbf{\Omega}} \left\{ \frac{\partial^2 \mathbf{I}}{\partial y \partial t} \right\}^2 d\mathbf{y} & \int_{\mathbf{\Omega}} \frac{\partial^2 \mathbf{I}}{\partial y \partial t} \frac{\partial^2 \mathbf{I}}{\partial t^2} d\mathbf{y} \\ \int_{\mathbf{\Omega}} \frac{\partial^2 \mathbf{I}}{\partial t^2} \frac{\partial^2 \mathbf{I}}{\partial x \partial t} d\mathbf{y} & \int_{\mathbf{\Omega}} \frac{\partial^2 \mathbf{I}}{\partial t^2} \frac{\partial^2 \mathbf{I}}{\partial y \partial t} d\mathbf{y} & \int_{\mathbf{\Omega}} \left\{ \frac{\partial^2 \mathbf{I}}{\partial t^2} \right\}^2 d\mathbf{y} \end{bmatrix} \tag{53}$$

As seen from Eq. 53, the elements of the flux tensor incorporate information about temporal gradient changes which leads to efficient discrimination between stationary and moving image features. Thus the trace of the flux tensor matrix which can be compactly written and computed as

$$\mathbf{trace}(\mathbf{J_F}) = \int_{\mathbf{\Omega}} || \frac{\partial}{\partial t} \nabla \mathbf{I} ||^2 d\mathbf{y}\ , \tag{54}$$

and can be directly used to classify moving and non-moving regions without the need for expensive eigenvalues decompositions. If motion vectors are needed then Eq. 51 can be minimized to get ^v(x) using

$$\mathbf{J_F}(\mathbf{x}, \mathbf{W})\ \hat{\mathbf{v}}(\mathbf{x}) = \lambda\ \hat{\mathbf{v}}(\mathbf{x}) \tag{55}$$

In this approach the eigenvectors need to be calculated at just moving feature points.

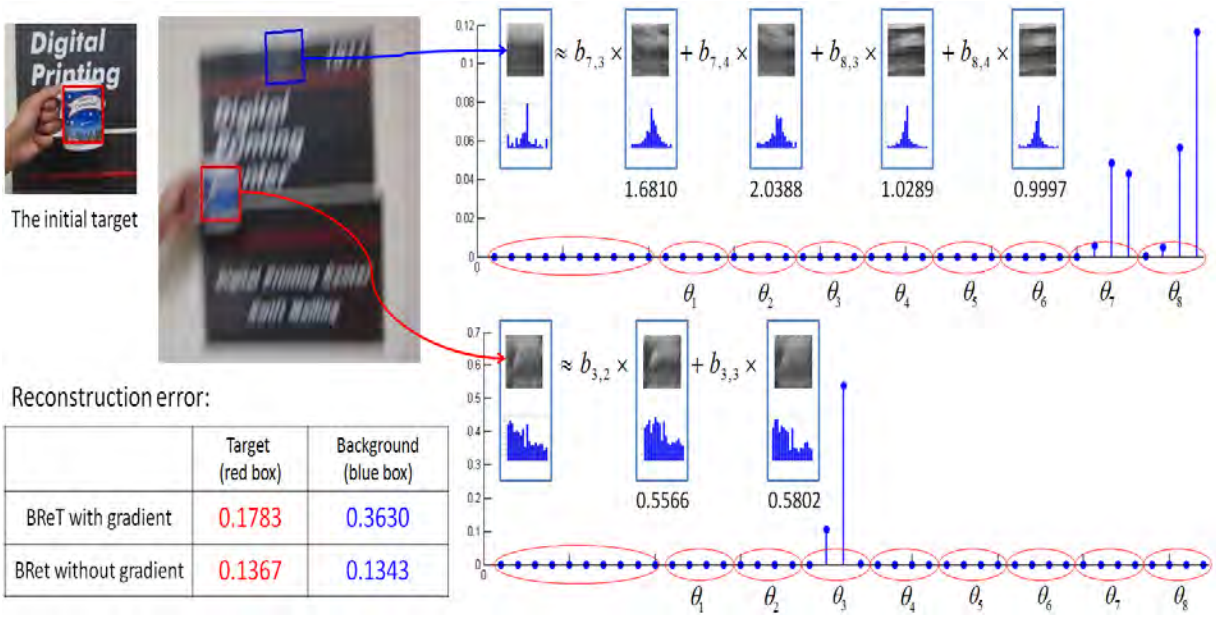
3.2.3 Blur-Resilient Tracking Using Group Sparsity

A Blur Resilient target Tracking algorithm (BReT) is developed by modeling target appearance by a groupwise sparse approximation over a template set. Since blur templates of different directions are added to the template set to accommodate motion blur, there is a natural group structure among the templates. In order to enforce the solution of the sparse approximation problem to have group structure, we employ the mixed $\ell_1 + \ell_1/\ell_2$ norm to regularize the model coefficients. Having observed the similarity of gradient distributions in the blur templates of the same direction, we further boost the tracking robustness by including gradient histograms in the appearance model. Then, we use an accelerated proximal gradient scheme to develop an efficient algorithm for the non-smooth optimization resulted from the representation. After that, blur estimation is performed by investigating the energy of the coefficients, and when the estimated target can be well approximated by the normal templates, we dynamically update the template set to reduce the drifting problem. Experimental results show that the proposed BReT algorithm outperforms state-of-the-art trackers on blurred sequences.

3.2.3.1 Motivation and Background

In our previous experiments we found that blur effect often challenges visual tracking algorithms and such effect appears frequently in arial videos due to fast camera motion. For this reason, we plan to develop a blur resilient visual tracking algorithm. Unlike other challenges in visual tracking, blur effect has not been seriously addressed in tracking algorithms except in a few recent studies. In particular, on our previous work, a blur driven tracker using sparse representation is proposed, which incorporates blur templates of different directions into the template space to model blur degradations. However, though the enhanced appearance space is more expressive, ambiguity also increases. For example, a target candidate that belongs to the background might be well represented by some blur templates. Also, the templates of the blur driven tracker are fixed, therefore when the appearance of the target changes significantly, the tracker is susceptible to drifting.

To address these issues, we propose a robust blurred target tracking algorithm using group sparse representation under a particle filter framework with enhanced template space. Three components distinguish our work from previous ones: (1) since blur templates of different directions are added to the template space and the motion blur of the target always tends only one direction in a frame, there is a natural group structure among the templates, i.e., the blur templates of one direction belong to the same group. In order to enforce the solution of the sparse representation of a target candidate to have group structure, we adopt a structured sparsity inducing norm which is a combination of ℓ_1 norm and a sum of ℓ_2 norms over groups of variables; (2) to account for the increase of ambiguity in the template space after enhancing it with blur templates, based on the observation that blur templates of the same direction have much more similar gradient histograms than blur templates in different directions, we use a combination of the reconstruction error and a sum of weighted distances between gradient histograms of a target candidate and each of the non-trivial templates as loss function. The resulting non-smooth convex optimization problem is solved using an accelerated proximal gradient method that guarantees fast convergence; and (3) in order to capture the appearance changes of the target and reduce the drifting problem, we perform blur detection by investigating the energy of the reconstruction coefficients. The template set is updated dynamically when two criteria based on the coefficients associated with templates are satisfied. Figure 16 illustrates the intution of the developed BReT tracker.


Digital Printing
The initial target
$\approx b_{7,3}\times$ $+b_{7,4}\times$ $+b_{8,3}\times$ $+b_{8,4}\times$
1.6810
2.0388
1.0289
0.9997
θ_1 θ_2 θ_3 θ_4 θ_5 θ_6 θ_7 θ_8
$\approx b_{3,2}\times$ $+b_{3,3}\times$
0.5566
0.5802
θ_1 θ_2 θ_3 θ_4 θ_5 θ_6 θ_7 θ_8

Reconstruction error:

	Target (red box)	Background (blue box)
BReT with gradient	0.1783	0.3630
BRet without gradient	0.1367	0.1343

Figure 16: Intution of the BReT tracker

Top left: The tracking results of BReT with and without gradient information, indicated by red box and blue box respectively. **Bottom left**: the reconstruction error of the two candidates measured by $0.5\|\mathbf{Tc}-\mathbf{y}\|_2^2$ using different tracking approaches. **Right**: The group sparse representation of the two candidates using BReT with gradient information, the L1 distance between the gradient histograms of the estimated target and each of the selected templates are also given.

3.2.3.2 Review of the Blur-driven Tracker (BLUT)

The particle filter is a Bayesian sequential importance sampling technique for estimating the posterior distribution of state variables characterizing a dynamic system. It uses finite set of weighted samples to approximate the posterior distribution regardless of the underlying distribution. For visual tracking, we use $\mathbf{x}_t$ as the state variable to describe the location and shape of the target at time t. Given all available observations $\mathbf{y}_{1:t}=\{\mathbf{y}_1,\mathbf{y}_2,\ldots,\mathbf{y}_t\}$ up to time t, the posterior $p(\mathbf{x}_t\,|\,\mathbf{y}_{1:t})$ is approximated by a set of N samples $\{\mathbf{x}_t\}_{i=1}^N$ with importance weights w_t^i. The optimal $\mathbf{x}_t$ is obtained by maximizing the approximate posterior probability: $\mathbf{x}_t^* = argmax_{\mathbf{x}}\,p(\mathbf{x}\,|\,\mathbf{y}_{1:t})$.

In order to model the blur degradations, blur templates are incorporated into the appearance space. The appearance of the tracking target $\mathbf{y}\in\mathrm{R}^d$ is represented by templates $\mathbf{T}=[\mathbf{T}_a,\mathbf{T}_b,\eta\mathbf{I}]$,

$$\mathbf{y} = [\mathbf{T}_a, \mathbf{T}_b, \eta\mathbf{I}]\begin{bmatrix}\mathbf{a}\\ \mathbf{b}\\ \mathbf{e}\end{bmatrix} \hat{=} \mathbf{Tc}, \quad \text{s.t.} \quad \mathbf{c}_T \pm 0, \tag{56}$$

where $\mathbf{T}_a = [\mathbf{t}_1, \cdots, \mathbf{t}_{n_a}] \in \mathrm{R}^{d \times n_a}$ contains n_a normal templates, $\mathbf{T}_b = [\mathbf{t}_{1,1}, \cdots, \mathbf{t}_{1,n_l}, \cdots, \mathbf{t}_{n_\theta,1}, \cdots, \mathbf{t}_{n_\theta,n_l}] \in \mathrm{R}^{d \times n_b}$ contains n_b blur templates, $\mathbf{I}$ is the $d \times d$ identity matrix containing the trivial templates used for modeling image corruption, η is used to control the weight of the trivial templates. Accordingly, $\mathbf{a} = (a_1, a_2, \cdots, a_{n_a})^{\mathrm{T}} \in \mathrm{R}^{n_a}$, and $\mathbf{b} \in \mathrm{R}^{n_b}$ are called *normal coefficients* and *blur coefficients* respectively, $\mathbf{e} = (e_1, e_2, \cdots, e_d)^{\mathrm{T}}$ is called *trivial coefficients*, $\mathbf{c} = [\mathbf{a}^{\mathrm{T}}, \mathbf{b}^{\mathrm{T}}, \mathbf{e}^{\mathrm{T}}]^{\mathrm{T}}$ and $\mathbf{c}_T = [\mathbf{a}^{\mathrm{T}}, \mathbf{b}^{\mathrm{T}}]^{\mathrm{T}}$.

The first normal template $\mathbf{t}_1$ is obtained from the unblurred object patch of the target in the first frame, which is usually selected manually or by detection algorithms, other templates are shifted from it. Given a blur free patch I of the target, different blurred versions I_b of the target can be modeled as convolving I with different kernels. In our framework, $\mathbf{t}_{i,j} = \mathbf{t}_1 \otimes \mathbf{k}_{i,j}$ is the $(i,j)^{th}$ blur template, where $\mathbf{k}_{i,j}$ is a Gaussian kernel that represents a 2D motion toward direction θ_i with magnitude l_j, where $\theta_i \in \Theta = \{\theta_1, \cdots, \theta_{n_\theta}\}$, and $l_j \in \mathrm{L} = \{l_1, \cdots, l_{n_l}\}$. Consequently, we have $n_b = n_\theta \times n_l$ blur templates. Based on the directions of the blur kernels, we have $\mathbf{b} = [\mathbf{b}_1^{\mathrm{T}}, \cdots, \mathbf{b}_{n_\theta}^{\mathrm{T}}]^{\mathrm{T}} \in \mathrm{R}^{n_b}$, where $\mathbf{b}_i = (b_{i,1}, b_{i,2}, \cdots, b_{i,n_l})^{\mathrm{T}} \in \mathrm{R}^{n_l}$ are the coefficients for the blur templates toward i^{th} direction.

To use the estimated motion information from the sparse representation to guide the particle sampling process, estimated motion information from different sources are integrated into the proposal distribution, which is a combination of the first-order Markov transition $p(\mathbf{x}_t \mid \mathbf{x}_{t-1})$, the second-order Markov transition $p(\mathbf{x}_t \mid \mathbf{x}_{t-1}, \mathbf{x}_{t-2})$, and $q_i(\mathbf{x}_t \mid \mathbf{x}_{t-1}, \mathbf{y}_{t-1})$ based on the blur motion estimation along direction θ_i.

3.2.3.3 Loss Function with Gradient Information

Incorporating blur templates into the appearance space allows for a more expressive appearance space to model blur degradations. However, with the augmented template space, ambiguity also increases, and some background might be well represented by some blur templates, especially when only grayscale information is used, as shown in Figure 16. In order to make the tracking algorithm more robust, based on the observation that though motion blur significantly changes the statistics of the gradients of the templates, the blur templates in the same direction have much more similar gradient histograms than blur templates of different directions, we propose to use the combination of the reconstruction error and a sum of weighted distances between the target candidate and each of the non-trivial templates as loss function.

For each template of $[\mathbf{T}_a, \mathbf{T}_b]$, we calculate its gradient histogram by letting each pixel vote for an gradient histogram channel, and get $\mathbf{D} = [\mathbf{d}_1, \mathbf{d}_2, \cdots, \mathbf{d}_{n_a+n_b}] \in \mathbb{R}^{h\times(n_a+n_b)}$, where h is the number of bins of the gradient histogram; and for the target candidate, we calculate its gradient histogram $\mathbf{g} \in \mathbb{R}^h$. Since we don't consider the trivial templates when calculating the sum of weighted distances, we let $\mathbf{d} = [\|\mathbf{d}_1 - \mathbf{g}\|_1, \|\mathbf{d}_2 - \mathbf{g}\|_1, \cdots, \|\mathbf{d}_{n_a+n_b} - \mathbf{g}\|_1, 0, \cdots, 0] \in \mathbb{R}^{(n_a+n_b+d)}$ indicate the distance between $\mathbf{g}$ and the gradient histogram of each element in $\mathbf{T}$. $\|\mathbf{dc}\|_2^2$ is used to measure the sum of the weighted distances, and

$$\frac{1}{2}\|\mathbf{Tc} - \mathbf{y}\|_2^2 + \beta\|\mathbf{dc}\|_2^2 \tag{57}$$

is used as the loss function.

3.2.3.4 Group Sparsity via ℓ_1+ℓ_1/ℓ_2 Mixed Norm

For the augmented template set with blur templates of different directions, since the motion blur of the target is always toward only one direction at time t, there is a natural group structure among the templates. The representation of the target candidate should not only be sparse, but also have group structure, i.e., the coefficients should also be sparse at the group level. In our tracking framework, we divide the templates into $n_g = n_\theta + d + 1$ groups $\mathbb{G} = \{G_1, G_2, \cdots, G_{n_\theta+d+1}\}$ using the following scheme: the normal templates are in one group; the blur templates in the same direction forms a group; and each trivial template is an individual group. In order to capture the group information among the templates and achieve sparsity at the same time, we employ a structured sparsity inducing norm which combines the ℓ_1 norm and a sum of ℓ_2 norms over groups of variables. The mixed norm is known as “sparse group Lasso".

Combining the loss function (57) and the $\ell_1 + \ell_1/\ell_2$ mixed norm results in the following non-smooth convex optimization problem:

$$\begin{aligned} &\min_{\mathbf{c}} \frac{1}{2}\|\mathbf{Tc} - \mathbf{y}\|_2^2 + \beta\|\mathbf{dc}\|_2^2 + \lambda_1\|\mathbf{c}\|_1 + \lambda_2\sum_{i=1}^{n_g}\left\|\mathbf{c}_{G_i}\right\|_2, \\ &\text{s.t.} \quad \mathbf{c}_T \geq 0 \end{aligned} \tag{58}$$

where $\mathbf{c}_{G_i}$ are coefficients associated with G_i.

Once (58) is solved, the observation likelihood can be derived from the reconstruction error of $\mathbf{y}$ as $p(\mathbf{y}_t \mid \mathbf{x}_t) \propto \exp\{-\alpha\|\mathbf{Tc} - \mathbf{y}\|_2^2\}$, where α is a constant used to control the shape of the Gaussian kernel.

3.2.3.5 Solve Eq. (58) by Accelerated Proximal Gradient

To solve the non-smooth convex optimization problem in Eq.(58), we adopt the accelerated proximal gradient method FISTA which has convergence rate of $O(\frac{1}{k^2})$, where k is the number of iterations. FISTA is designed for solving the following unconstrained optimization problem:

$$\min_{\mathbf{z}} F(\mathbf{z}) = f(\mathbf{z}) + g(\mathbf{z}) \tag{59}$$

where f is a smooth convex function with Lipschitz continuous gradient, and g is a continuous convex function which is possibly non-smooth.

In order to solve Eq.(58) with FISTA, we let $\mathbf{z} = [z_1, z_2, \cdots, z_{n_a+n_b+d}]^{\mathrm{T}}$ and make the substitution $\mathbf{c} = [z_1^2, z_2^2, \cdots, z_{n_a+n_b}^2, z_{n_a+n_b+1}, \cdots, z_{n_a+n_b+d}]^{\mathrm{T}}$, and solve the following optimization problem:

$$\min_{\mathbf{z}} \frac{1}{2}\|\mathbf{Tc}-\mathbf{y}\|_2^2 + \beta\|\mathbf{dc}\|_2^2 + \lambda_1\|\mathbf{z}\|_1 + \lambda_2\sum_{i=1}^{n_g}\|\mathbf{z}_{G_i}\|_2 \tag{60}$$

where $\mathbf{z}_{G_i}$ is associated with group G_i. Then, Eq.(60) can be re-expressed as Eq.(59), where $f(\mathbf{z}) = \frac{1}{2}\|\mathbf{Tc}-\mathbf{y}\|_2^2 + \beta\|\mathbf{dc}\|_2^2$ and $g(\mathbf{z}) = \lambda_1\|\mathbf{z}\|_1 + \lambda_2\sum_{i=1}^{n_g}\|\mathbf{z}_{G_i}\|_2$.

To develop a proximal gradient method, the following quadratic approximation of $F(\mathbf{z})$ at a given point $\mathbf{z}^{(k)}$ is considered, for $L > 0$

$$\begin{aligned} Q_L(\mathbf{z},\mathbf{z}^{(k)}) = \quad & f(\mathbf{z}^{(k)}) + \langle \mathbf{z}-\mathbf{z}^{(k)}, \nabla f(\mathbf{z}^{(k)})\rangle \\ & + \frac{L}{2}\|\mathbf{z}-\mathbf{z}^{(k)}\|_2^2 + g(\mathbf{z}) \end{aligned} \tag{61}$$

where $\nabla f(\mathbf{z}^{(k)})$ is the gradient function of $f(\cdot)$ at point $\mathbf{z}^{(k)}$.

Lemma 1 *Let f be a continuously differentiable function with Lipschitz continuous gradient and Lipschitz constant $L(f)$. Then, for any $L \ge L(f)$,*

$$F(\mathbf{z}) \le Q_L(\mathbf{z},\mathbf{z}^{(k)})$$

According to Lemma 1, given $L \ge L(f)$, a unique solution of $F(\mathbf{z})$ can be obtained by minimizing $Q_L(\mathbf{z},\mathbf{z}^{(k)})$,

$$p_L(\mathbf{z}^{(k)}) = argmin_{\mathbf{z}} \frac{1}{2}\|\mathbf{z}-\hat{\mathbf{z}}\| + \frac{1}{L}g(\mathbf{z}) \tag{62}$$

where $\hat{\mathbf{z}} = \mathbf{z}^{(k)} - \frac{1}{L}\nabla f(\mathbf{z}^{(k)})$, and

$$\nabla f(\mathbf{z}) = \mathrm{diag}(\mathbf{w})(\mathbf{T}^{\mathrm{T}}\mathbf{Tc} - \mathbf{T}^{\mathrm{T}}\mathbf{y}) + 2\beta\mathrm{diag}(\mathbf{w})\mathbf{d}^{\mathrm{T}}\mathbf{dc} \tag{63}$$

where $\mathbf{w} = [2z_1, 2z_2, \cdots, 2z_{n_a+n_b}, 1, \cdots, 1] \in \mathbb{R}^{n_a+n_b+d}$. Alg. 8 describes FISTA with backtracking.

```
Algorithms 8 FISTA with backtracking
Input: $L_0 > 0$, $\tau > 1$, $\mathbf{v}^{(1)} = \mathbf{z}^{(0)}$, $t_1 = 1$
1: for k=1,2,..., iterate until convergence do
```

```
set $L = L_{k-1}$,
while $F(p_L(\mathbf{v}^{(k)})) > Q_L(p_L(\mathbf{v}^{(k)}), \mathbf{v}^{(k)})$ do
    $L = \tau L$
end while
set $L_k = L$ and update
$\mathbf{z}^{(k)} = p_{L_k}(\mathbf{v}^{(k)})$,
$t_{k+1} = \frac{1+\sqrt{1+4t_k^2}}{2}$,
$\mathbf{v}^{(k+1)} = \mathbf{z}^{(k)} + (\frac{t_k - 1}{t_{k+1}})(\mathbf{z}^{(k)} - \mathbf{z}^{(k-1)})$
end for
```

A critical step is to solve Eq.(62) efficiently. Since the $\ell_1 + \ell_1/\ell_2$-norm is a special case of the tree structured group Lasso, Eq.(62) can be converted to

$$p_L(\mathbf{z}^{(k)}) = argmin_{\mathbf{z}} \frac{1}{2} \| \mathbf{z} - \hat{\mathbf{z}} \| + \sum_{i=0}^{m} \sum_{j=1}^{n_i} w_j^i \| \mathbf{z}_{G_j^i} \|_2 \qquad (64)$$

where m is the depth of the index tree, n_i is the number of groups at depth i, $w_j^i \geq 0 (i = 0,1,\cdots,m, j = 1,2,\cdots,n_i)$ is the pre-defined weight for group G_j^i. We apply the $\mathrm{MY}_{\mathrm{tgLasso}}$ algorithm to solve Eq.(64) efficiently. $\mathrm{MY}_{\mathrm{tgLasso}}$ algorithm maintains a working variable $\mathbf{u}$ initialized with $\hat{\mathbf{z}}$, then it traverses the index tree in the reverse breadth-first order to update $\mathbf{u}$ with $\mathbf{u}^i_{G_j^i} = \mathbf{u}^{i+1}_{G_j^i} \max(0, 1 - w_j^i / \left\| \mathbf{u}^{i+1}_{G_j^i} \right\|)$.

The time complexity of $\mathrm{MY}_{\mathrm{tgLasso}}$ algorithm is $O(mn)$, where n is the dimension of $\mathbf{z}$. After converting Eq.(62) to Eq.(64), the index tree has a constant depth 2, so the time complexity for solving Eq.(62) is $O(n)$, where $n = n_a + n_b + d$.

3.2.3.6 Template Update with Blur Detection

In order to capture the appearance variations of the target caused by illumination or pose changes, the template set needs to be updated during tracking. Since the appearance of the target is corrupted when heavy blur appears, updating the template set with heavily blurred target cannot capture the appearance changes of the target. So we propose to perform blur detection of the tracking result before updating the template set.

To detect blur, we investigate the response of both normal coefficients and blur coefficients obtained from solving the optimization problem Eq.(58). If the target is not blurred, the energy of the normal coefficients will be dominant. One criterion for updating the template set is $\frac{E(\mathbf{a})}{E(\mathbf{a}) + E(\mathbf{b})} > 0.9$, where $E(\cdot)$ represents the energy associated with the corresponding coefficients. Also, trivial templates are activated when the target cannot be well approximated by the template set. In order to avoid contaminating the template set, another criterion for updating template set is $\frac{E(\mathbf{e})}{E(\mathbf{a})+E(\mathbf{b})+E(\mathbf{e})} < 0.1$. When the target is not similar to any of the normal templates, and both of the above two criteria are satisfied, we replace the normal template having lowest response with the target template.

3.2.4 Cloud Implementation of Registration and Tracking

3.2.4.1 Application Component

Considering the nature of the tracking procedure, we divided the tracking application into three types of basic components: *registers*, *trackers* and *plotters*, described as follows.

- ***Registers*** carry the most computation task in the application, hence is the bottleneck of the performance. A straightforward way to improve the processing frame rate is to assign the registration calculation of multiple workers and merge their result together. These workers can calculate the H matrix simultaneously. Ideally, the more workers we use, the more speedup we can get from the computing power of the Cloud.

- A ***Tracker*** is dedicated to track a target in the video sequence. The computation of tracker task is pretty lightweight and we don't need to further distribute the tracker to multiple workers.

- The ***Plotter*** provides a web interface between the users and the tracking system. Users can see the real time tracking result through the plotter. It combines data sent by the register and tracker and plot the tail of the targets. See the "Web-based GUI" section for more details of this web interface.

Based on the above analysis, we implement a prototype in our Cloud environment. The prototype consists of a register, tracker and plotter, each of which runs on a separate virtual machine in the Cloud. In practical application, there will be large amount of video sequence and targets, each of these three VM can be easily duplicated to scale up the application's capability. We can also use container based virtualization technology such as OpenVZ to securely convert each VM to a container.

3.2.4.2 Synchronization

Since the register, tracker and plotter are distributed in the Cloud, the synchronization becomes a problem when the data from the register and tracker does not come in the same order. To counter this problem, the plotter will wait until all necessary data arrive before a new frame is produced and displayed to the user.

The communication between the three VMs uses TCP/IP socket for reliable data transmission.

3.2.4.3 Web-based GUI

The Web-based GUI is hosted as an HTTP server on the same VM as the plotter. To display the tracking result in real time, the HTTP server will need different web technologies. In our implementation, we choose HTML + PHP + JQuery to write the web GUI. HTML provides the framework of the webpage. PHP is used to pass parameters between the user and the server. JQuery plays the most important role to display the result in real time. Figure 10 shows the screenshot of the tracking result display through the web GUI.

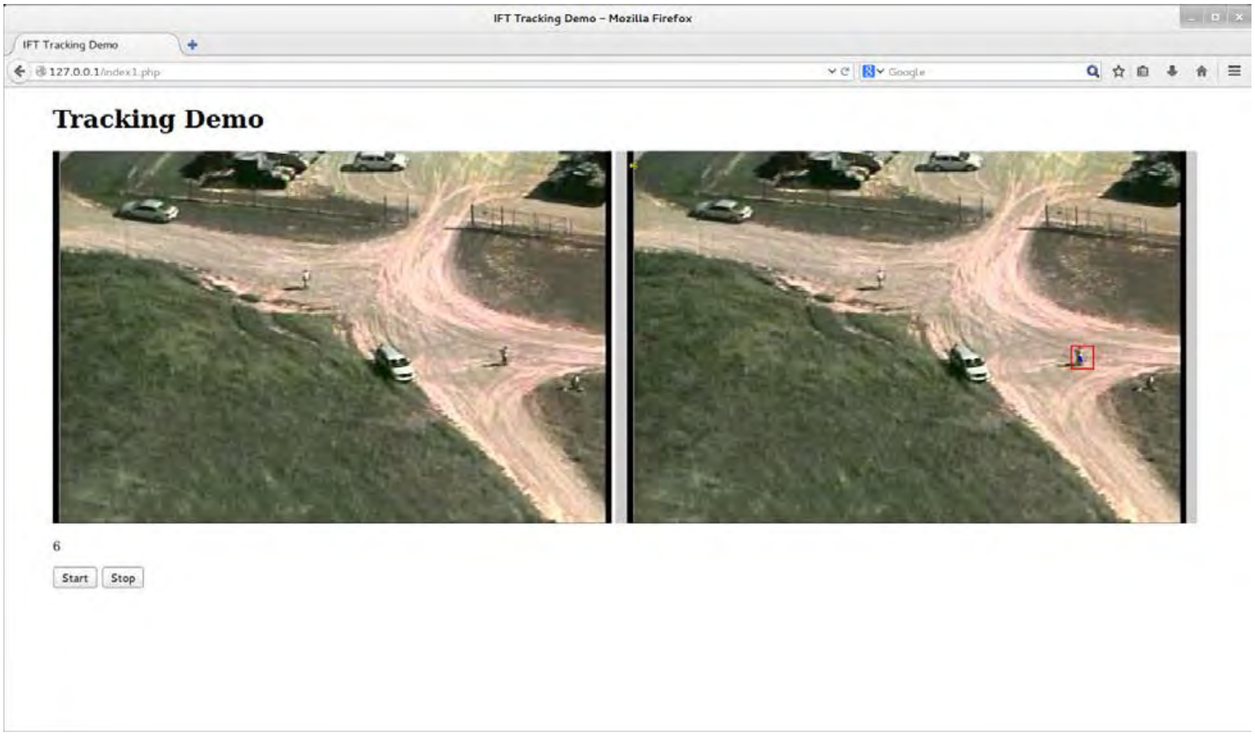


Figure 17: Web service based GUI showing the input video (left) and the tracking results (right).

3.2.5 Integration with GATER Framework and Kitware VsPlay

3.2.5.1 GATER Introduction

Government Algorithms for Tracking Exploitation Research (GATER) is a suite of data exploitation tools designed to support research and development for government purposes. These tools include front end processing, target detection, tracking, visualization, metrics, and various utilities such as tracking tuning tools. While GATER was first developed for image exploitation, much of the functionality will support any sensor type.

The overall objective of the GATER project is to develop software tools to provide an in-house data exploitation capability for both simple and complex problems. These tools must be open source and extensible for use by government employees and approved government contractors. As such, the software is developed with Government Purpose Rights (GPR) as the most restrictive data rights assertion. Furthermore, a DoD Community Source Agreement is tied to the software (see appendix for details), meaning that all development tied to GATER becomes the property of the DoD. This does not preclude individual contractors from asserting GPR for newly developed tools they deliver as part of GATER.

The following figure 18 illustrates the anatomy of a multiple target tracking system. While other architectures exist, the GATER tracker will utilize this basic framework. This approach supports an open architecture that allows various components to be interchanged for development and evaluation. The discussion that follows introduces each component.

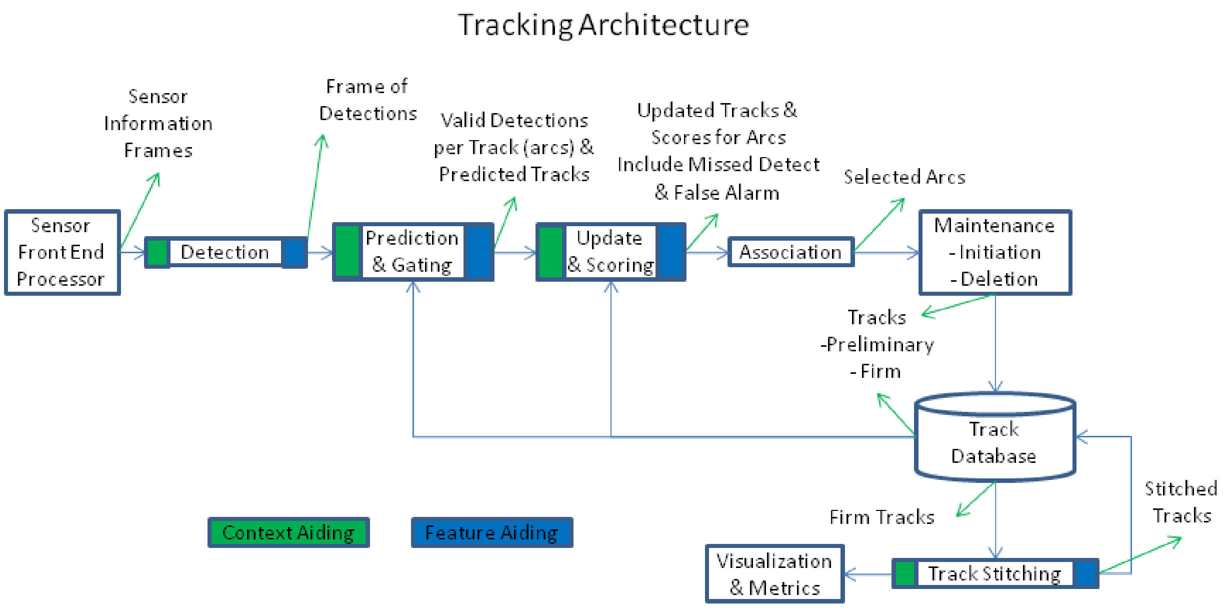
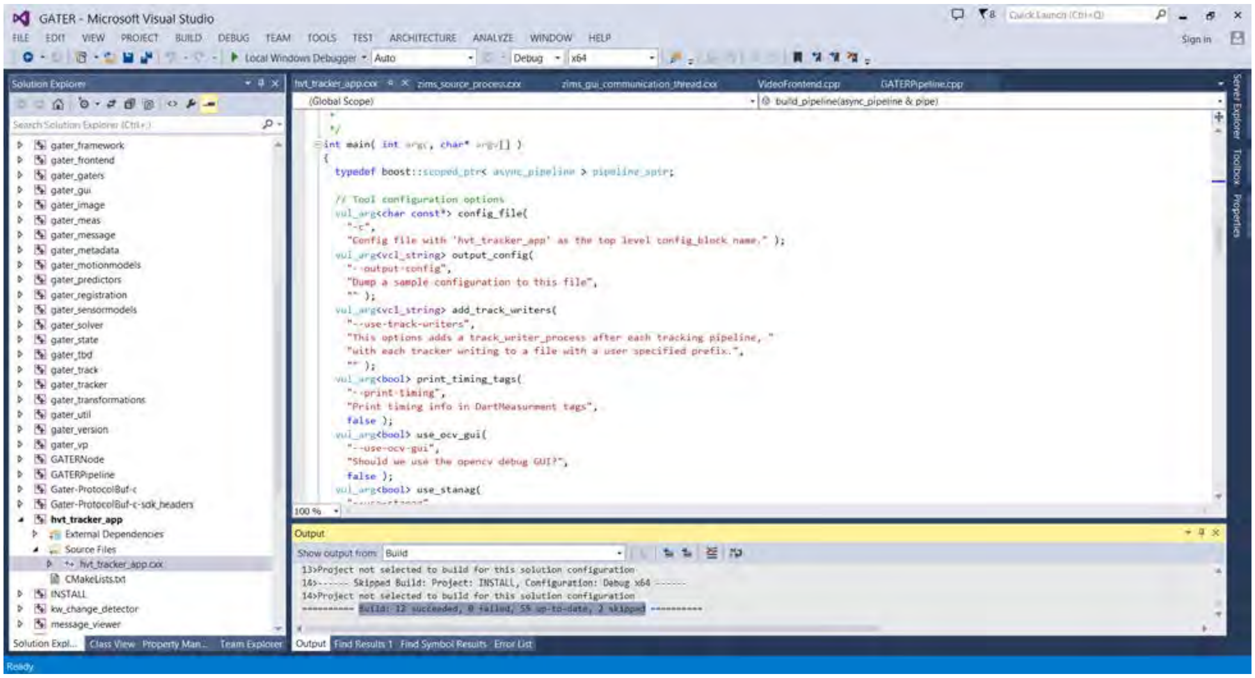


Figure 18: The system architecture of GATER

3.2.5.2 Building and Installing GATER in Windows

In this project, we install and build the GATER system in windows 8 64bit with Microsoft Visual Studio 2013. Figure 19 shows a successful build. The installation instruction is listed as follows.



```
 */
int main( int argc, char* argv[] )
{
  typedef boost::scoped_ptr< async_pipeline > pipeline_sptr;

  // Tool configuration options
  vul_arg<char const*> config_file(
    "-c",
    "Config file with 'hvt_tracker_app' as the top level config_block name." );
  vul_arg<vcl_string> output_config(
    "--output-config",
    "Dump a sample configuration to this file",
    "" );
  vul_arg<vcl_string> add_track_writers(
    "--use-track-writers",
    "This options adds a track_writer_process after each tracking pipeline, "
    "with each tracker writing to a file with a user specified prefix.",
    "" );
  vul_arg<bool> print_timing_tags(
    "--print-timing",
    "Print timing info in DartMeasurment tags",
    false );
  vul_arg<bool> use_ocv_gui(
    "--use-ocv-gui",
    "Should we use the opencv debug GUI?",
    false );
  vul_arg<bool> use_stanag(
```

```
13>Project not selected to build for this solution configuration
14>------ Skipped Build: Project: INSTALL, Configuration: Debug x64 ------
14>Project not selected to build for this solution configuration
========== Build: 12 succeeded, 0 failed, 55 up-to-date, 2 skipped ==========
```

Figure 19: The successful build of GATER in windows 8.1 with VC 2013.

<u>Building Boost for Windows</u>

Download Boost from http://sourceforge.net/projects/boost/files/boost/. After downloading Boost (we used version 1.55.0), extract it to a directory of your choice (e.g. C:\libraries\boost_1_55_0). Open a visual studio command prompt window and change to this directory.

Execute the following commands to build and install boost (commands are described in more detail below):

1. bootstrap.bat
2. bjam -j4 --toolset=msvc-10.0 --build-type=complete address-model=64 --prefix=PREFIX install

Description:

1. Builds the Boost Build engine called bjam. PREFIX should be replaced with the path to the directory where you want to install boost headers and shared libraries.
2. Builds 64-bit Boost static libraries. In order to build 32-bit libraries, change address-model=64 to address-model=32.
3. Installs the static libraries and headers to a directory of your choice. Make sure to replace PREFIX with the desired directory (e.g. bjam install -prefix=C:\boost). Note that the bjam install command allows multiple versions and variants of boost to be

installed without conflict.

4. Execute bjam --help or bjam --help--options to see additional options.

Building COIN Osi Clp for Windows

Download COIN Osi Clp from http://www.coin-or.org/download/source/Clp/. After downloading Clp, extract it and navigate to the Clp-<version>\Clp\MSVisualStudio\v10 directory. If a v10 directory does not exist, just copy the v9 directory and rename it to v10. Open Clp.sln in the v10 directory. (complete the Visual Studio Conversion Wizard if necessary)
Once the solution is opened in Visual Studio, make sure the build configuration is set to "Release" and "x64" ("Win32" for a 32-bit build) using the menu bar at the top. Right click Solution "Clp" and select Build Solution or hit F7. The compiled libraries and executables will be placed in "v10\Release". If you have any issues building the test libraries, disable building them through the build configuration settings.

See https://projects.coin-or.org/MSVisualStudio for additional compilation info.
Do the following to install COIN Osi Clp:

1. Save the following into a file e.g. "CopyLibs.bat" stored in the main Clp-<version> folder, then execute the file: (note: replace x64 with Win32 for 32-bit builds)
   ```
   xcopy Clp\MSVisualStudio\v10\x64\Release lib /exclude:Exclusions.txt
   ```
2. Save the following into a file e.g. "CopyHeaders.bat" stored in the main Clp-<version> folder:
   ```
   xcopy Osi\src include\coin /exclude:Exclusions.txt /Y
   xcopy Osi\src\Osi include\coin /exclude:Exclusions.txt /Y
   xcopy Clp\src include\coin /exclude:Exclusions.txt /Y
   xcopy Clp\src\OsiClp include\coin /exclude:Exclusions.txt /Y
   xcopy Buildtools\headers include\coin /exclude:Exclusions.txt /Y
   xcopy CoinUtils\src include\coin /exclude:Exclusions.txt /Y
   ```
3. Save the following into a file named "Exclusions.txt" stored in the main Clp-<version> folder:
   ```
   .obj
   .am
   .cpp
   .in
   .exe
   ```
4. Execute the two batch files. COIN-OSI is now installed.

Building FFmpeg for Windows

1. download:
   a. mingw64-w64 msys (http://sourceforge.net/projects/mingw-w64/files/External%20binary%20packages%20%28Win64%20hosted%29/MSYS%20%2832-bit%29/)
   b. c99 to c89
   c. msinttypes
   d. pkg-config-lite for msys

   e. yasm
2. Place makedef, c99wrap.exe, c99conv.exe, pkg-config.exe, and yasm.exe somewhere in your PATH.
3. Next, make sure inttypes.h (don't include stdint.h) and any other headers and libs you want to use are located in a spot that the compiler can see. Note that additional headers and libs are only necessary when building non-default ffmpeg features and are not used at this time.
   a. Do so by modifying the LIB and INCLUDE environment variables to include the Windows paths to these directories. Note that a semi-colon is used as the path delimiter.
      i. set INCLUDE=%INCLUDE%;<new path>
      ii. set LIB=%LIB%;<new path>
4. Don't forget to rename /msys/bin/link.exe to something different (e.g. link.exe_) to avoid shadowing msvc linker.
5. To set up a proper environment in MSYS, you need to run msys.bat from the Visual Studio or Intel Compiler command prompt. This will open a MINGW32 window.
6. Compile release and debug versions of the using the following commands in the MINGW32 window:
   a. ./configure --toolchain=msvc --prefix=build --extra-cflags="-MD" --extra-ldflags="-DEBUG"
   b. make –j <number of cores to use>
   c. make install
   d. make clean
   e. ./configure –toolchain=msvc –prefix=build2 --extra-cflags="-MDd" –extra-ldflags="-DEBUG"
   f. make –j <number of cores to use>
   g. make install
7. Rename all the libraries under the build2/lib directory such that their names end with a d (e.g. libavcodec.lib becomes libavcodecd.lib)
8. Copy all the libraries under the build2/lib directory to the build/lib directory
9. Copy the contents of the build directory to where you want FFMPEG installed
   a. If you are using the GATER_DEPENDENCIES_DIR, you would place the contents in to an ffmpeg folder in that directory

Building and installing GDAL for Windows

1. Open nmake.opt in top level GDAL directory and modify the following options:
   a. Set MSVC_VER to your corresponding MSVC version (e.g. 1700 for MSVC 11.0/2012)
   b. Uncomment the line reading "WIN64=YES"to compile for 64-bit
   c. Set GDAL_HOME to the directory where you want GDAL to be installed
2. Open a Visual Studio 2010 x64 command prompt

3. nmake /f makefile.vc
4. nmake /f makefile.vc devinstall

Building and installing gflags for Windows

1. Open the gflags Visual Studio solution file
2. If building for 64-bit, add an x64 configuration
    a. Open Configuration Manager
    b. Select new configuration and copy settings from Win32 configuration
    c. Ensure the appropriate Platform Toolset is selected
    d. Set the build configuration to “x64”
3. Configure the libgflags project to build a static library
4. Add the following preprocessor definitions: “GFLAGS_DLL_DECLARE_FLAG=” “GFLAGS_DLL_DEFINE_FLAG=” “GFLAGS_DLL_DECL=”
5. Ensure that the libgflags project uses the Multi-Threaded DLL and Multi-Threaded Debug DLL for runtime libraries for Release and Debug modes, respectively.
6. Build the libgflags project in Release and Debug modes

The CMake file will take care of moving the libraries and include files to an appropriate location

Building and installing glog for Windows

1. Open the google-glog Visual Studio solution file
2. If building for 64-bit, add an x64 configuration
    a. Open Configuration Manager
    b. Select new configuration and copy settings from Win32 configuration
    c. Ensure the appropriate Platform Toolset is selected
    d. Set the build configuration to “x64”
3. Open the project properties for the libglog_static project
    a. Add the path to the gflags include directory to the project’s “Additional Include Directories”
    b. Add the following preprocessor definitions: “GFLAGS_DLL_DECLARE_FLAG=” “GFLAGS_DLL_DEFINE_FLAG=” “GFLAGS_DLL_DECL=” “HAVE_LIB_GFLAGS”
4. Open the “config.h” file in the libglog_static project
    a. Comment out the line “#undef HAVE_LIB_GFLAGS” (line 22)
5. Open the “logging.h” file in the libglog_static project
    a. Change the line “#if 0” (line 88) before the line “#include <gflags/gflags.h>” to “#ifdef HAVE_LIB_GFLAGS”
6. Open the “logging.cc” file in the libglog_statis project
    a. Change the line “_asm int 3” (line 1442) to “__debugbreak();”
7. Ensure that the libglog_static project uses the Multi-Threaded DLL and Multi-Threaded Debug DLL for runtime libraries for Release and Debug modes, respectively.

8. Build the libglog_static project in Release and Debug modes

The CMake file will take care of moving the libraries and include files to an appropriate location

Building and installing gstreamer for Windows

1. Install with all optional components:
    a. gstreamer-sdk-devel-x86_64-<version>.msi
    b. gstreamer-sdk-x86_64-<version>.msi
2. Compile fluendo mpeg demuxer:
    a. Perform svn checkout of fluendo mpeg demuxer from https://core.fluendo.com/gstreamer/svn/trunk/gst-fluendo-mpegdemux
    b. Apply patch from https://subversion.vdl.afrl.af.mil/gater/branches/old_interface/MGS/FMVOT/3rdparty/gstreamer/gst-fluendo-mpegdemux.patch to top level gst-fluendo-mpegdemux directory. TortoiseSVN is capable of applying this patch.
    c. Compile Visual Studio project in gst-fluendo-mpegdemux/win32/vs10/libgstflumpegdemux_sdk.vcxproj
3. Verify install (optional):
    a. gst-inspect (in the gstreamer bin dir)
        i. verify that flutsdemux and ffdec_h264 elements are reported
        ii. if gstreamer fails to find any elements, make sure that all required dlls are available on the system PATH and that GST_PLUGIN_PATH is set correctly

Building and installing gtest for Windows

1. Open the gtest-md solution under the “msvc” directory
2. If building for 64-bit, add an x64 configuration
    a. Open Configuration Manager
    b. Select new configuration and copy settings from Win32 configuration
    c. Ensure the appropriate Platform Toolset is selected
    a. Set the build configuration to “x64”
3. If you’re building using Visual Studio 2012, add the following preprocessor definition to both projects for all configurations: “_VARIADIC_MAX=10”
4. Build the gtest and gtest_main projects for both Debug and Release configurations

Building and installing MGS for Windows

MGS is currently a required dependency of GATER. Use the following instructions to build MGS:

1. Obtain and if necessary build required dependencies. (Boost, GDAL, OpenCV)
2. Obtain and if necessary build optional dependencies.
3. Set up any necessary environment variables.

4. Open MGS directory containing CMakeLists.txt in CMake.
    a. The SVN url for this directory is https://subversion.vdl.afrl.af.mil/gater/trunk/MGS
5. Run configure.
6. Modify CMake variables as needed. Some useful variables are:
    a. WITH_GSTREAMER – enable if GStreamer support is needed.
    b. WITH_FFMPEG – enable if FFmpeg support is needed.
7. Run configure.
8. Fill in any remaining unpopulated fields if any and run configure again if necessary. Repeat this step until all necessary variables are populated.
9. Run generate.
10. Open the Visual Studio solution and build/install.

Building and installing OpenCV for Windows

1. Obtain the OpenCV installer for Windows
2. Open the installer to extract the files to the desired location
3. Delete the extracted build folder under the main opencv directory
4. Configure OpenCV with CMake
    a. Set the source code directory to the extracted opencv directory
    b. Set the build directory to a "build" directory in the extracted opencv folder
    c. Press the Configure button
    d. Allow the build directory to be created
    e. Select the appropriate generator for your version of Visual Studio
    f. Uncheck the WITH_DSHOW and WITH_FFMPEG options
    g. Check the WITH_TBB option
    h. Press the Configure button
    i. Set TBB_INCLUDE_DIRS to the location of the TBB include directory
    j. Set EIGEN_INCLUDE_PATH to the location of the eigen directory
    k. Press the Configure button
    l. Press the Generate button
5. Open the generated OpenCV.sln with Visual Studio
6. If you are using Visual Studio 2012
    a. Open the project properties for opencv_stitching
    b. Under Configuration Properties->C/C++/All Options add "/Zm130" to the end of the "Additional Options" field
7. Build the solution in both Release and Debug configurations

Building and installing protobuf for Windows

1. Open the protobuf Visual Studio solution file (Located under vsprojects)
2. If building for 64-bit, add an x64 configuration
    a. Open Configuration Manager

    b. Select new configuration and copy settings from Win32 configuration
    c. Ensure the appropriate Platform Toolset is selected
    d. Set the build configuration to "Release" and "x64"
3. Build the libprotobuf, libprotobuf-lite, libprotoc, and protoc projects

<u>Building and installing Qt for Windows</u>

1. Download Qt and install to desired location.
2. Open a Visual Studio 2010 x64 command prompt.
3. Edit your PATH to include your Qt bin directory.
4. Create an environment variable called QTDIR pointing to your Qt top level directory.
5. Navigate to your Qt top level directory and issue the following commands:
    a. configure -debug-and-release -no-qt3support -no-webkit -opensource
    b. Build using either jom (recommended) or nmake
        i. Using jom
            1. Unzip jom.zip to your top level Qt directory
            2. jom.exe –j <cores_to_use> (e.g. "jom.exe –j 4")
        ii. Using NMake
            1. nmake

<u>Building and installing ZMQ for Windows</u>

1. Unzip the ZMQ source code
2. Open the appropriate solution file in the builds/msvc folder
3. Build Debug and Release versions of ZMQ

3.2.5.3 Kitware VsPlay

The vsPlay user interface is designed for full motion video exploitation. There are three primary modules: manual event identification, change detection and moving target tracker. The change detection, tracking, and associated database functions are provided as part of a system capability. This document will focus on the capabilities of vsPlay as the primary user interface. In addition, there are exploitation aides such as tripwire, selector, and filtering. Basic FMV exploitation functions such as pause, play, fast-forward, rewind are also available. For general exploitation functions there are capabilities to measure distance, magnify and change polarity.

The screen layout (Fig. 20) for vsPlay includes a row of tabs, which contain drop down menu functions at the top of the screen. These drop down menus include a number of icon functions and provide additional capabilities for the user to interact with the screen display layout and video based upon tab selection. To provide maximum situational awareness of activity occurring within the FMV feed an analyst should set their screen layout to display the tracks pane (by selecting the show track list from the drop down menu), events pane (by selecting the show events list from the drop down menu), and the change detections pane (by selecting the

Descriptors tab and ensuring the show alert list is activated). Each screen layout tab is described below:

- Video Tab - Controls the FMV feed such as start, stop, decrease speed;
- Tracks Tab – Provides on screen displays for the MTT such as track ID’s, entity bounding boxes, PVO scores;
- Events Tab – Displays events that correlate with the FMV feed such as show all person/vehicle events;
- Descriptors Tab – Works with alerts to activate/deactivate or shows alerts;
- Regions Tab – Supports analyst ability to create/select/de-select or display regions of interest for activity or non-activity within the FMV feed;
- Tools Tab – Provides report generation, measuring/ruler, display change detection list functions.

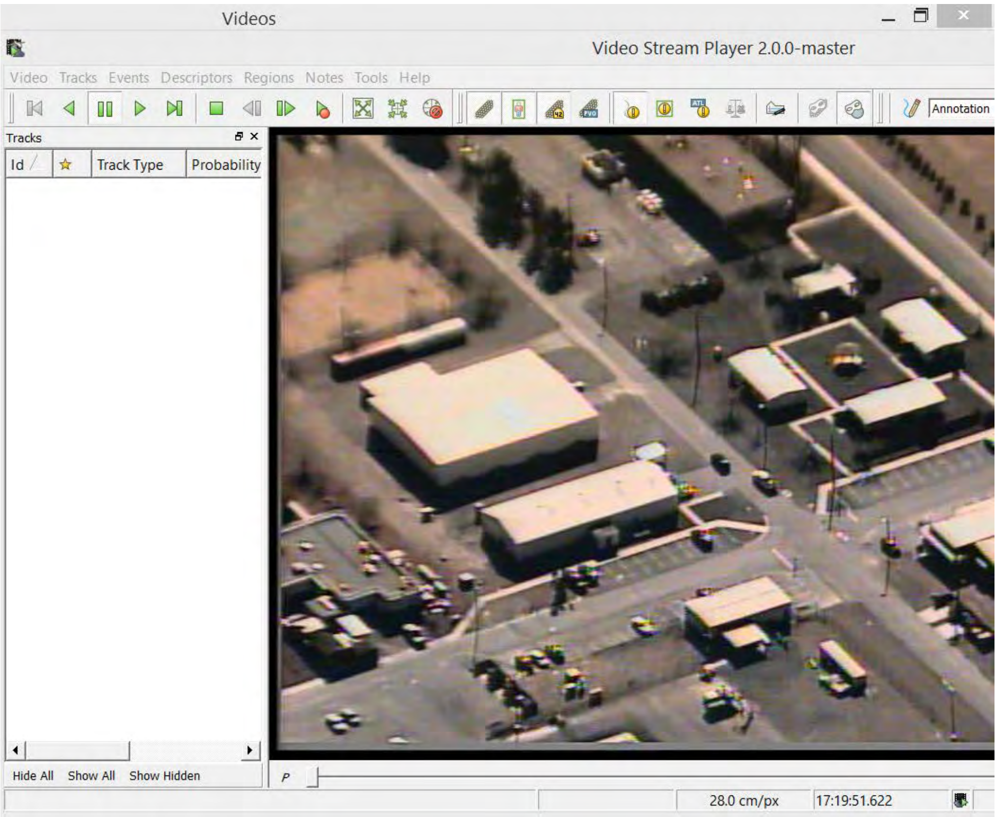


Figure 20: vsPlay screen layout

3.2.5.4 Integration with GATER and Kitware VsPlay

To integrate with GATER and vsPlay, we designed the system flowchart in Fig. 21.

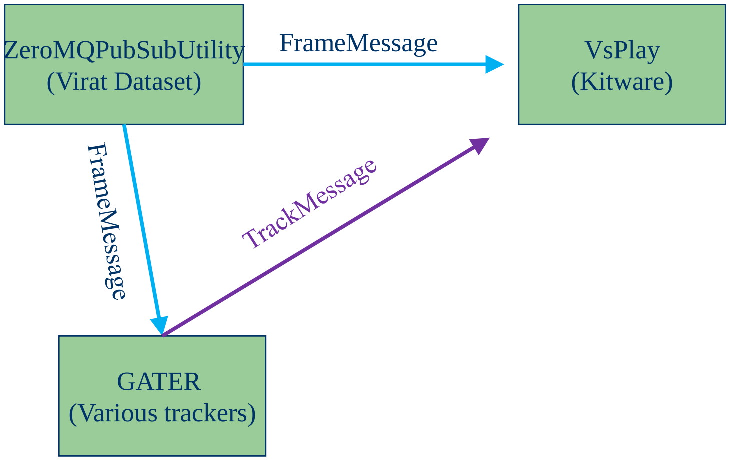


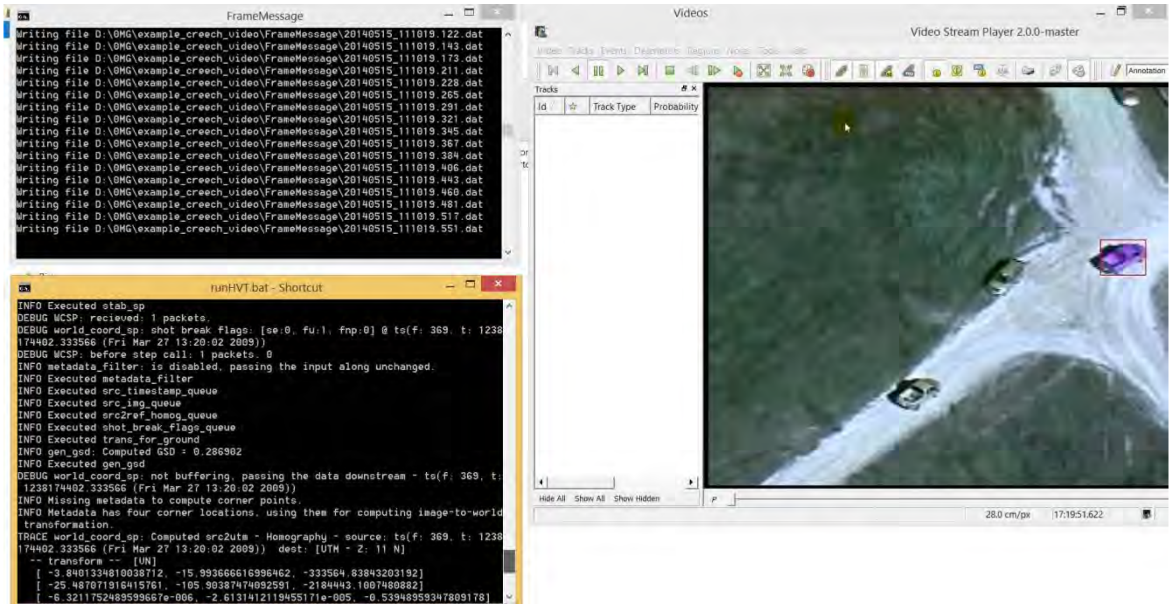
Figure 21: A flowchart for the integration with GATER and VsPlay

There are three major parts:

1. ZeroMQPubSubUtility will send the video data (virat dataset) to GATER and VsPlay via the zeromq and protobuf
2. GATER with various trackers will process the frame messages and run the pipeline. The outputs (track of GATER will send to VsPlay.
3. The Kitware VsPlay will receive both and frame messages and track messages. The video and tracking results will be displayed in VsPlay.

Figure 22 shows the running results of the integration. The left upper corn shows that the ZEROMQPUBSUBUtility is broadcasting the Kitware VIRAT dataset. The left bottom corn shows the GATER pipeline is running to process the video data. The tracker will be called by GATER pipeline, whose setup is specified by XML files:

```
<param>
      <include>param/detector.xml</include>
      <include>param/meas.xml</include>
      <include>param/frame.xml</include>
      <include>param/tracker.xml</include>
      <include>param/sequential_registration.xml</include>
      <include>param/nonsequential_registration.xml</include>
      <include>param/frontend.xml</include>
      <include>param/backend.xml</include>
      <include>param/tbd.xml</include>
      <include>param/gater.xml</include>
      <!-- sequence of modules to run -->
      <moduleSequence>VideoFrontend,SequentialFrameRegistrationFREAK,No
nsequentialFrameRegistrationFREAK,VideoDetector,SHT,FrameProtoBackend
      </moduleSequence>
</param>
```



```
Writing file D:\OMG\example_creech_video\FrameMessage\20140515_111019.122.dat
Writing file D:\OMG\example_creech_video\FrameMessage\20140515_111019.143.dat
Writing file D:\OMG\example_creech_video\FrameMessage\20140515_111019.173.dat
Writing file D:\OMG\example_creech_video\FrameMessage\20140515_111019.211.dat
Writing file D:\OMG\example_creech_video\FrameMessage\20140515_111019.228.dat
Writing file D:\OMG\example_creech_video\FrameMessage\20140515_111019.265.dat
Writing file D:\OMG\example_creech_video\FrameMessage\20140515_111019.291.dat
Writing file D:\OMG\example_creech_video\FrameMessage\20140515_111019.321.dat
Writing file D:\OMG\example_creech_video\FrameMessage\20140515_111019.345.dat
Writing file D:\OMG\example_creech_video\FrameMessage\20140515_111019.367.dat
Writing file D:\OMG\example_creech_video\FrameMessage\20140515_111019.384.dat
Writing file D:\OMG\example_creech_video\FrameMessage\20140515_111019.406.dat
Writing file D:\OMG\example_creech_video\FrameMessage\20140515_111019.443.dat
Writing file D:\OMG\example_creech_video\FrameMessage\20140515_111019.460.dat
Writing file D:\OMG\example_creech_video\FrameMessage\20140515_111019.481.dat
Writing file D:\OMG\example_creech_video\FrameMessage\20140515_111019.517.dat
Writing file D:\OMG\example_creech_video\FrameMessage\20140515_111019.551.dat
```

```
INFO Executed stab_sp
DEBUG WCSP: recieved: 1 packets.
DEBUG world_coord_sp: shot break flags: [se:0, fu:1, fnp:0] @ ts(f: 369, t: 1238
174402.333566 (Fri Mar 27 13:20:02 2009))
DEBUG WCSP: before step call: 1 packets. 0
INFO metadata_filter: is disabled, passing the input along unchanged.
INFO Executed metadata_filter
INFO Executed src_timestamp_queue
INFO Executed src_img_queue
INFO Executed src2ref_homog_queue
INFO Executed shot_break_flags_queue
INFO Executed trans_for_ground
INFO gen_gsd: Computed GSD = 0.286902
INFO Executed gen_gsd
DEBUG world_coord_sp: not buffering, passing the data downstream - ts(f: 369, t:
 1238174402.333566 (Fri Mar 27 13:20:02 2009))
INFO Missing metadata to compute corner points.
INFO Metadata has four corner locations, using them for computing image-to-world
 transformation.
TRACE world_coord_sp: Computed src2utm - Homography - source: ts(f: 369, t: 1238
174402.333566 (Fri Mar 27 13:20:02 2009))  dest: [UTM - Z: 11 N]
  -- transform --  [UN]
   [ -3.8401334810038712, -15.993666616996462, -333564.83843203192]
   [ -25.487071916415761, -105.90387474092591, -2184443.1007480882]
   [ -6.3211752489599667e-006, -2.6131412119455171e-005, -0.53948959347809178]
```

Figure 22: The screen shot of a demo run with Virat data, GATER, and VsPlay

3.3 Hard and Soft Data Fusion

3.3.1 CMU Sphinx for Speech Recognition

3.3.1.1 CMU Sphinx Setup

CMU Sphix is a complete state-of-the-art hidden Markov model (HMM) based open source speech recognition system. Designed at Carnegie Mellon University, SPHINX is one of the best and most versatile recognition systems in the world today. An HMM-based system, like all other speech recognition systems, functions by first learning the characteristics (or parameters) of a set of sound units, and then using what it has learned about the units to find the most probable sequence of sound units for a given speech signal. The process of learning about the sound units is called training. The process of using the knowledge acquired to deduce the most probable sequence of units in a given signal is called decoding, or simply recognition.

Accordingly, we will need those components of the SPHINX system that we can use for training and for recognition. In other words, we will need the *SPHINX trainer* and a *SPHINX decoder*. In this resarch period, we set up and tested the speech recogintion system. Some primitive results are obtained.

CMU Sphinx toolkit has a number of packages for different tasks and applications. It's sometimes confusing what to choose. To cleanup, here is the list

- Pocketsphinx — recognizer library written in C.
- Sphinxtrain — acoustic model training tools
- Sphinxbase — support library required by Pocketsphinx and Sphinxtrain

- Sphinx4 — adjustable, modifiable recognizer written in Java
- CMUclmtk — language model tools

In this project, we will use Sphinxbase, Sphinxtrain, and Pocketsphinx. We downloaded latest available releases from the following links:

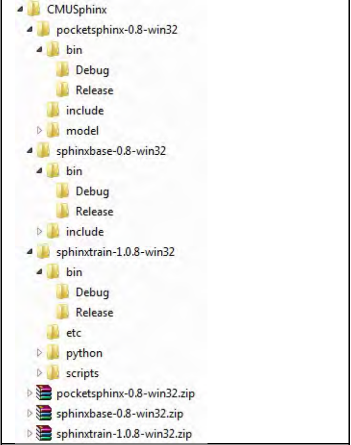
http://sourceforge.net/projects/cmusphinx/files/sphinxbase/0.8/sphinxbase-0.8-win32.zip/download

http://sourceforge.net/projects/cmusphinx/files/pocketsphinx/0.8/pocketsphinx-0.8-win32.zip/download

http://sourceforge.net/projects/cmusphinx/files/sphinxtrain/1.0.8/sphinxtrain-1.0.8-win32.zip/download

The setup and test prodecures are:

- Make subdirectory CMUSphinx, for an example, C:\Users\dshen\Desktop\CMUSphinx
- Decompress the three downloaded zips into seperated folders. The folder struct of CMUSphinx is shown in Figure 23**Error! Reference source not found.**.



```
CMUSphinx
  pocketsphinx-0.8-win32
    bin
      Debug
      Release
    include
    model
  sphinxbase-0.8-win32
    bin
      Debug
      Release
    include
  sphinxtrain-1.0.8-win32
    bin
      Debug
      Release
    etc
    python
    scripts
  pocketsphinx-0.8-win32.zip
  sphinxbase-0.8-win32.zip
  sphinxtrain-1.0.8-win32.zip
```

Figure 23: The folder structure of CUMSphinx

- Running pocketsphnix to obtain transcrift from an audio file
  - Generate or record a message [we selected the following message: "*1 white pick-up truck, turns left, travels north, center of screen*"], the audio file can be downloaded from https://www.dropbox.com/s/j6x7g0vk8nvblva/chat1.wav]
  - Copy (or download and copy) "chat1.wav" to "C:\Users\Dan\Desktop\V2T\CMUSphinx\pocketsphinx-0.8-win32\bin\Release"
  - In the same folder, create a new text file and rename it "argFile.txt", add the following lines to the file (this step specify the HMM, language model, and dictionary):
    - -hmm ../../model/hmm/en_US/hub4wsj_sc_8k
    - -lm ../../model/lm/en_US/hub4.5000.DMP
    - -dict ../../model/lm/en_US/cmu07a.dic
  - In the same folder, create a new text file and rename it "ctlFile.txt", add the following line to the file (this step specify the audio file to be processed):
    - chat1
  - Create a bat file, runme.bat, and add the following file to the file:
    - pocketsphinx_batch.exe -argfile argFile.txt -cepdir ./ -ctl ctlFile.txt -cepext .wav -adcin true -hyp out.txt
  - Run the runme.bat and will get a out.txt file in the same folder. In this example, it shows:
    - "*one like pick up truck turns left troubles north center of screen* (chat1 -

62894)”

It seems the Sphix recoginze ‘white’ as ‘like’. Therefore we need to perform trainning to calibrate the speech recogination toolkit.

3.3.1.2 Training Acoustic Model for CMU Sphinx

CMUSphinx project comes with several high-quality acoustic models. There are US English acoustic models for microphone and broadcast speech as well as a model for speech over a telephone. Most command-and-control apps could use them directly as well as large vocabulary applications.

Besides models, CMUSphinx provides ways for adaptation which is sufficient for most cases when more accuracy is required. Adaptation is known to work well when you are using different recording environments (close-distance or far microphone or telephone channel), or when a slightly different accent is present (UK English or even Indian English) or even another language. Adaptation, for example, works well if you need to quickly add support for some new language just by mapping acoustic model phoneset to target phoneset with the dictionary.

There are, however, applications where the current models won't work. For example, the exmaple in section 3.3.1.1 In these cases, we need to train our own model.

The trainer learns the parameters of the models of the sound units using a set of sample speech signals. This is called a training database. The database contains information required to extract statistics from the speech in form of the acoustic model. The trainer needs to be told which sound units you want it to learn the parameters of, and at least the sequence in which they occur in every speech signal in your training database. This information is provided to the trainer through a file called the transcript file, in which the sequence of words and non-speech sounds are written exactly as they occurred in a speech signal, followed by a tag which can be used to associate this sequence with the corresponding speech signal.

The trainer then looks into a dictionary which maps every word to a sequence of sound units, to derive the sequence of sound units associated with each signal. Thus, in addition to the speech signals, you will also be given a set of transcripts for the database (in a single file) and two dictionaries, one in which legitimate words in the language are mapped sequences of sound units (or sub-word units), and another in which non-speech sounds are mapped to corresponding non-speech or speech-like sound units. We will refer to the former as the language dictionary and the latter as the filler dictionary.

After training, it's mandatory to run the decoder to check training results. The Decoder takes a model, tests part of the database and reference transcriptions and estimates the quality (WER) of the model. During the testing stage we use the language model with the description of the order of words in the language.

Database should have enough speakers recording, variety of recording conditions, enough acoustic variations and all possible linguistic sentences. The size of the database depends on the

complexity of the task you want to handle as mentioned above. A Database should have the two parts mentioned above - training part and test part. Usually test part is about 1/10th of the full data size, but we don't recommend you to have test data more than 4 hours of recordings.
The file structure for the database is:

- etc
  - your_db.dic - Phonetic dictionary
  - your_db.phone - Phoneset file
  - your_db.lm.DMP - Language model
  - your_db.filler - List of fillers
  - your_db_train.fileids - List of files for training
  - your_db_train.transcription - Transcription for training
  - your_db_test.fileids - List of files for testing
  - your_db_test.transcription - Transcription for testing
- wav
  - speaker_1
    - file_1.wav - Recording of speech utterance
  - speaker_2
    - file_2.wav

It's critical to have audio files in a specific format. Sphinxtrain does support some variety of sample rates but by default it's configured to train from 16khz 16bit mono files in MS WAV format.

To start the training change to the database folder and run the following commands:

```
python ../sphinxtrain/scripts/sphinxtrain -t T1 setup
```

where “T1” is the task name.

This will copy all the required configuration files into etc subfolder of your database folder and prepare database for training, the structure will be:

etc
feat
logdir
model_parameters
 model_architecture
wav

After setup step only two folders of the above will be present, others will be created during the training process:

etc
wav

After that, we need to edit the configuration files in etc folder, there are many variables but to get started we need to change only a few.

To start the training, run “python ../sphinxtrain/scripts/sphinxtrain run”

3.3.2 Text Matching Based on Big Data Analysis

Understanding open-end simple natural language requires huge amount of knowledge and a variety of reasoning skills. Natural language data typically include text [101]. For text classification, Information Extraction (IE) is applied from a chat message (i.e., microtext) or a document (e.g., using Sphinx or Apache NLP) as an automated approach [102]. Previous work in natural language full-text searching has demonstrated that significant improvements in model accuracy are possible by leveraging feature relations [103], including when modeling microtext. For analysts, the goal is to apply text analytics to provide semantic indications and warnings to video data [104].

Pattern matching in natural language are widely used in information fusion. Current trends in data fusion include machine analytics for big data [105], use of pattern matching for cloud computing applications of simultaneous target tracking and classification [106], and robotics control [107]. Techniques for big data analysis are needed imaging, text and cyber analysis which includes scalable and elastic learning methods.

When the datasets are large, some information fusion algorithms might not scale up well. For example, if an algorithm needs to load data into memory constantly, the program may run out of memory for large datasets. One promising approach is to utilize and adapt MapReduce for some machine learning technologies to resolve these large-scale problems. Apache Mahout is a machine learning library for clustering, classification and filtering, implemented on top of Hadoop, the open source version of MapReduce. Although there are some machine learning algorithms implemented in Mahout, it is still helpful to study how to convert a machine learning algorithm to a Hadoop program and to optimize the algorithm scalability in large datasets.

3.3.2.1 MapReduce

The MapReduce framework has been used to process large datasets since the original paper [106] was published. Google’s clusters process more than 20 Petabytes of data every day by running one hundred thousand MapReduce jobs on average. Using this framework, programmers only need to focuson problem solving versus implementation. The MapReduce runtime system will take care of the underlining parallelization, fault tolerance, data distribution and load balance. Google file system (GFS) is a distributed file system that MapReduce uses for the storage of large amount of data across inexpensive hard drives. The availability and reliability of underlining unreliable hardware are provided by replicating file blocks and distributing them across different nodes.

A MapReduce job consists of at least a map function and a reduce function, called mapper and reducer respectively. The mapper takes as input a pair of key/value and produces a set of key/value pairs. All key/value pairs are sorted by their keys and sent to different reducers, indexed by the key. Each reducer receives a key and a set of values that has the same key. This makes MapReduce an excellent tool for computations that need sorting or counting. The map and reduce functions are left to the user to implement their desired functionalities to process each key/value pair. Hadoop2 is an open source implementation of the MapReduce framework that is commonly used by academic and industry for Big Data analysis. In the core of Hadoop are Hadoop MapReduce and Hadoop Distributed File System (HDFS), the open source counterpart of GFS. There are also a bundle of Hadoop-related projects supported by Apache Foundation, such as HBase (database), Hive (data warehouse), Pig (highlevel data-flow), Zookeeper (high-performance coordination) and Mahout (scalable machine learning and data mining). Therefore, we choose Hadoop as the develop platform to study the scalability of Naive Bayes classifier.

3.3.2.2 System Components

As shown in the Figure 24, the system adds four modules on top of Hadoop: the work flow controller (WFC), the data parser, the user terminal and the result collector. This system is designed based on the need to generate different size of datasets and test the Hadoop program on them respectively. We also need to perform ten-fold cross validation for accuracy computation that requires calling the same program multiple times. The raw data comes from a microtext from an analyst. The data parser is responsible to produce the desired data format to assist the program to efficiently process each word. The user submits jobs through the user terminal. Experiment results are also accessible through the user terminal after the result collector finishes collection.

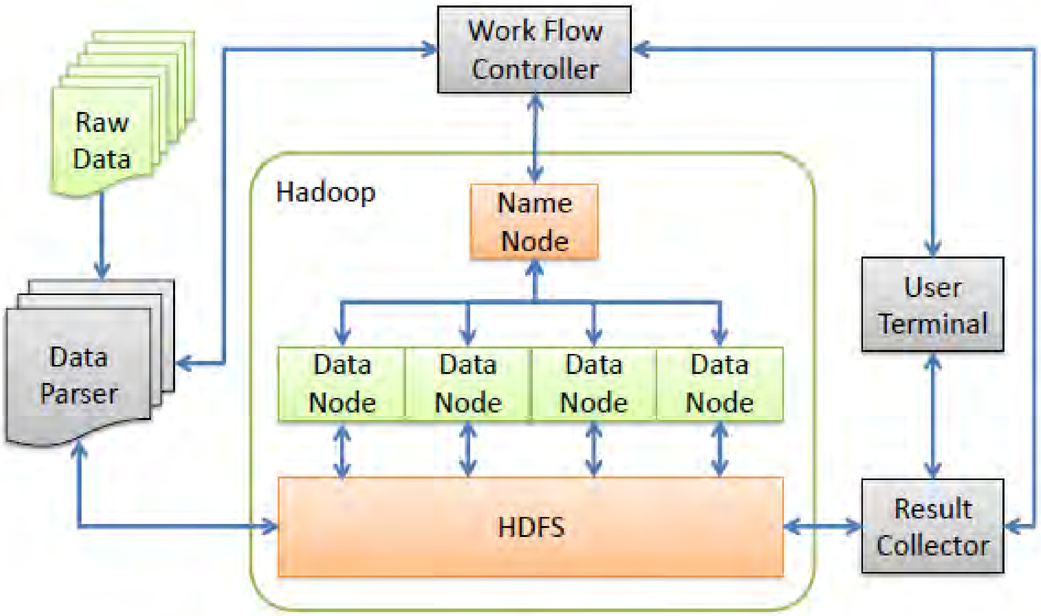


Figure 24: A system of process data using Pattern matching on Hadoop.

The work flow controller manages the work flow of the whole system, which includes:

1) Instruct data parser of the format of input data and the desired output;
2) Transmit source code to the name node and execute Hadoop jobs;
3) Trigger the result collector to collect computing results once they are available on Hadoop Distributed File System (HDFS)

3.3.2.3 Overall Work Flow

1) Preprocessing: The data parser first preprocesses all messages or sentences into a common format. After the processing, each message or sentence is one component in the dataset, with document ID prefixed. This is useful because by default MapReduce splits the input files by text into a mapper. To pre-process a raw sentence, unwanted context such as punctuation, special symbols and numbers is deleted. A lexicon or vocabulary is implemented to filter out meaningless words. Several prefixed key words tags were applied in the system: “red” and “white” refer to “color”, “turn” and “accelerate” refer to “behavior”, “left” and “right” refer to “direction”, “north” and “south” refer to “orientation”, “car” and “truck” refer to “object”. All pre-processed text are stored in the name node as a repository, waiting for further sampling.

2) Preparing Input Datasets: The WFC and the data parser work together to prepare input datasets for all test trials. When the WFC requests a dataset with certain size, the data parser extracts from the repository the desired number of each key word. The result is an input dataset of several equal size classes of key words. After a dataset is generated, the data parser divides it into 10 subsets for the convenience of ten-fold cross validation. The WFC then moves them to the right locations in HDFS for every trial and calls the Hadoop matching program.

3) Pattern Matching and Extracting using Hadoop: The pattern matching is the key step in the work flow. Figure 10 shows the job sequence of this step. Once the vocabulary and message data are ready in HDFS, the matching job combines the vocabulary and the message data, resulting message IDs and keyword pairs. Finally, the pattern extracting job sorts the messages with the pre-processed document ID according to the key words matching.

4) Result Collection: After the extracting job finished, the result collector retrieves the vocabulary, intermediate table, matching results and statistics of the pattern from HDFS.

3.3.2.4 Automatic Scheduling

The WFC coordinates the automation of the whole system. All test messages are automatically scheduled by the WFC without supervision. This automatic scheduling method can be easily applied to other programs with minor change of the parameters. The cloud computing framework for the text processing is shown in Fig. 25.

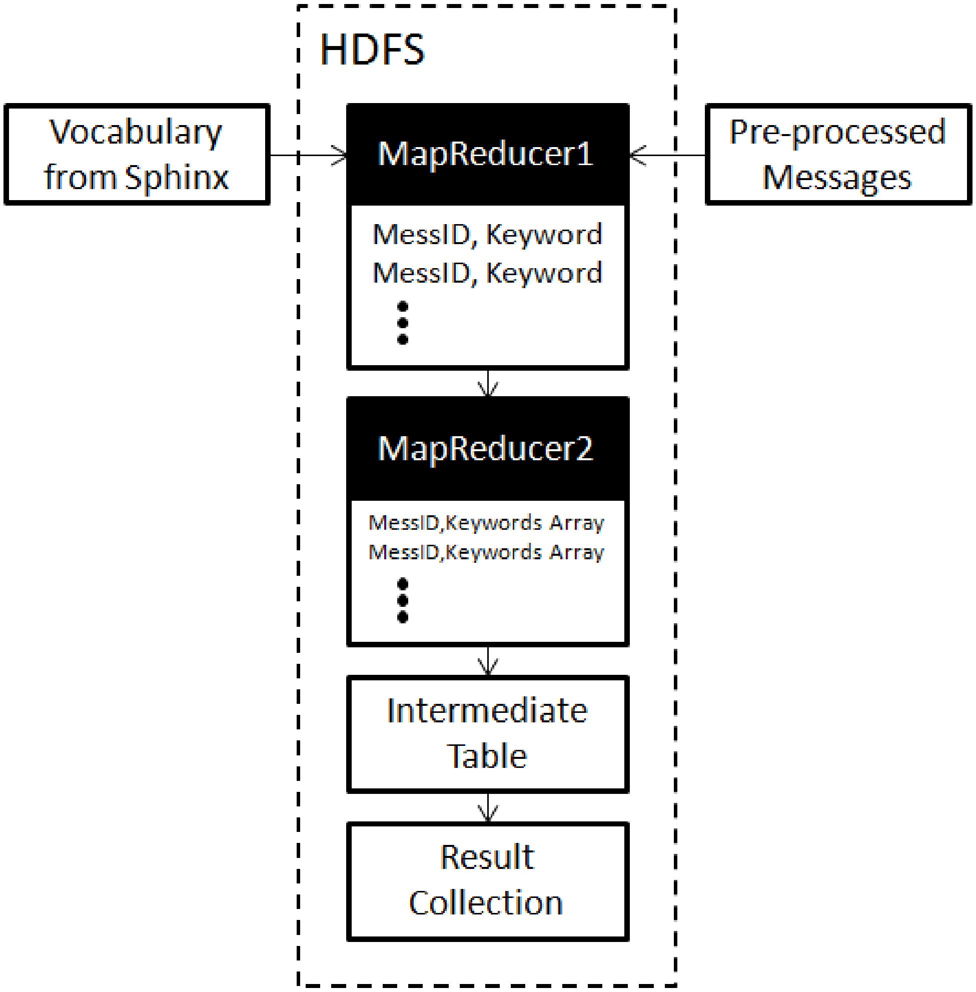


Figure 25: Job sequence of Pattern Matching on Hadoop

3.3.3 Hard-Soft Information Fusion

It is not unusual that a tracker loses its target and assigns a new label to the newly established track of the same moving target. It happens more frequently when a video sequence consists of many discontinue shots of the same moving target (for example, zoom in/out), because the trackers do not have enough information to associate the newly detected target to one of the previously tracked targets. Fortunately, for reviewed video, soft information in terms of chat messages are usually available which may contain valuable information for connecting tracklets of the same moving target. By fusing the soft data (chat messages) into the hard data (video tracks), it is possible to link the originally separated tracks into a single track with unique label for each moving target. To tackle this problem, we propose a fusion scheme as shown in Figure 26.

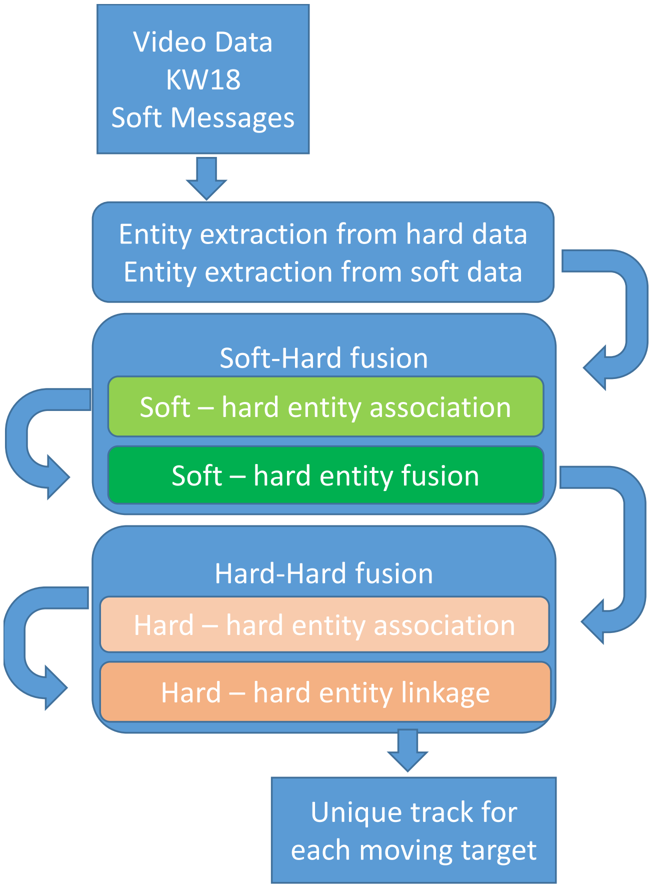

Video Data
KW18
Soft Messages
Entity extraction from hard data
Entity extraction from soft data
Soft-Hard fusion
Soft – hard entity association
Soft – hard entity fusion
Hard-Hard fusion
Hard – hard entity association
Hard – hard entity linkage
Unique track for each moving target

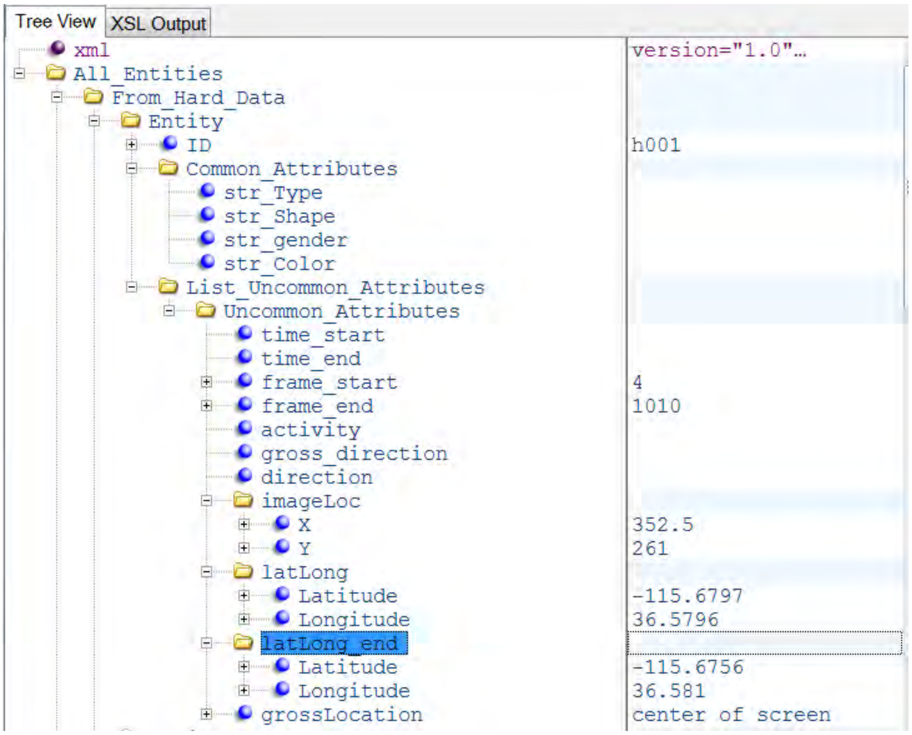
Figure 26: Workflow of the proposed fusion scheme for combining differently labeled tracklets of the same moving target into a single long-duration track with unique track ID

In Figure 26, the workflow takes the raw video sequence, tracking results in KW18 format and chat messages as its input. The first step is to construct entities from both the hard and soft data. Once required information is extracted and entities are constructed, the next module will perform hard – soft entity association as well as fusion. The purpose of this module is to link each entity constructed from soft data with the entity constructed from its corresponding hard data and fuse the linked entities to generate entities with more complete information. The next module, hard – hard entity association is devised to associate each hard entity with a nearby hard entity which is supposed to be constructed from the closest and later track of the same moving target. The last module, hard – hard entity linkage, links the associated hard entities sequentially and aims at producing a single, long duration track for each moving target.

In our implementation, each entity corresponds to one tracklet with a unique track ID. Following [109], each entity consists of two sets of attributes: common attributes and uncommon attributes. Common attributes are those which will not change over the lifetime of a target track like type and color of the target. Uncommon attributes are those changing over time like target location, direction, and activity. The same sets of attribute definitions are used for entities constructed from both hard and soft data. Details of each module are described in the following sections.

3.3.3.1 Constructing entity from hard data

Constructing entities from hard data is straightforward. No common attribute is obtained from hard data as such information is not contained in KW18 format. Figure 27 shows a sample entity constructed from the track with ID = 1 from Creech data. Note that the uncommon attribute grossLocation is derived from imageLoc by dividing each image frame into nine equal portions as described in [109].



```
xml                                         version="1.0"…
All_Entities
  From_Hard_Data
    Entity
      ID                                    h001
      Common_Attributes
        str_Type
        str_Shape
        str_gender
        str_Color
      List_Uncommon_Attributes
        Uncommon_Attributes
          time_start
          time_end
          frame_start                       4
          frame_end                         1010
          activity
          gross_direction
          direction
          imageLoc
            X                               352.5
            Y                               261
          latLong
            Latitude                        -115.6797
            Longitude                       36.5796
          latLong_end
            Latitude                        -115.6756
            Longitude                       36.581
          grossLocation                     center of screen
```

Figure 27: A sample entity constructed from the track with ID = 1 from Creech data set.

3.3.3.2 Constructing entity from soft data

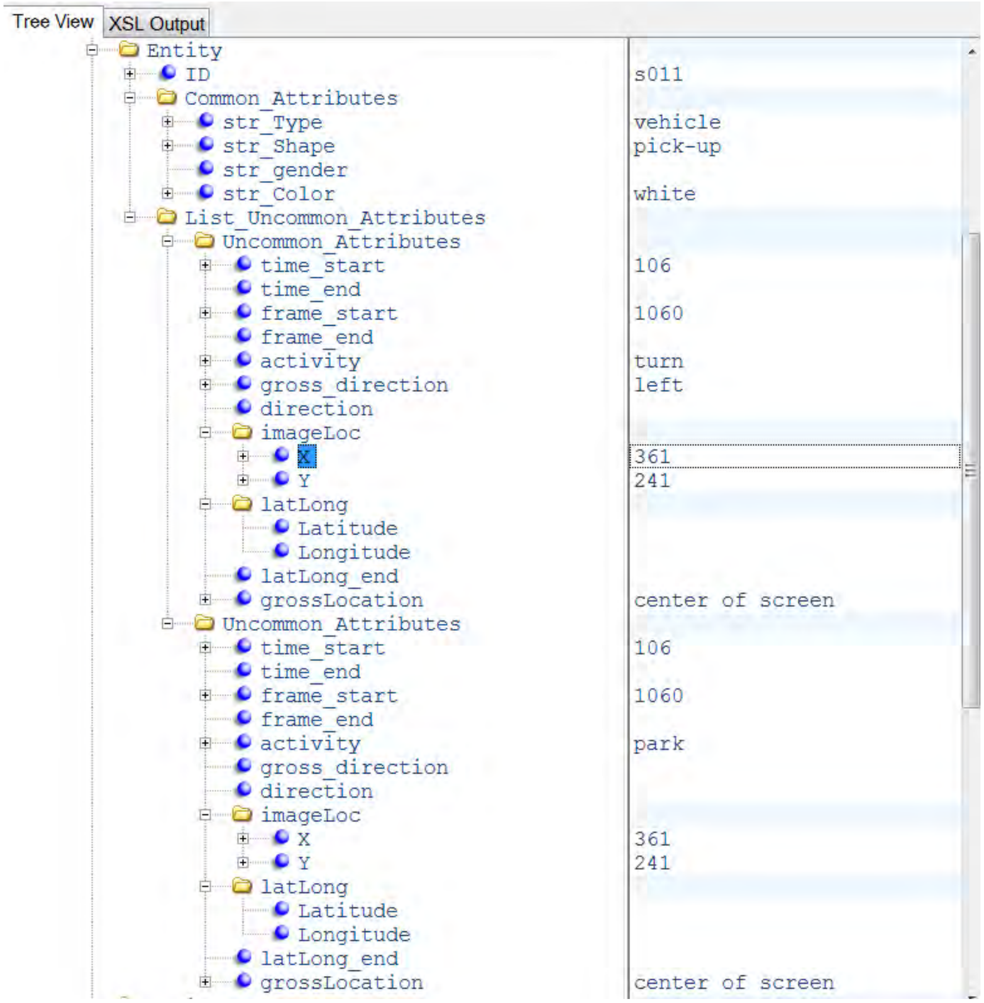
Constructing entities from soft data is more challenging as it involves reasoning over text (chat messages) to a certain extent. Similar to [109], we assume all chat messages follow a certain text structure/syntax and developed a structured text processing module to extract necessary information to construct the corresponding entities. In the future, this module will be replaced by a more general and powerful natural language processing (NLP) based module.

The heart of the structured text processing module includes the following seven key word banks:

TypeBank = {'individule','vehicle','car',…};

```
vShapeBank = {'suv','pick-up','truck',…};
pShapeBank = {'adult','kid',…};
GenderBank = {'male','female'};
ColorBank = {'black','white',…};
ActivityBank = {'travel','turn','back',…};
DirectionBank = {'north','west','south',…};
```

Figure 28 shows a sample entity constructed from chat message with ID = 11 from Creech data. Note that this time, the uncommon attribute imageLoc is derived from grossLocation by taking the center location of each value of the grossLocation attribute.



```
Entity
  ID                                s011
  Common_Attributes
    str_Type                        vehicle
    str_Shape                       pick-up
    str_gender
    str_Color                       white
  List_Uncommon_Attributes
    Uncommon_Attributes
      time_start                    106
      time_end
      frame_start                   1060
      frame_end
      activity                      turn
      gross_direction               left
      direction
      imageLoc
        X                           361
        Y                           241
      latLong
        Latitude
        Longitude
      latLong_end
      grossLocation                 center of screen
    Uncommon_Attributes
      time_start                    106
      time_end
      frame_start                   1060
      frame_end
      activity                      park
      gross_direction
      direction
      imageLoc
        X                           361
        Y                           241
      latLong
        Latitude
        Longitude
      latLong_end
      grossLocation                 center of screen
```

Figure 28: A sample entity constructed from the chat message with ID = 3 from Creech data set.

The message is “1 white pick-up truck turns left and parks under covered parking area north of runway center of screen”.

3.3.3.3 Soft-hard entity association

Soft-hard entity association is the first step toward soft-hard information fusion and hard-hard entity association is the first step toward linking tracklets of the same moving target. Both involve the calculation of the similarity between two given entities: soft-hard or hard-hard. Though it is not considered in this work, soft-soft entity association may bring some valuable information that can be exploited in one way or another. The approach of associating soft-soft entity and its benefit will be investigated in the future.

The purpose of soft-hard entity association is to identify, for each chat message, the corresponding track when the message is being uttered. Since the reviewer uttered the call-out message when he/she observed the moving target of interest, there must be a significant overlap in time/frames between the entities constructed from the chat message and its corresponding track. The first step of this module is to filter out those hard entities which do not have significant overlap with the soft entity being processed. If a hard entity is determined to have significant overlap with the current soft entity, -6 to + 4 seconds from the time the message is called out in this work, a similarity score can be computed based on the common attributes as well as uncommon attributes.

Similarity score for common attributes between a soft entity E_S and a hard entity E_H is given by

$$\begin{aligned} S_{common}(E_s, E_H) = \; & sign(E_S.type, E_H.type) \cdot W_t \\ & + sign(E_S.shape, E_H.shape) \cdot W_s \\ & + sign(E_S.gender, E_H.gender) \cdot W_g \\ & + likelihood(E_S.color, E_H.color) \cdot W_c \end{aligned} \tag{65}$$

Similarity score for uncommon attributes between two entities E_S and E_H is given by

$$\begin{aligned} S_{uncommon}(E_s, E_H) \; = \; & likelihood(E_S.activity, E_H.activity) \cdot W_a \\ & + likelihood(E_S.location, E_H.location) \cdot W_l \end{aligned} \tag{66}$$

where

$$sign(E_S.attri, E_H.attri) = \begin{cases} -1 \;\; if \; E_S.attri \; \neq \; E_H.attri \\ 1 \qquad otherwise \end{cases} \tag{67}$$

$likelihood(E_S.attri, E_H.attri)$ is the likelihood that $E_S.attri$ and $E_H.attri$ are the same, and W_t, W_s, W_g, W_c, W_a, W_l, are predetermined weights associated with each attribute such that $W_t + W_s + W_g + W_c = 1$ and $W_a + W_l = 1$.

Note that (66) penalizes only mismatched attributes. If $E_H.attri$ is not determined, e.g. before soft-hard entity fusion, which is to be explained later, no penalty is given.

The calculation of $likelihood(E_S.color, E_H.color)$, depending on the information available for $E_H.color$, consists of the following three cases:

a. $E_H.color = E_S.color$, ➔ $likelihood(E_S.color, E_H.color) = 1$.
b. $E_H.color \neq E_S.color$, ➔ $likelihood(E_S.color, E_H.color) = -1$.
c. $E_H.color$ is not determined, $likelihood(E_S.color, E_H.color)$ = the likelihood that the HUE histogram of E_H represents the color given by $E_S.color$.

The computation of $likelihood(E_S.activity, E_H.activity)$ is heuristically formulated depending on the value of $E_S.activity$. If $E_S.activity$ = 'travel', the velocity of E_H should maintain the same direction during the overlapped time period. In this case, we have

$$likelihood('travel', E_H.activity) = \frac{1}{2} \cdot (\cos d(\lceil 2 \cdot std(direction) \rceil)+1) \qquad (68)$$

Where *direction* is a vector of length equaling to the number of overlapped frames and each element is the direction of the moving target at that frame. The operator *std* stands for standard deviation and

$$\lceil x \rceil = \begin{cases} 180 \; if \; x > 180 \\ x \;\; otherwise \end{cases} \qquad (69)$$

If $E_S.activity$ = 'turn', the difference between the direction of E_H obtained from the earlier frames of the overlapped duration and that obtained from the later frames of the overlapped duration should be about 90 degree. Therefore we have

$$likelihood('turn\ left', E_H.activity) = \frac{1}{2} \cdot (\mathrm{cosd}(DIR_{begin} - DIR_{end} - 90) + 1) \qquad (70)$$

and

$$likelihood('turn\ right', E_H.activity) = \frac{1}{2} \cdot (\mathrm{cosd}(DIR_{end} - DIR_{begin} - 90) + 1) \qquad (71)$$

For $likelihood(E_S.location, E_H.location)$, we first convert the *grossLocation* attribute of E_S into *imageLoc* attribute by taking the indices of the center point of *grossLocation* attribute which is a description like 'center of screen', 'upper left of screen'. Then we formulate it as:

$$likelihood(E_S.location, E_H.location) = square(cos(norm(dist(E_S.location, E_H.location))))$$

where $dist(E_S.location, E_H.location)$ is the mean Manhattan distance of entities E_S and E_H calculated over the overlapped frames and *norm* is the normalization operator such that the resulting value is between 0 and $\pi/2$.

Finally, the similarity between E_S and E_H is given by

$$S(E_s, E_H) = S_{common}(E_s, E_H) \cdot W_{commom} + S_{uncommon}(E_s, E_H) \cdot W_{uncommom} \qquad (72)$$

with W_{commom} + $W_{uncommom}$ = 1.

3.3.3.4 Hard-hard entity association

The purpose of hard-hard entity association is to identify, for each hard entity, the closest later hard entity that is constructed from the track of the same moving target. This step is essential for linking tracklets from the same moving target but with different track IDs. Two hard entities can be associated if and only if the two entities do not have any overlapped frames. Thus, the first step of this module is to filter out those hard entities which either come before, or have overlap with, the hard entity being processed. Next, a similarity score can be computed based on the common attributes and uncommon attributes among those qualified candidates.

Similarity score for common attributes between two hard entities E_1 and E_2 is given by

$$\begin{aligned} S_{common}(E_1, E_2) = {} & sign(E_1.type, E_2.type) \cdot W_t \\ & + sign(E_1.shape, E_2.shape) \cdot W_s \\ & + sign(E_1.gender, E_2.gender) \cdot W_g \\ & + likelihood(E_1.color, E_2.color) \cdot W_c \end{aligned} \qquad (73)$$

which is very similar to eq.(65).

Similarity score for uncommon attributes between two entities E_1 and E_2 is given by

$$\begin{aligned} S_{uncommon}(E_s, E_H) = {} & likelihood(E_1.location, E_2.location) \cdot W_l \\ & + frame\_clossness(E_1.frame\_end, E_2.frame\_start) \cdot W_f \end{aligned} \qquad (74)$$

where $sign(E_1.attri, E_2.attri)$ is given by eq. (67) and the calculation of $likelihood(E_1.color, E_2.color)$, depending on the available information for $E_1.color$ and $E_2.color$, consists of the following four cases:

a. $E_1.color, = E_2.color$, ➔ $likelihood(E_1.color, E_2.color) = 1$.
b. $E_1.color, \neq E_2.color$, ➔ $likelihood(E_1.color, E_2.color) = -1$.
c. Either $E_1.color$ or $E_2.color$ is not determined. In this case, $likelihood(E_1.color, E_2.color)$ = the likelihood that the HUE histogram of E_2 represents the color given by $E_1.color$. Here we assume that the attribute $E_1.color$ is available and $E_2.color$ is not available.
d. Both $E_1.color$ and $E_2.color$ are not determined. In this case, $likelihood(E_1.color, E_2.color) = \overline{V_{E_1.HUE}} \cdot \overline{V_{E_2.HUE}}$, where $\overline{V_{E_1.HUE}}$ and $\overline{V_{E_2.HUE}}$ are normalized HUE histogram of E_1 and E_2.

The formula for $likelihood(E_1.location, E_2.location)$ is very similar to the one given by eq. (71) except that now both $E_1.location$ and $E_2.location$ are given as *imageLoc* attributes.

The term $frame\_clossness(E_1.frame\_end, E_2.frame\_start)$ is devised to estimate how likely the two hard entities can be sequentially linked in time. We simply take

$$frame\_clossness(E_1.frame\_end, E_2.frame\_start) =$$

$$½ \cdot (\cos(\mathrm{norm}(E_2.frame_{start} - E_1.frame\_end))+1) \qquad (75)$$

where norm is a normalization operator such that the resulting value is between 0 and π.

Finally, the similarity between E_1 and E_2 is given by eq.(72).

3.3.3.5 Hard-soft entity fusion

Soft-hard entity fusion is realized by combining common attributes from both entities. Uncommon attributes are not combined in the present work. One changeling in combining uncommon attributes from soft and hard entities is that one need to be able to segment a track based on its activity at each frame. Though this information is accommodated by KW18 format, the field is always empty in the current Cheech data set.

We want to note that soft-hard entity fusion is performed immediately after a soft entity is associated to a hard entity rather than being performed after the completion of soft-hard entity association as the fused hard entities will facilitate the remaining soft-hard entity association process.

3.3.3.6 Hard-soft entity linkage

Once all entities constructed from soft and hard data are associated and fused, we are ready to link the hard entities sequentially to generate a long-duration track for each moving target, which is the goal of this work. Figure 5 shows the proposed hard-hard entity linkage approach. In Figure 29, E_{end_tmp} is the hard entity satisfying the following three conditions:

1. It is associated with E_{end}.

2. It can be concatenated to E_i. That is, E_{end_tmp} appears after E_i and there is no overlapped frames between E_{end_tmp} and E_i.
3. It is the closest one to E_i if there are more than one hard entity satisfying 1 and 2.

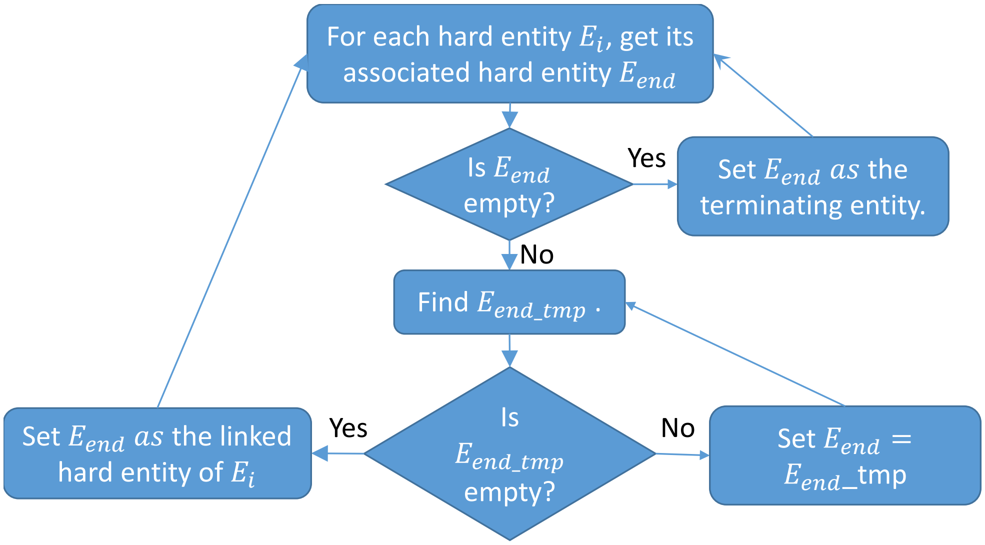

For each hard entity E_i, get its associated hard entity E_{end}
Is E_{end} empty?
Yes
Set E_{end} as the terminating entity.
No
Find E_{end_tmp} .
Set E_{end} as the linked hard entity of E_i
Yes
Is E_{end_tmp} empty?
No
Set E_{end} = E_{end}_tmp

Figure 29: The proposed hard-hard entity linkage workflow.

The approach described in this section has been prototyped as IFT's V2T fusion software and graphical user interface (GUI) to be presented in Section 4.

3.4 Event Detection

3.4.1 Introduction

Human activity recognition is an important area of computer vision research. It has the history for several decades. The many applications in this area include surveillance systems, patient monitoring systems, and a variety of systems. Along with the development of personal mobile devices, nowadays these applications also involve interactions between persons and electronic devices such as human-computer interfaces. Most of these applications require an automated recognition of high-level activities, composed of multiple simple (or atomic) actions of persons.

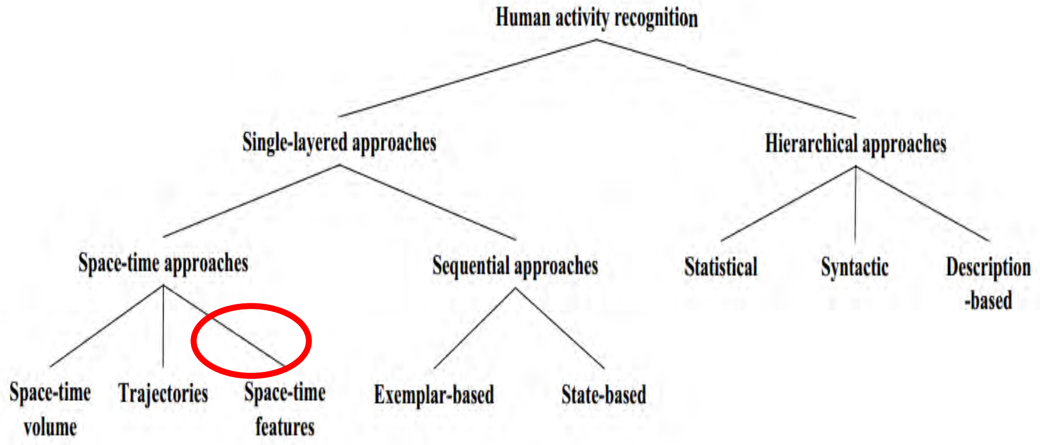


Figure 30: The hierarchical approach-based

Figure 30 provides a main idea of the overview of the tree-structured taxonomy that contains most popular approaches in human activity analysis. The space-time features under space-time approaches are the approach we use during solving event detection.

The event detection is aiming at two main aspects. First, a single layered approach on space-time features judges the motion property of multiple identical targets. After the event detection, the status whether a target is running or not is output. The second aspect is human-object interaction and group activities. All targets may interact during the process; the proposed framework should be able to detect any possible interaction. To be specific, the output of the event detection should be able to label whether a human target is getting into another target, a car or a building.

3.4.2 Action Recognition based on spatial-temporal features

We are using the space-time local features to build up our approach. The approach takes advantages of local features extracted from 3-dimensional (3-D) space-time volumes to represent and recognize activities.

Figure 31 shows an example from the annotated data of clip 01. This figure gives a main idea of what is a 3-D space-time (3D-ST) space. The 6 images given in the time axis are following the time sequence. They are from data clip 01 frame 231 to frame 281. The targets in these frames are marked in blue and red rectangular, which can be found on the top right corner of each frame. The features we are using are the location of all these targets in this 3-D space. Their motion trend is clearly indicated by a curve in the space.

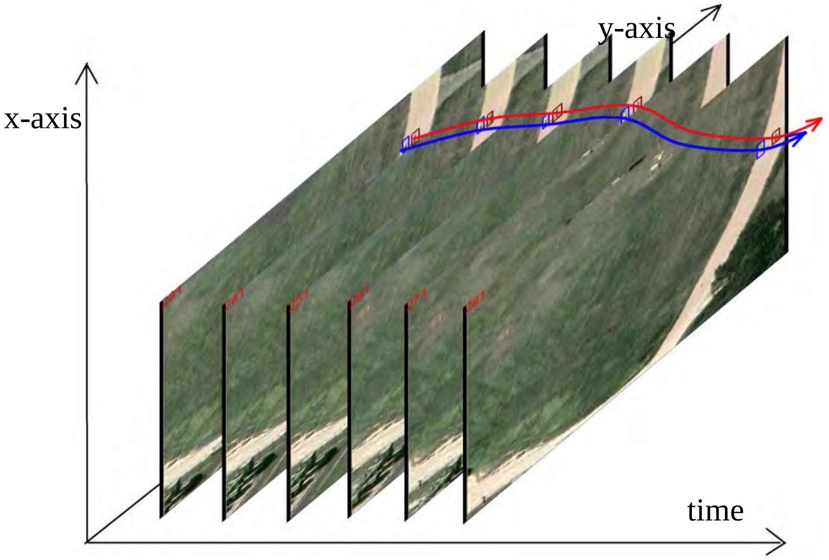


Figure 31: 3-D space-time annotation of multi targets

The motion trend of target in red rectangular is indicated by red curve. So do the other target. By analyzing these curves, we can get the motion property of a single target. In our implement, we have set a speed control threshold to judge whether a target is running or not. We manually choose this threshold according to the property of data clips, and this threshold can also be trained by machine learning strategy instead of by hand. If the moving distance of a target is larger than the preset threshold, then we mark this target the property of running as its event.

3.4.3 Human-Object Interactions and Group Activities

In order to recognize interactions between humans and objects, an integration of multiple components is required. The locations of both targets are required into one single interaction. Besides, the motion trend curves in 3D-ST space also indicate interactions in the real world. The identification of objects and motion involved in an activity as well as analysis of their interplay is essential for the reliable recognition of human activities involving humans and objects.

Specifically, in our study we aim at the recognition of a human getting into a car. As shown in Figure 32. The trend of the motion from a human target in 3D-ST space is approaching an annotated car, which we call this a human-car-approaching request. The motion trend line in 3D-ST space is indicated on the right image clip using a curved arrow. In our implementation, we have set a distance control threshold. This manually set threshold gives the criteria of judging the event of a human getting into a car. Under the circumstances that the motion trend meets the approaching request, the framework will mark a human is getting into a car when the distance between them is less than the threshold.

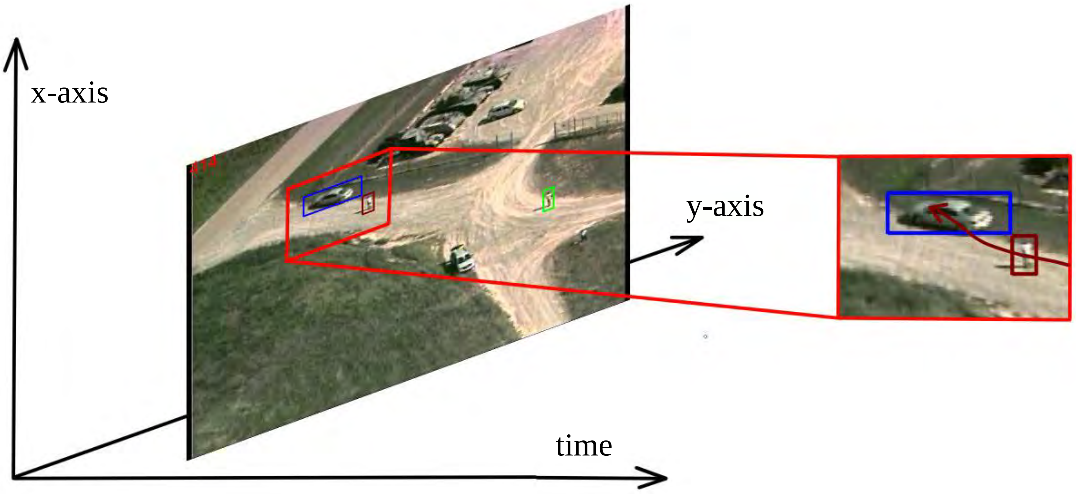


Figure 32: Interaction of a human and a car in a space-time 3D dimension

3.4.4 An event detection framework

Figure 33 presents the main idea of the implemented framework. The whole system is built on an online strategy. In each step, the judgment that the system made only depends on the evidence given by the target tracking results (target locations) before the time point. The event detection is isolated from the main system framework. This makes it easy to modify these two function modules to adapt for alternative usage.

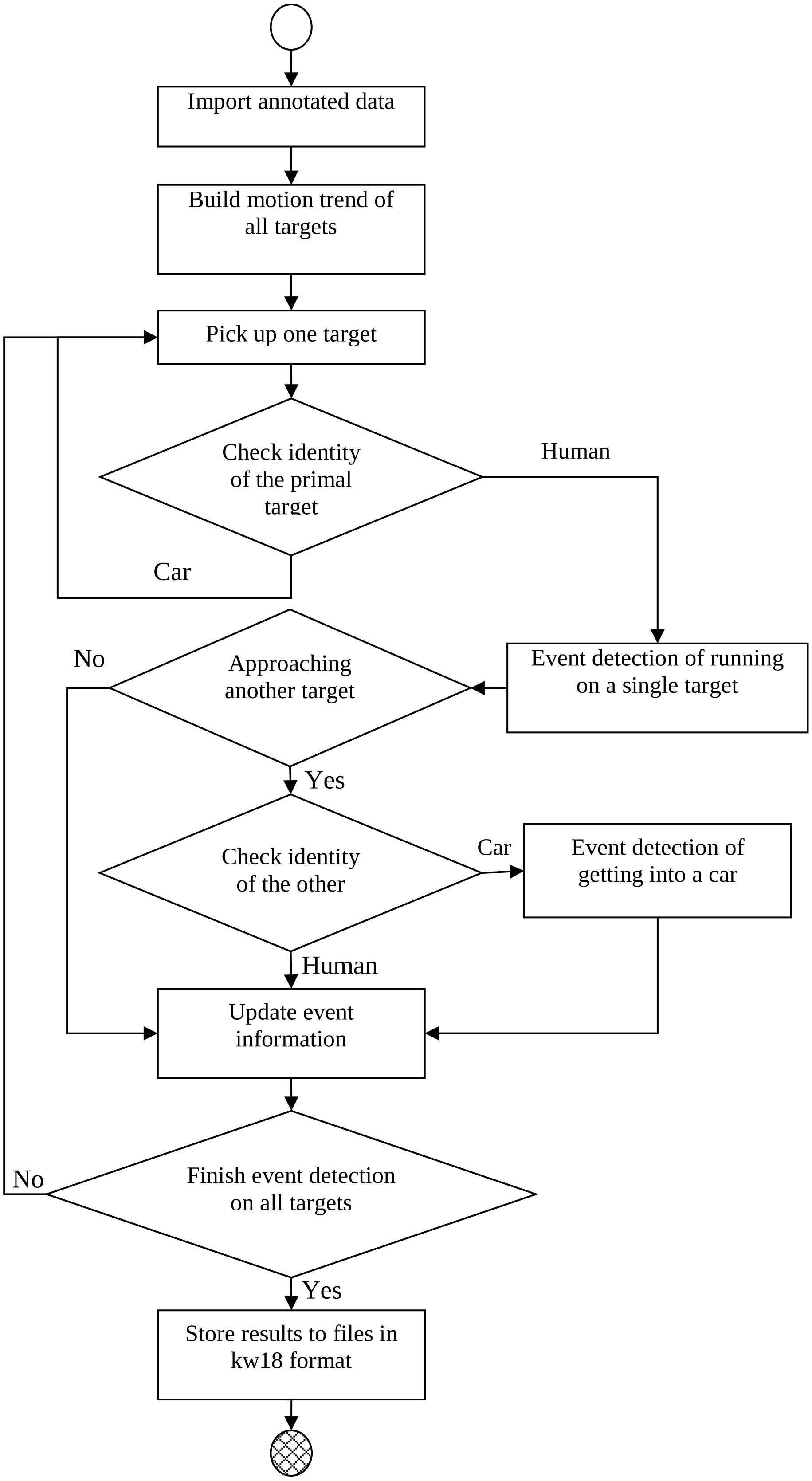

Import annotated data
Build motion trend of all targets
Pick up one target
Check identity of the primal target
Human
Car
Approaching another target
No
Event detection of running on a single target
Yes
Check identity of the other
Car
Event detection of getting into a car
Human
Update event information
Finish event detection on all targets
No
Yes
Store results to files in kw18 format

Figure 33: Algorithm Framework

3.4.5 Event Types

In Virat data Release 2.0, 12 different types of events have been defined including:

(1) Person loading an Object to a Vehicle

Description: An object moving from a person to a vehicle. The act of 'carrying' should not be included in this event.

Annotation: 'Person', 'Object' (optional), and 'Vehicle' should be annotated.

Start: The event begins immediately when the cargo to be loaded is “extended” toward the vehicle (i.e., before one's posture changes from one of 'carrying', to one of 'loading.').

End: The event ends after the cargo is placed in the vehicle and person-cargo contact is lost. In the event of an occlusion, it ends when the loss of contact is visible.

(2) Person Unloading an Object from a Vehicle

Description: An object moving from a vehicle to a person.

Annotation: 'Person', 'Object' (optional), and 'Vehicle' should be annotated.

Start: The event begins immediately when the cargo begins to move. If the start of the event is occluded, it begins when the cargo movement is first visible.

End: The event ends after the cargo is released. If a person, while holding the cargo, begins to walk away from the vehicle, the event ends (at which time the person is 'carrying'). The event also ends if the vehicle drives away while the person is still in contact with the cargo; after the vehicle has been in motion for more than 2 seconds, the person is 'carrying'.

(3) Person Opening a Vehicle Trunk

Description: A person opening a trunk. A trunk is defined as a container specifically designed to store nonhuman cargo on a vehicle. A trunk need not have a lid (i.e., the back of a pickup truck is a trunk), and it need not open from above (i.e., the back of a van, which opens via double doors, is also a trunk).

Annotation: 'Person', and 'Vehicle' should be annotated with bounding boxes for as many frames as possible during the event duration. The box annotation of 'Trunk' is optional.

Start: The event begins when the trunk starts to move.

End: The event ends after the trunk has stopped moving.

(4) Person Closing a Vehicle Trunk

Description: A person closing a trunk.

Annotation: 'Person', and 'Vehicle' should be annotated with bounding boxes for as many frames as possible during the event duration. The box annotation of 'Trunk' is optional.

Start: The event begins when the trunk starts to move.

End: The event ends after the trunk has stopped moving.

(5) Person getting into a Vehicle

Description: A person getting into, or mounting (e.g., a motorcycle), a vehicle.

Annotation: 'Person', and 'Vehicle' should be annotated.

Start: The event begins when the vehicle's door moves, or, if there is no door, 2 s
before ½ of the person's body is inside the vehicle.

End: The event ends when the person is in the vehicle. If the vehicle has a door, the event ends after the door is shut. If not, it ends when the person is in the seated position, or has been inside the vehicle for 2 seconds (whichever comes first).

(6) Person getting out of a Vehicle

Description: A person getting out of, or dismounting, a vehicle.

Annotation: 'Person', and 'Vehicle' should be annotated.

Start: The event begins when the vehicle's door moves. If the vehicle does not have a door, it begins 2 s before ½ of the person's body is outside the vehicle.

End: The event ends when standing, walking, or running begins.

(7) Person gesturing

Description: A person gesturing. Gesturing is defined as a movement, usually of the body or limbs, which expresses or emphasizes an idea, sentiment, or attitude. Examples of gesturing include pointing, waving, and sign language.

Annotation: 'Person' should be annotated.

Start: The event begins when the gesture is evident. For example, when waving, the gesture when the waver begins to raise their arm into the "waving position."

End: The event ends when the motion ends

(8) Person digging (Note: not existing in Release 2.0)

Description: A person digging. Digging may or may not involve the use of a tool (i.e., digging with one's hands is still considered 'digging'; hands are the tool).

Annotation: 'Person' should be annotated.

Start: The event begins when the tool makes contact with the ground.

End: The event ends 5 s after the tool has been removed from the ground, or immediately if the digging tool is dropped.

(9) Person Carrying an Object

Description: A person carrying an object. The object may be carried in either hand, with both hands, or on one's back. Objects annotated by boxes are optional and subject to the difficulty.

Annotation: 'Person', and 'Object' (optional) are annotated.

Start: The event begins when the person who will carry the object, makes contact with the object. If someone is carrying an object that is initially occluded, the event begins when the object is visible.

End: The event ends when the person is no longer supporting the object against gravity, and contact with the object is broken. In the event of an occlusion, it ends when the loss of contact is visible.

(10) Person running

Description: A person running for more than 2s.

Annotation: 'Person' should be annotated.

Start: When a person is visibly running.

End: The event will end 2 s after the person is no longer running. If transitioning to Standing, Walking or Sitting the event will end after Standing, Walking or Sitting.

(11) Person entering a facility

Description: A person entering a facility

Annotation: 'Person' should be annotated.

Start: The event begins 2 s before that person crosses the facility‘s threshold.

End: The event ends after the person has completely disappeared from view.

(12) Person exiting a facility

Description: A person exiting a facility
Annotation: 'Person' should be annotated.
Start: The event begins as soon as the person is visible.
End: The event ends 2 seconds after the person is completely out of the facility.

3.4.6 Event Detection based on L1 Tracking Results

In this project, we use the target tracking results provided by the L1 tracker to extract target activity and event. We obtain the tracked targets' position and speed and then process the obtained targets position and speed to extract their activities and the corresponding events. In this VIRAT dataset, we can extract following events:

(1) Person getting into a Vehicle;
(2) Person running; and
(3) Person entering a facility.

Except the above event, we can also obtain

(4) Person walking;
(5) Person stop; and,
(6) Car stop.

In the selected video sequence, there is clearly camera movement and this movement leads to target position shift in consecutive video frames. Thus we add a static target in tracking to calibrate the camera movement when there is no desired static target (e.g., in Figure 34). In this compensation process, we assume camera movement is limited to linear in plane translation, without zoom in/out or rotation.

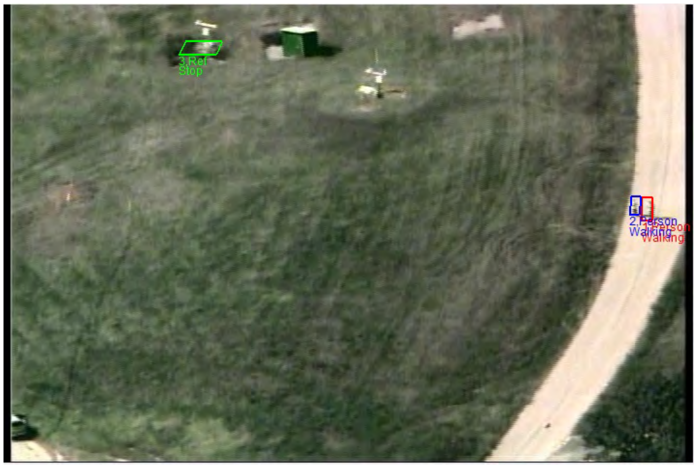


Figure 34: Two person walking.

Assuming the static target is at $(x_{s,n}, y_{s,n})$ in frame n, and due to the camera movement the static target is at $(x_{s,n+1}, y_{s,n+1})$ in frame $n+1$, then the camera movement compensation parameter is

$$x_{c,n+1} = x_{s,n+1} - x_{s,n},\, y_{c,n+1} = y_{s,n+1} - y_{s,n}. \tag{76}$$

Assume the desired mobile target 1 location is at $(x_{t1,n+1}, y_{t1,n+1})$ in frame $n + 1$, then the calibrated target 1 location in frame $n + 1$ is

$$x'_{t1,n+1} = x_{t1,n+1} - x_{c,n+1},\;\; y'_{t1,n+1} = y_{t1,n+1} - y_{c,n+1}. \tag{77}$$

The location obtained after compensation is the target 1's position in the scene of the first frame. Thus we can calculate the target's speed. Please note, with two reference object, we can calibrate both the camera linear movement and rotation.

With the compensated target position in each frame, we calculate the target speed in x and y direction with

$$v_{x,t1,n+1} = N\left(x'_{t1,n+1} - x'_{t1,n}\right),\, v_{y,t1,n+1} = N\left(y'_{t1,n+1} - y'_{t1,n}\right), \tag{78}$$

where *N* is the number of frames per second of the video. Please note that the speed is in pixels per second.

There is no camera parameters provided with the video, so we use an object as reference in the video to obtain the approximate speed of the interested targets. We used person's height in pixels as a reference, and calculate the target speed with respect to the reference and set a threshold to separate running and walking.

The event, for example Person getting into a Vehicle or Person entering a facility is detected using the distance between the person and the vehicle (facility). The distance is calculated using the compensated target location. We also use the target histogram comparison to detect if the target is still in the scene to help the event detection.

4 RESULTS AND DISCUSSION

4.1 Results of tracking and multi-target association

We tested our algorithm on 3 different sequences obtained from Virat data named “09152008flight2tape1_5.mpg”, the first sequence with two people interacting on the road contains 600 frames, we are able to track it around 500 frames until a large movement from the camera causes a large blur. We obtained good results as shown in the following figures.

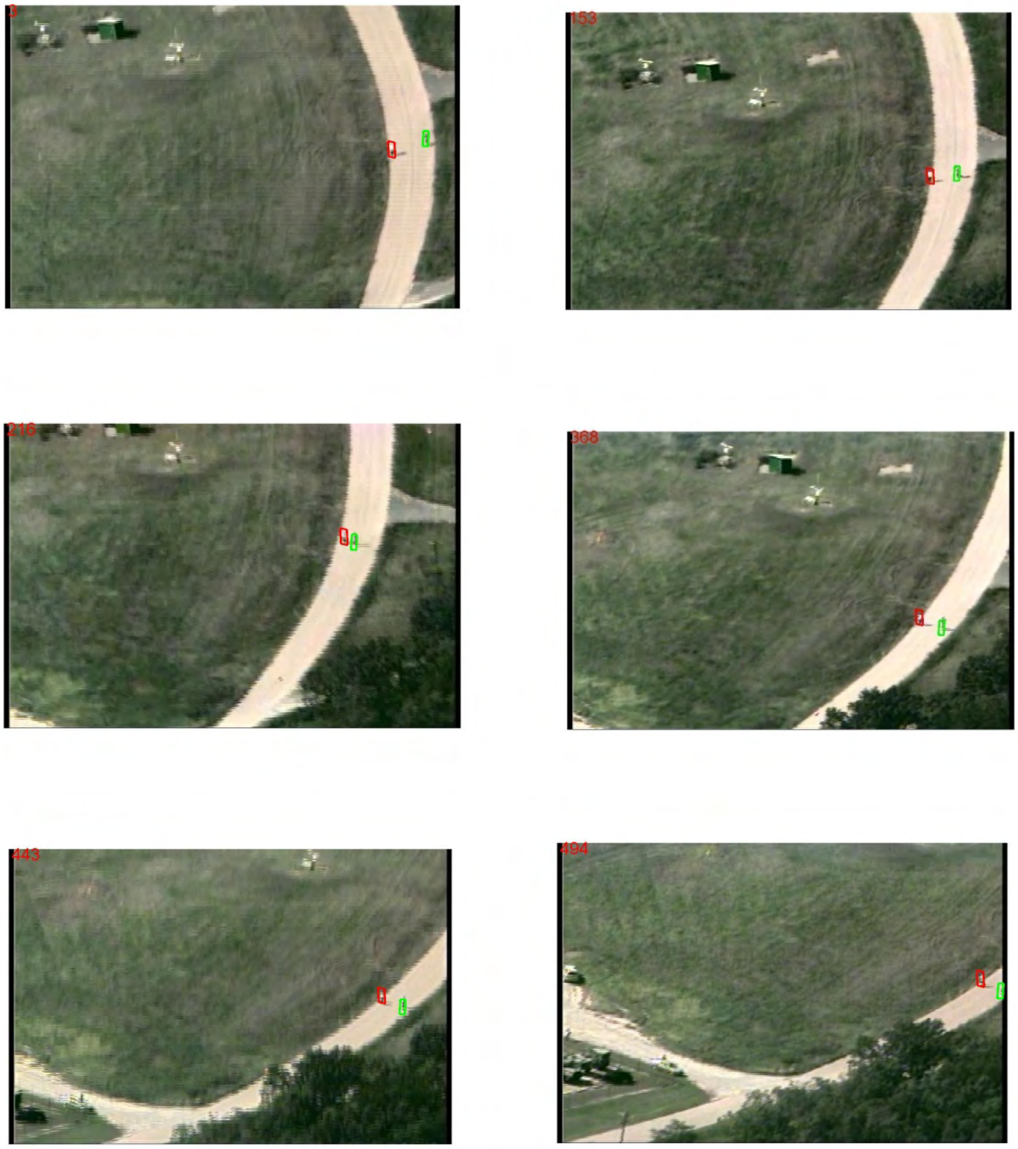


Figure 35: Sequence 1 results.

Sequence 2 contains a moving vehicle, since it has neither big blur nor appearance change, we can track it perfectly.

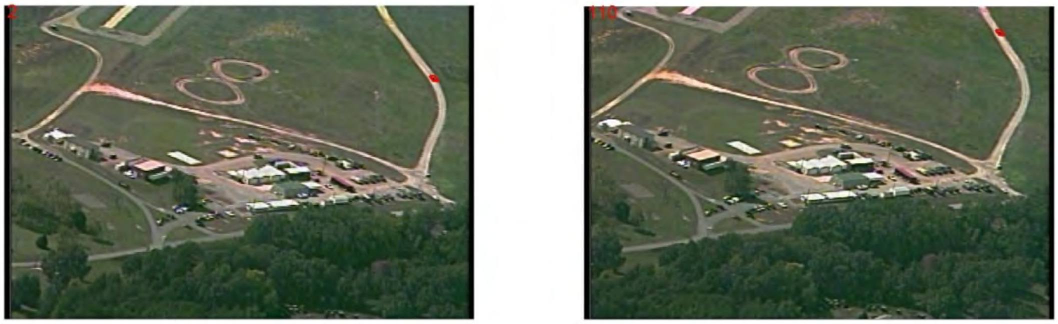


Figure 36: Sequence 2 results.

However, the sequence 3 with several people interacting with cars and each other remains a challenge. The main reason is the big blur and the initial low image quality of the target. At the first trial, the low quality of the initialization yields an early failure. So we pick up a frame where the targets are stable and run the trial 2. It can track over 300 frames until again the large blur occurs.

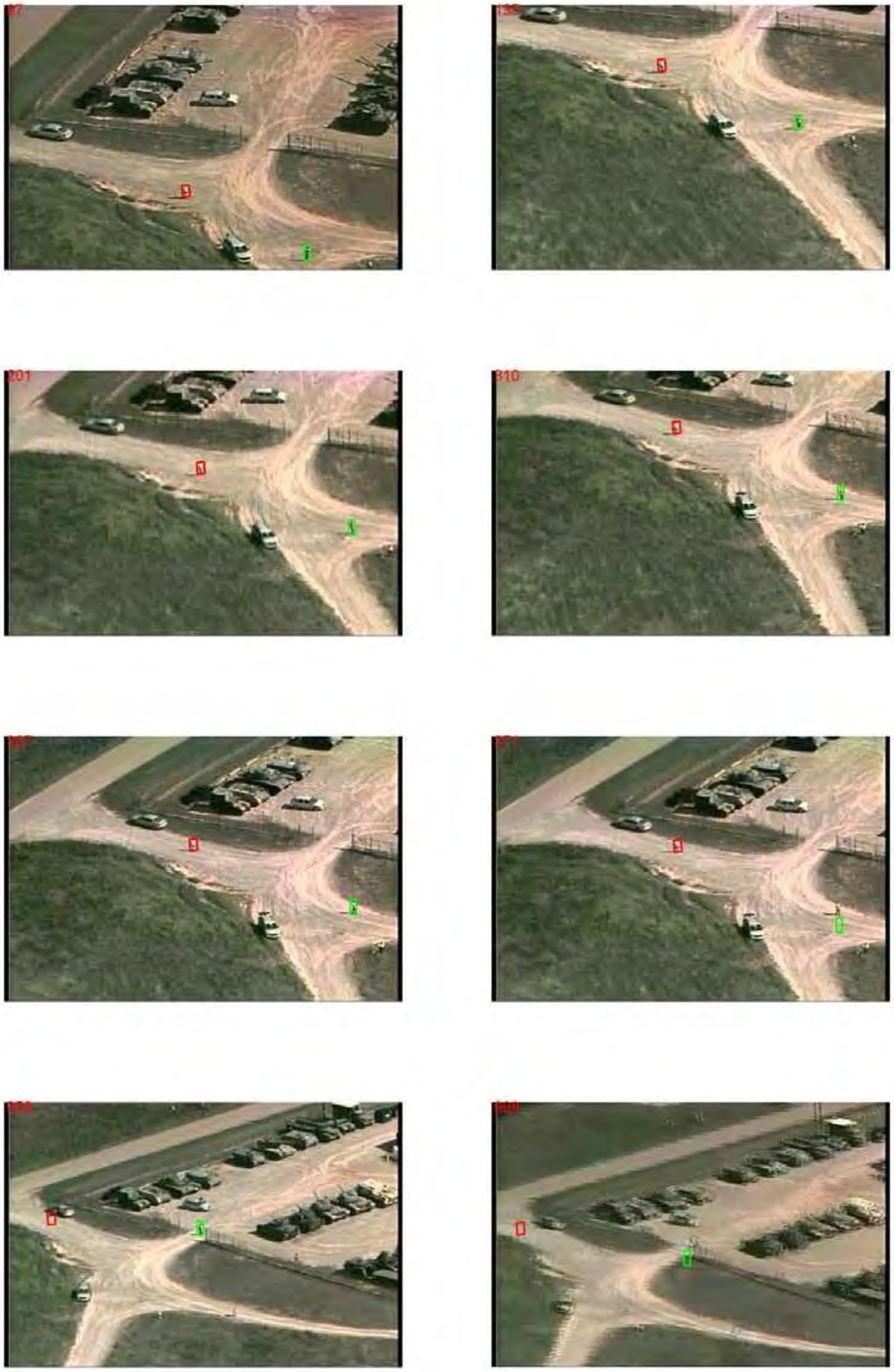


Figure 37: Sequence 3 results trial 2, starting from frame 87.

From all these experiments, it shows that our algorithm could track the targets in most of the cases except when there is large motion blur. We will make further analysis to find the problem lies behind it and make updates accordingly.

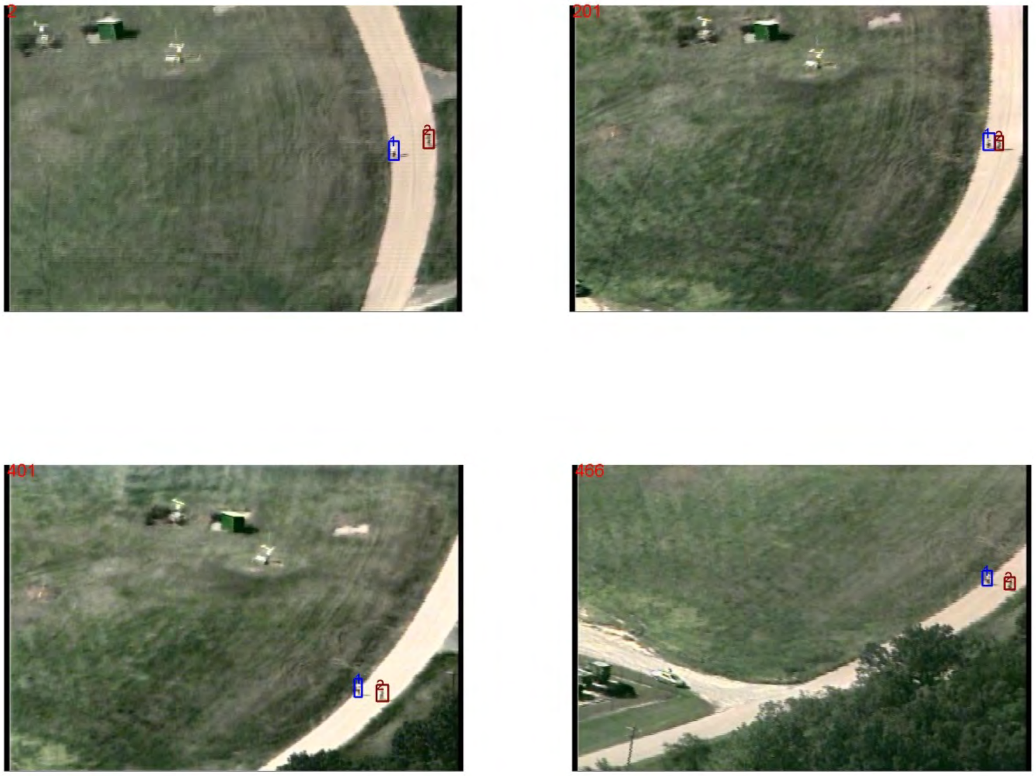


Figure 38: Association result on dataset 1

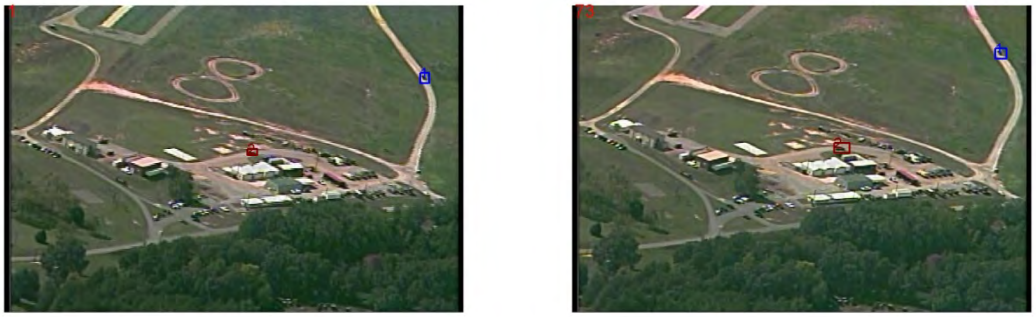


Figure 39: Association result on dataset 3

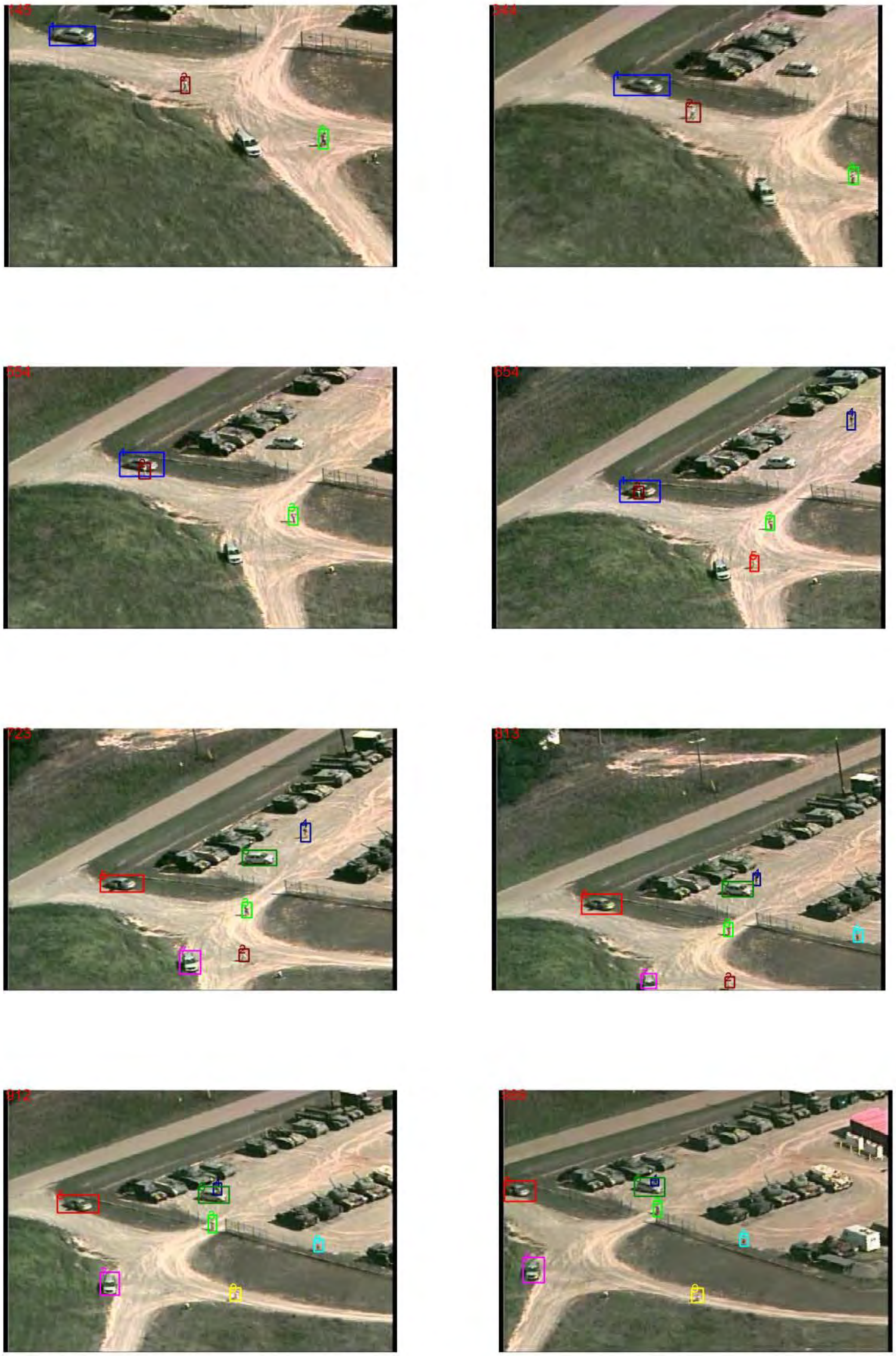
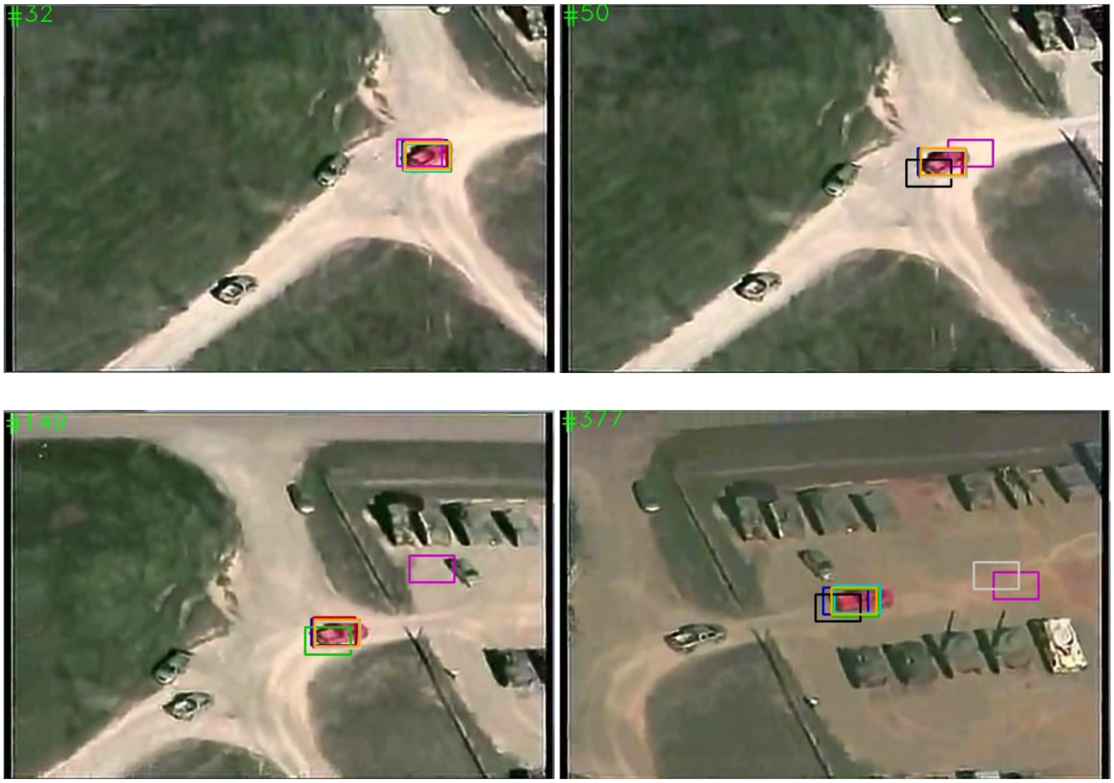


Figure 40: Association result on dataset 2

4.2 Tracker Comparison

4.2.1 Virat Dataset

In the experiments, the platform of running all the trackers is Intel core i7-4500U 2.4GHz and 8 GB memory. To quantitatively compare robustness under challenging conditions, we manually annotated the target's bounding box in each frame for all the test sequences. The test sequences we selected are the classical sequences for video tracking "jogging" and "pole". Video "car" is a very challenging airborne video sequence in VIRAT Video Dataset. As can be seen in Figure 41, Tracker PF(pink) and S-BOOST(white) do not perform robustly under low-resolution and realistic conditions. As shown in Table 1 and Table 2, TLD and Compressive Trackers show their robustness in average tracking errors and tracking quality comparison respectively.



Legend: PF(pink) FRAG(green) STRUCK(cyan) BOOST(black) S-BOOST(white) MIL(orange) TLD(red) CT(blue)

Figure 41: Tracking results of different algorithms in video "car"

Table 1: The average tracking errors on Virat dataset

	PF	FRAG	STRUCK	BOOST	S-BOOST	MIL	TLD	CT
jogging	0.1885	0.6383	0.8526	0.0570	0.8916	0.8211	**0.0056**	0.0085
pole	0.7520	0.0409	0.5728	0.0109	0.8591	0.0072	**0.0068**	0.0093
car	6.9842	0.3216	0.4685	0.4169	4.2561	0.5714	0.2034	**0.1026**
Average	2.6416	0.3336	0.6313	0.1616	2.0022	0.4665	**0.0719**	0.1204

The error is measured using the Euclidean distance of two center points, which has been normalized by the size of the target from the ground truth. The last row is the average error for each tracker over all the test sequences.

Table 2: The average tracking quality on Virat dataset

	PF	FRAG	STRUCK	BOOST	S-BOOST	MIL	TLD	CT
jogging	0.4141	0.1643	0.1339	0.1761	0.1294	0.5333	0.5369	**0.6836**
pole	0.3063	0.2791	0.3524	0.3939	0.0176	0.3422	**0.5449**	0.5263
car	0.0120	0.2941	0.2516	0.4224	0.0202	0.3618	0.4865	**0.6572**
Average	0.2441	0.2458	0.2459	0.3308	0.0557	0.4124	0.5227	**0.6223**

The quality is measured using the area coverage between the tracking box and the annotation.

4.2.2 Skybox Dataset

The video we used to evaluate the trackers is the “Skybox Imaging HD Video of Mining Activity in Uşak, Western Turkey”: http://player.vimeo.com/video/95913805

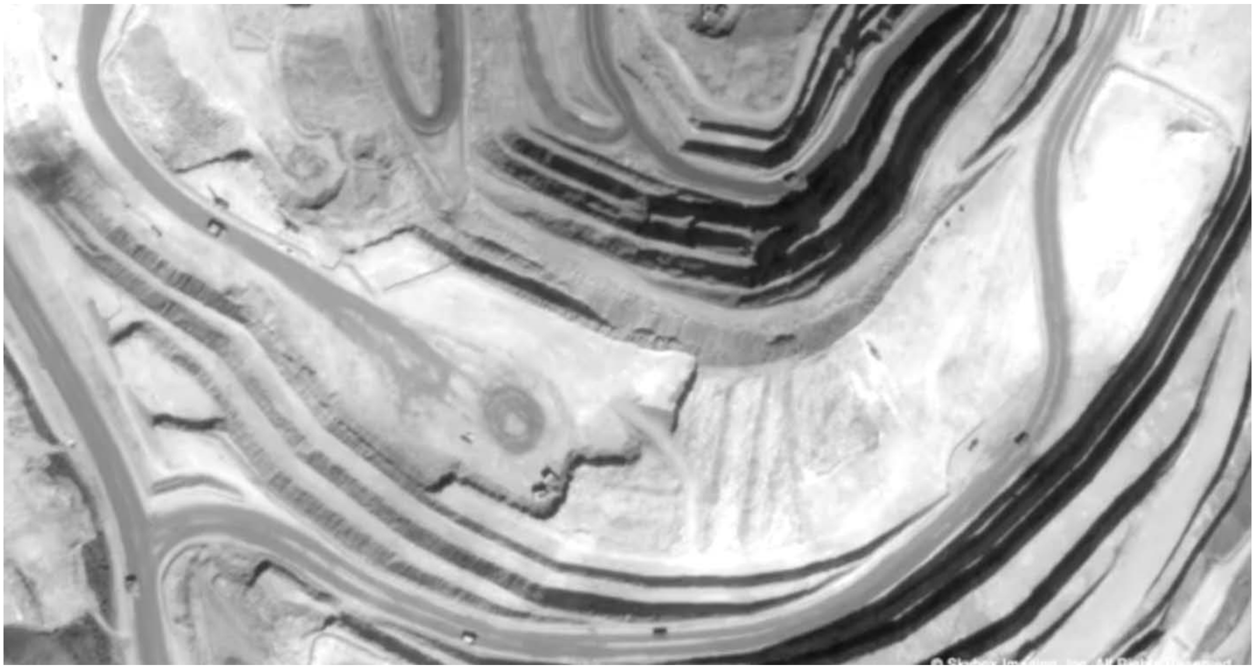


Figure 42: The screen shot of the Skybox Imaging HD Video

The video is a full motion HD video from space of a gold mine in Uşak, Western Turkey. It was collected by SkySat-1 on March 23, 2014.

In the experiments, the platform of running all the trackers is Intel core i7-4500U 2.4GHz and 8 G memory. To quantitatively compare robustness under challenging conditions, we manually annotated the target's bounding box in each frame for all the test sequences. The test sequences we selected are a skybox image sequence filmed in Turkey. Vehicle 1 and 2 are selected as the objects of interests.

As can be seen in Table 3 and 4, VTD and TLD(red) do not perform robustly under low-resolution and realistic conditions. Also as shown in Table 1 and Table 2, OAB, L1 and CT tracker show their robustness in average tracking errors and tracking quality comparison respectively.

Table 3: The average tracking errors (%) on a Skybox video

	CT	TLD	MIL	OAB	VTD	L1	LOFT	CSURF
Vehicle1	0.0000	0.8703	0.0000	0.0000	0.9499	0.0000	0.3190	0.3580
Vehicle2	0.5410	0.9165	0.4046	0.0000	0.9722	0.6510	0.2327	0.2977
Average	0.2707	0.8934	0.2023	0.0000	0.9611	0.3255	0.2728	0.3279

The error is measured using the Euclidean distance of two center points, which has been normalized by the size of the target from the ground truth. The last row is the average error for each tracker over all the test sequences.

Table 4: The average tracking quality on a Skybox video

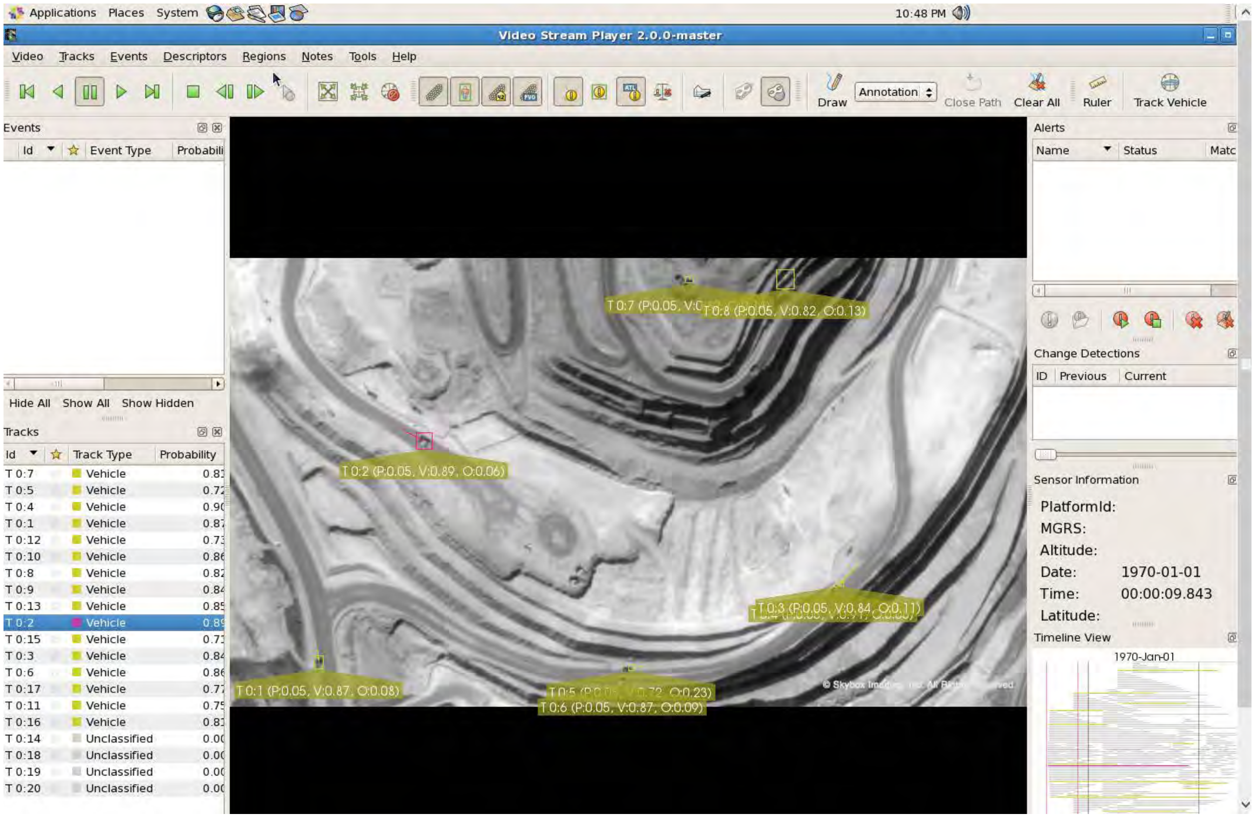
	CT	TLD	MIL	OAB	VTD	L1	LOFT	CSURF
Vehicle1	0.8253	0.4143	0.3113	0.9117	0.2012	0.8953	0.62	0.6813
Vehicle2	0.8442	0.2191	0.2535	0.8739	0.1501	0.9034	0.55	0.5673
Average	0.8348	0.3166	0.2824	0.8928	0.1757	0.8994	0.585	0.6243
Frame rate (frames/sec)	15.96	4.629	0.269	15.6	16.06	1.743	2.674	2.563

The quality is measured using the area coverage between the tracking box and the annotation.

Figure 43 shows tracking results of LOFT tracker on a Skybox video. The screen layout for vsPlay includes a row of tabs, which contain drop down menu functions at the top of the screen. These drop down menus include a number of icon functions and provide additional capabilities for the user to interact with the screen display layout and video based upon tab selection. To provide maximum situational awareness of activity occurring within the FMV feed an analyst should set their screen layout to display the tracks pane (by selecting the show track list from the drop down menu), events pane (by selecting the show events list from the drop down menu), and the change detections pane (by selecting the Descriptors tab and ensuring the show alert list is activated). Each screen layout tab is described below:

- Video Tab - Controls the FMV feed such as start, stop, decrease speed;

- Tracks Tab – Provides on screen displays for the MTT such as track ID’s, entity bounding boxes, PVO scores;
- Events Tab – Displays events that correlate with the FMV feed such as show all person/vehicle events;
- Descriptors Tab – Works with alerts to activate/deactivate or shows alerts;
- Regions Tab – Supports analyst ability to create/select/de-select or display regions of interest for activity or non-activity within the FMV feed;
- Tools Tab – Provides report generation, measuring/ruler, display change detection list functions.


Applications Places System
10:48 PM
Video Stream Player 2.0.0-master
Video Tracks Events Descriptors Regions Notes Tools Help
Draw
Annotation
Close Path
Clear All
Ruler
Track Vehicle
Events
Id Event Type Probabili
Hide All Show All Show Hidden
Alerts
Name Status Matc
Change Detections
ID Previous Current
Sensor Information
Timeline View
1970-Jan-01
T 0:7 (P:0.05, V:0
T 0:8 (P:0.05, V:0.82, O:0.13)
T 0:2 (P:0.05, V:0.89, O:0.06)
T 0:3 (P:0.05, V:0.84, O:0.11)
T 0:1 (P:0.05, V:0.87, O:0.08)
T 0:5 (P:0.0 V:0.72, O:0.23)
T 0:6 (P:0.05, V:0.87, O:0.09)
© Skybox Imaging

Tracks

Id	☆	Track Type	Probability
T 0:7		🟨 Vehicle	0.81
T 0:5		🟨 Vehicle	0.72
T 0:4		🟨 Vehicle	0.90
T 0:1		🟨 Vehicle	0.87
T 0:12		🟨 Vehicle	0.73
T 0:10		🟨 Vehicle	0.86
T 0:8		🟨 Vehicle	0.82
T 0:9		🟨 Vehicle	0.84
T 0:13		🟨 Vehicle	0.85
T 0:2		🟪 Vehicle	0.89
T 0:15		🟨 Vehicle	0.71
T 0:3		🟨 Vehicle	0.84
T 0:6		🟨 Vehicle	0.86
T 0:17		🟨 Vehicle	0.77
T 0:11		🟨 Vehicle	0.75
T 0:16		🟨 Vehicle	0.81
T 0:14		⬜ Unclassified	0.00
T 0:18		⬜ Unclassified	0.00
T 0:19		⬜ Unclassified	0.00
T 0:20		⬜ Unclassified	0.00

PlatformId:
MGRS:
Altitude:
Date: 1970-01-01
Time: 00:00:09.843
Latitude:

Figure 43: The tracking results of LOFT tracker displayed in Kitware vsPlay

4.3 Hard-Soft and Hard-Hard Fusion

4.3.1 Software and GUI for Fusion

Our software tool for hard-soft fusion can:

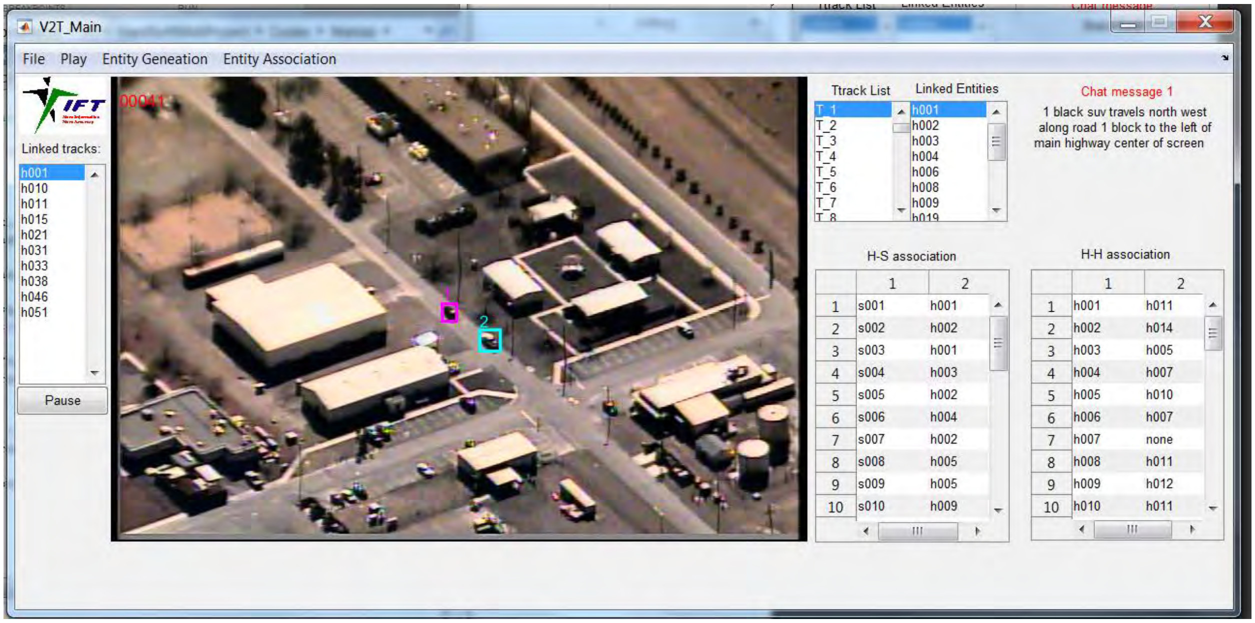
1) Display tracked moving targets represented in KW18 format individually or collectively on the screen.

2) Display soft messages synchronized with hard (video) data.
3) Associate soft messages with their corresponding tracks.
4) Link tracklets of the same moving target with *different* labels into a complete track with a unique label.

Figure 44 shows the layout of the GUI. An overview of the GUI follows:

At the center of the screen is the main display of the video data with tracked targets.

1. Once the tracks are loaded, they are displayed in *Track List.*
2. Current chat message is displayed in *Chat message* while the video is being played.
3. To perform soft-hard (chat messages and video tracks) and hard-hard (tracks and tracks) data association, entities need to be extracted first. The results of these associations are displayed in *H-S association* table and *H-H association* table.
4. After hard-hard data association, entities (each entity corresponds to a tracklet) can be linked to produce a complete track of the same moving target. The linked entities are listed in *Linked Entities*.
5. User is able to display each linked entity individually. When a linked entity is selected, the concatenated tracklets (each corresponds to a hard entity) are displayed in *Linked tracks*.



Linked tracks:
h001
h010
h011
h015
h021
h031
h033
h038
h046
h051

Ttrack List
T_1
T_2
T_3
T_4
T_5
T_6
T_7
T_8

Linked Entities
h001
h002
h003
h004
h006
h008
h009
h019

Chat message 1

1 black suv travels north west along road 1 block to the left of main highway center of screen

H-S association

	1	2
1	s001	h001
2	s002	h002
3	s003	h001
4	s004	h003
5	s005	h002
6	s006	h004
7	s007	h002
8	s008	h005
9	s009	h005
10	s010	h009

H-H association

	1	2
1	h001	h011
2	h002	h014
3	h003	h005
4	h004	h007
5	h005	h010
6	h006	h007
7	h007	none
8	h008	h011
9	h009	h012
10	h010	h011

Figure 44: GUI layout of the developed video-to-text fusion software prototype

There are several steps to use IFT's fusion software (GUI):

1. Load data
   IFT's V2T Fusion System requires three sets of data to perform soft-hard data fusion. The raw video data in terms of image sequence, the tracking results in kw18 format, and a text file containing chat messages. To load in these data, click File➔LoadImages, File➔LoadTracks, and File➔LoadMessages as shown in Figure 8a.

2. Play data
   Once all data are loaded, user is able to view the raw video data, the selected track one at a time, or all tracks at once by click the *Play* menu item as shown in Figure 8b. All loaded tracks are listed in Track List as shown in Figure 8c. Click on a track ID to select the track to be played. Note that *Play*➔*Selected Entity* is used to display the final concatenated track of each distinct entity after hard-hard association is performed. Figure 8d displays a screen shot of playing all tracks.

3. Entity generation
   The first step of soft-hard data fusion is to construct entities from soft and hard data. This is done through *Entity Generation*➔*From hard data* and *Entity Generation*➔*From soft data* as shown in Figure 8e. After entities are constructed, user can create a XML file to view the constructed entities. Figure 8f displays a small portion of the generated XML file in XML Notepad.

4. Entity association

   The next steps are soft-hard and hard-hard entity association and fusion. They are accessed through *Entity Association*➔*Hard-Soft* and *Entity Association*➔*Hard-Hard* as shown in Figure 9a. The results of hard-soft association and hard-hard association are displayed *in H-S association* and *H-H association* tables as shown in Figure 9b. Hard-soft entity fusion is automatically invoked during hard-soft entity association and hard-hard entity linkage is automatically carried out after hard-hard association task is finished. The result of hard-soft entity fusion is an entity with more attributes filled, which can be viewed by generating the XML file as stated in 3 above. The result of hard-hard entity linkage is a set of distinct entities listed in *Distinct Entities* listbox.

5. Concatenated tracks

   The final product of this system is a set of concatenated tracks, each is from a distinct entity listed in *Distinct Entities* listbox. To view the concatenated tracks, select a distinct entity and click *Play*➔ *Selected Entity.* This will display the concatenated tracks with a unique track ID. While the tracks are being displayed, their original track IDs will be highlighted in *Linked tracks* listbox under IFT logo. Figure 10 displays the concatenated tracks of entity 1. The left column shows the concatenated tracks of with a unique track ID = 1. The right column is the original corresponding frames with original track IDs.

4.3.2 Results

There are two major moving targets being tracked in Creech data. Their labels are 5 and 6 initially. During the course of tracking process, the checker failed to maintain the same label after the tracks were lost and picked up at a later time. This resulted in a set of tracklets of the same moving target but each has a different ID label. Figure 45 shows a screen shot illustrating this issue. The same two moving targets are now labeled as 21 and 23 respectively. To evaluate the performance of the proposed fusion method in tackling this issue, we visually examined each track and concatenated them manually and produced the ground truth for the two main moving targets. They are used to evaluate the proposed soft-hard fusion scheme. In Table I, we list all the tracks obtained from the .kw18 file associated with the Creech data in the first column. The value in each parenthesis indicates the duration of the track in seconds. Based on the quality of each track, track IDs are colored coded in green, blue, and red. The color code is provided in Table 6.

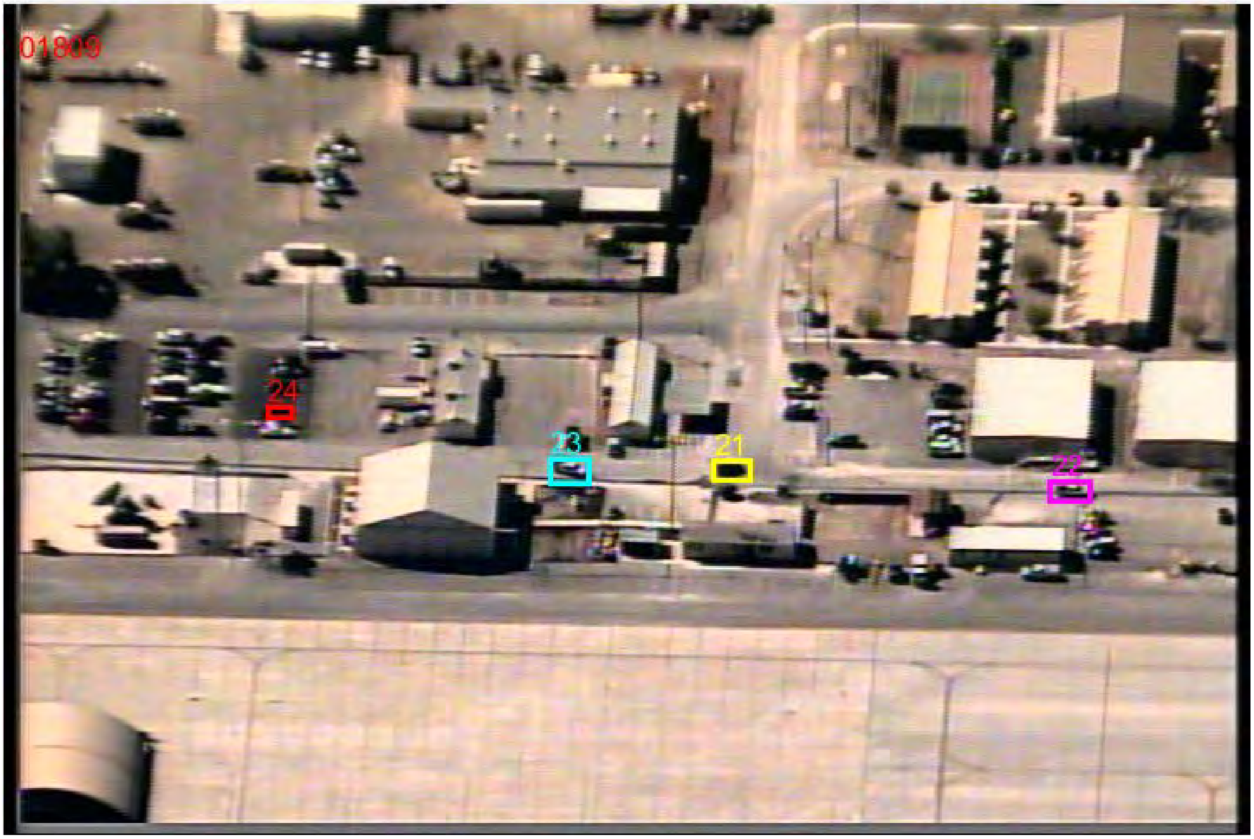


Figure 45: A screen shot illustrating the issue of the same target labeled differently during the course of tracking process. The targets with labels 21 and 23 are originally labeled as 1 and 2.

From Table 5, we observe the following:

1. For target 1, all good (green) tracks are picked up with zero false alarm rate (a good track from a wrong target is picked up) using the proposed soft-hard fusion scheme. This is compared to three good tracks (T15, T21, T 38) are missed out of 4 good tracks (T01,T15, T21, T 38) with 5 wrongly detected good tracks (T14, T18, T22, T26, and T40).
2. For target 1, considering all tracks (including green, blue, and red ones), the total length from ground truth is 153.2 seconds. With the proposed soft-hard fusion scheme, the total length of the concatenated track is 160.2 seconds, within which, 147.8 seconds are overlapped with ground truth (92.3%). This is compared to 67.6% without resorting to soft-hard fusion.
3. For target 2, 5 out of 6 good (green) tracks are picked up with zero false alarm rate using the proposed soft-hard fusion scheme. This is compared to three good tracks (T02, T17, T23) are missed out of 6 good tracks (T02,T14, T17, T19, T23, T40) with 3 wrongly detected good tracks (T18, T22, T26).
4. For target 2, considering all tracks (including green, blue, and red ones), the total length from ground truth is 173.0 second. With the proposed soft-hard fusion scheme, the total length of the concatenated track is 162.1 seconds, within which, 148.0 seconds are overlapped with ground truth (91.3%). This is compared to 14.3 % without resorting to soft-hard fusion.

Table 5: Track IDs

Track IDs of targets 1 (columns 2,3,4) and 2 (columns 5,6,7). Columns 2 and 5 are ground truths. Columns 3 and 6 are results with soft-hard fusion and columns 4 and 7 are results without resorting to soft-hard fusion.

	Target 1 Ground Truth	With S-H fusion	Without S-H fusion	Target 2 Ground Truth	With S-H fusion	Without S-H fusion
T01 (99.8)	√	√	√			√
T02 (102.5)				√	√	
T03 (15.7)						
T04 (35.5)						
T05 (60.7)						
T06 (2.0)						
T07 (1.7)						
T08 (10.9)						
T09 (6.4)						
T10 (2.6)	√	√	√			√
T11 (5.5)	√	√	√			√
T12 (1.5)			√			√
T13 (2.8)			√	√		√
T14 (1.3)			√	√	√	√
T15 (6.6)	√	√				

T16 (1.4)			√	√		√
T17 (4.8)				√	√	
T18 (4.5)			√			√
T19 (6.5)				√		
T20 (3.4)						
T21 (13.7)	√	√				
T22 (11.0)			√			√
T23 (19.0)				√	√	
T24 (7.0)						
T25 (14.1)						
T26 (5.6)			√			√
T27 (3.8)						
T28 (4.0)						
T29 (4.9)						
T30 (9.3)						√
T31 (3.1)		√				
T32 (3.2)	√		√			
T33 (1.5)		√		√		
T34 (4.1)			√		√	√
T35 (1.3)				√		
T36 (1.3)						
T37 (17.0)						
T38 (19.6)	√	√				
T39 (7.9)						
T40 (20.4)			√	√	√	√
T41 (1.6)						
T42 (1.2)						
T43 (10)			√		√	√
T44 (6.5)						
T45 (1.5)				√		
T46 (5.5)		√		√		
T47 (7.5)						
T48 (1.3)			√		√	√
T49 (10.8)			√		√	√
T50 (2.2)	√		√		√	√
T51 (2.3)		√				

Table 6: Color code used in Table 5.

COLOR	COMMENTS
GREEN	Good track
BLUE	Bounding boxes contain only a portion of the moving target, more than one moving target, or a significant fraction of tracks contains no moving target.
RED	Bounding boxes contains no valid moving target.

Table 7 summarized this result.

Table 7: Evaluation of proposed soft-hard fusion compared to without fusion

	CONSIDER GOOD TRACKS ONLY	CONSIDER ALL TRACKS
TARGET 1		
GROUND TRUTH TOTAL LENGTH	139.7 seconds	153.2 seconds
WITH FUSION TOTAL LENGTH	139.7 seconds	160.2 seconds
WITH FUSION CORRECTLY DETECTED	100%	92.3%
WITH FUSION MISSED	0 %	3.5%
WITHOUT FUSION TOTAL LENGTH	142.6 seconds	159.6 seconds
WITHOUT FUSION CORRECTLY DETECTED	70.0%	67.6%
WITHOUT FUSION MISSED	28.6%	26.0%
TARGET 2		
GROUND TRUTH TOTAL LENGTH	154.5 seconds	173.0 seconds
WITH FUSION TOTAL LENGTH	148.0 seconds	162.1 seconds
WITH FUSION CORRECTLY DETECTED	100%	91.3%
WITH FUSION MISSED	4.2%	11.9%
WITHOUT FUSION TOTAL LENGTH	142.6 seconds	180.7 seconds
WITHOUT FUSION CORRECTLY DETECTED	15.2%	14.3%
WITHOUT FUSION MISSED	86.0%	82.4%

4.4 Event Detection

Following are example results achieved using the implemented event detection system (section 3.4). Discussions are made below figures to illustrate the capability of the system.

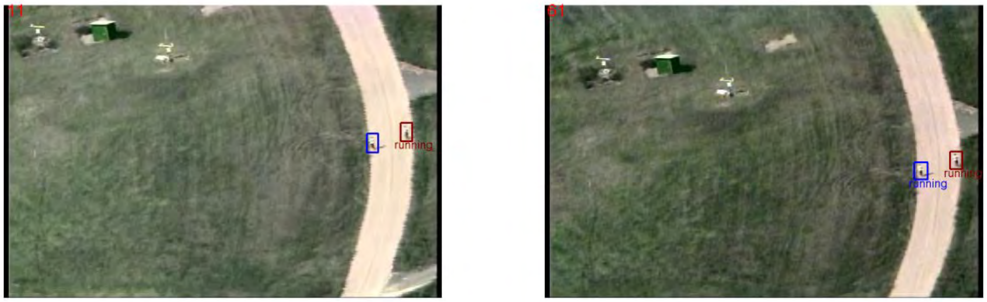
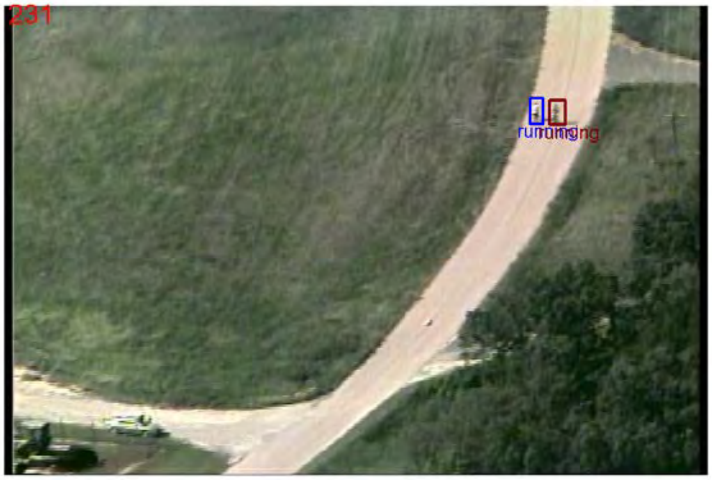


Figure 46: Multiple targets running event detection.

The system can detect multiple running events happened at the same time. As we can see, on the right hand side, the two running events are both detected. Word 'running' under the bounding box demos the event detection.



.

Figure 47: Non false detection of 'getting into car'.

These two targets are very close to each other, and one of them has the motion trend towards the other target. But since in the system we have a function module checking target identity, thus it is not miss-detected as 'getting into car'.

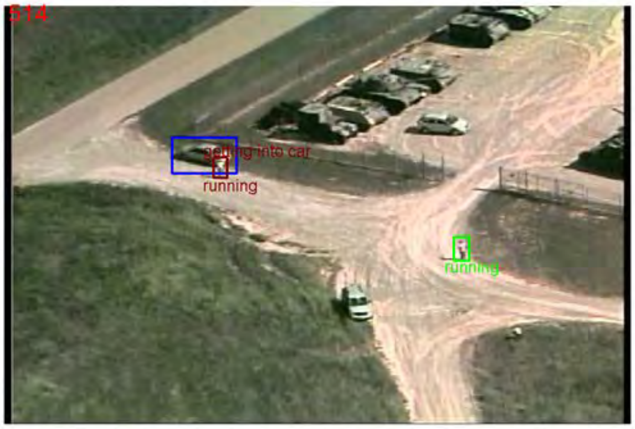


Figure 48: 'getting into car' event.

When a person target is within the distance control range of a vehicle target, it will be marked 'getting into car'. As we can see from the figure, word 'getting into car' is on top of the red bounding box.

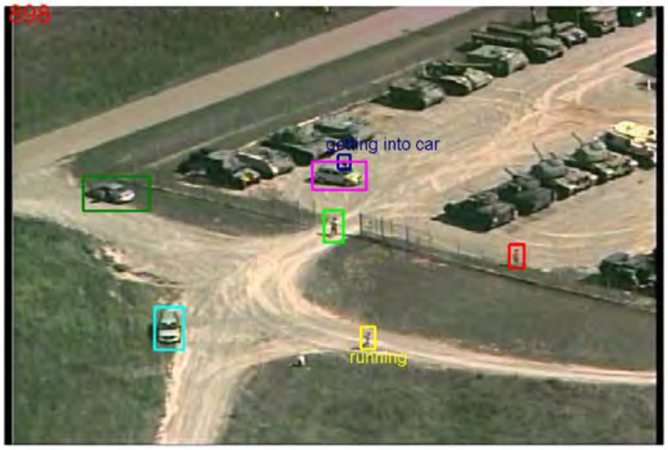


Figure 49: Another successfully detected event 'getting into car'.

5 CONCLUSIONS

The Phase I effort has resulted in a hard (video) and soft (text, voice chat) information fusion prototype to automatically generate videos with annotation that can be easily used by future human or machine users. The tracking results following the standard format (.kw18) can also be output in a separated file for interfacing with other modules in the E2AT system integration. In our implementation, each entity corresponds to one tracklet with a unique track ID. Each entity consists of two sets of attributes: common attributes and uncommon attributes. Common attributes are those which will not change over the lifetime of a target track like type and color of the target. Uncommon attributes are those changing over time like target location, direction, and activity. The same sets of attribute definitions are used for entities constructed from both hard and soft data. The association, linkage, fusion, and concatenation can improve the visual tracking results.

6 REFERENCES

[1] X. Shi, H. Ling, *et al*., "Context-Driven Moving Vehicle Detection in Wide Area Motion Imagery," In *Proc. of the Int'l Conf. on Pattern Recognition (ICPR)*, 2012.

[2] Y. Wu, J. Cheng, J. Wang, H. Lu, J. Wang, H. Ling, *et al.*, "Real-time Probabilistic Covariance Tracking with Efficient Model Update," *IEEE Trans. on Image Processing (T-IP)*, 21(5): 2824-2837, 2012.

[3] X. Mei, H. Ling, *et al*., "Minimum Error Bounded Efficient L1 Tracker with Occlusion Detection," in *Proc. of IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 1257-1264, Colorado Springs, 2011.

[4] X. Mei and H. Ling, "Robust Visual Tracking and Vehicle Classification via Sparse Representation," *IEEE Trans. on Pattern Analysis and Machine Intelligence (PAMI)*, 33(11): 2259-2272, 2011.

[5] Y. Wu, G. Chen, *et al*., "Feature-based background registration in wide-area motion imagery," in *SPIE Conf. on Defense Security+Sensing*, 8402-8403, 2012.

[6] P. Liang, G. Teodoro, H. Ling, *et al*., "Multiple Kernel Learning for Vehicle Detection in Wide Area Motion Imagery," In *Proc. of the Int'l Conf. on Information Fusion (FUSION)*, 2012.

[7] K. Palaniappan, *et al*., "Wide-Area Persistent Airborne Video: Architecture and Challenges," in *Distributed Video Sensor Networks, B. Bhanu, et al (eds)*, 2011.

[8] R. Pelapur, *et al.*, "Persistent Target Tracking Using Likelihood Fusion in Wide-Area and Full Motion Video Sequences," In *Proc. of the Int'l Conf. on Information Fusion (FUSION)*, 2012.

[9] J.L. Graham, *et al.*, "A COIN-inspired synthetic dataset for qualitative evaluation of hard and soft fusion systems," *Information Fusion (FUSION)*, 2011, pp. 1-8, 5-8, July 2011.

[10] G.L. Jacob, *et al.*, "A synthetic dataset for evaluating soft and hard fusion algorithms," *Proc. SPIE* 8062, May 23, 2011.

[11] H. Ling, and K. Okada, "An Efficient Earth Mover's Distance Algorithm for Robust Histogram Comparison," *IEEE Trans. on Pattern Analysis and Machine Intelligence (PAMI)*, 29(5): 840-853, 2007.

[12] N. Xie, H. Ling, *et al.*, "Use Bin-Ratio Information for Category and Scene Classification," in *Proc. of IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pp. 2313-2319, 2010.

[13] J. Kim, *et al.*, "A field relevance model for structured document retrieval," *Proceedings of ECIR'12 Proceedings of the 34th European conference on Advances in Information Retrieval*, pp. 97-108, 2012.

[14] Gross, Geoff A., *et al.*, "Towards hard+ soft data fusion: Processing architecture and implementation for the joint fusion and analysis of hard and soft intelligence data," *Information Fusion (FUSION)*, 2012 *15th International Conference* on. *IEEE*, 2012.

[15] J.L. Graham, *et al.*, "SYNCOIN: A synthetic data set for evaluation of hard and soft fusion systems," *Proc. of the 14th international conference on information fusion, 2011,* Chicago, IL.

[16] J.L. Graham, *et al.*, "A new synthetic dataset for evaluating soft and hard fusion algorithms," *SPIE proceedings*, Fl, USA.

[17] M.S. Baran, *et al.*, "Hard sensor fusion for COIN inspired situation awareness," *Proc. of the 14th international conference on information fusion*, 2011, Chicago, IL.

[18] I. Ersoy, K. Palaniappan, *et al.*, “Visual tracking with robust target localization,” *IEEEInt. Conf. Image Processing*, 2012.

[19] S. Candemir, K. Palaniappan, *et al.*, “Feature prominence-based weighting scheme for video tracking,” *ICVGIP*, 2012.

[20] R. Pelapur, K. Palaniappan, *et al.*, “Robust orientation and appearance adaptation for wide-area large format video object tracking,” *9th IEEE Int. Conf. Advanced Video and Signal-Based Surveillance (AVSS)*, 2012.

[21] I. Ersoy, K. Palaniappan, *et al.*, “Interactive tracking for persistent wide-area surveillance,” *Proc. SPIE Conf. Geospatial InfoFusion II, SPIE Defense, Security and Sensing: Sensor Data and Information Exploitation*, Vol. 8396, April 2012.

[22] S. Candemir, K. Palaniappan, *et al.*, “Feature fusion using ranking for object tracking in aerial imagery,” *Proc. SPIE Conf. Geospatial InfoFusion II, SPIE Defense, Security and Sensing: Sensor Data and Information Exploitation*, Vol. 8396, April 2012.

[23] http://cmusphinx.sourceforge.net/

[24] http://worldwind.arc.nasa.gov/java/

[25] AFRL: Columbus large image format (CLIF) 2006. https://www.sdms.afrl.af.mil/index.php?collection=clif2006.

[26] L. Breiman. Random forests. Machine learning, 2001; 45(1): 5–32.

[27] Flickner, M.; Sawhney, H.; Niblack, W.; Ashley, J.; Huang, Q.; Dom, B.; Gorkani, M.; Hafner, J.; Lee, D.; Petkovic, D.; Steele, D. & Yanker, P., “Query by Image and Video Content: The QBIC System Computer,” *IEEE Computer Society Press*, 1995, 28, 23-32

[28] T. Hofmann, “Learning the Similarity of Documents: an information-geometric approach to document retrieval and categorization,” *Advances in Neural Information Processing Systems* 12, pp-914-920, MIT Press, 2000.

[29] H. Ling, Y. Wu, E. Blasch, G. Chen, H. Lang, and L. Bai. "Evaluation of Visual Tracking in Extremely Low Frame Rate Wide Area Motion Imagery." In Proc. of the *Int'l Conf. on Information Fusion (FUSION)*, 2011.

[30] H. Ling and K. Okada, "Diffusion Distance for Histogram Comparison", in *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, I:246-253, New York, NY, USA, June 2006.

[31] H. Ling, L. Bai, E. Blasch, and X. Mei, “Robust Infrared Vehicle Tracking across Target Pose Change using L1 Regularization,” Proc. of the *International Conference on Information Fusion* (*FUSION*), Edinburgh, UK, 2010.

[32] H. Ling and D.W. Jacobs, ``Shape Classification Using the Inner-Distance", *IEEE Trans. on Pattern Analysis and Machine Intelligence (PAMI)*, 29(2):286-299, 2007.

[33] H. Ling, X. Yang, and L.J. Latecki. ``Balancing Deformability and Discriminability for Shape Matching", in *ECCV*, 2010.

[34] H. Ling and S. Soatto, ``Proximity Distribution Kernels for Geometric Context in Category Recognition", in *ICCV*, 2007.

[35] D.G. Lowe, "Distinctive image features from scale-invariant keypoints," *International Journal of Computer Vision*, vol. 60, pp. 91-110, 2004.

[36] C. Lu, N. Adluru, H. Ling, G. Zhu, and L.J. Latecki. ``Contour Based Object Detection Using Part-Bundles," *Computer Vision and Image Understanding (CVIU)*, 114(7):827-834, 2010.

[37] C. Lu, L.J. Latecki, N. Adluru, X. Yang, and H. Ling, ``Shape Guided Contour Grouping with Particle Filters", in Proc. of the *IEEE Int'l Conf. on Computer Vision (ICCV)*, pp. 2288-2295, 2009.

[38] R. Porter, D. Hush, and A. Fraser, "*Narrowing the Semantic Gap in Wide Area Motion Imagery*", *IEEE Signal Processing Magazine*, 27(5): pp. 56-65, 2010.

[39] Rubner Y, Tomasi C, Guibas LJ. "The Earth Mover's Distance as a Metric for Image Retrieval." *Int. J. Comput. Vision*, 40(42):99-121, 2000.

[40] X. Shi, X. Zhang, Y. Liu, W. Hu, and H. Ling. ``Multi-cue Based Multi-target Tracking Using Online Random Forests," in Proc. *Int'l Conf. on Acoustics, Speech and Signal Processing (ICASSP)*, 2011.

[41] Q. Wei, X. Zhang, W. Hu, and H. Ling. ``Compact Visual Codebook for Action Recognition." In Proc. of *IEEE Int'l Conf. on Image Processing (ICIP)*, 3805--3808, Hong Kong, China, 2010.

[42] Q. Wei, X. Zhang, Y. Kong, W. Hu, and H. Ling. ``Group Action Recognition Using Space-Time Interest Points." In Proc. of the *5th Int'l Symposium on Visual Computing (ISVC),* Vol. 2, pp. 757--766, 2009.

[43] G. Chen, D. Shen, C. Kwan, J. B. Cruz, Jr., M. Kruger, and E. Blasch, “Game Theoretic Approach to Threat Prediction and Situation Awareness,” *Journal of Advances in Information Fusion*, Vol. 2, No. 1, June 2007, pp. 35-48.

[44] L. S. Shapley, "Stochastic games," in Proceedings of the *National Academy of Sciences of the United States of America*, vol. 39, pp. 1095-1100, 1953.

[45] H.K. SAWANT and DIPALI KADAM, “Context Based Approaches to Learn Text and Image Association and Processing Semantics,” www.ejournal.aessangli.in/ASEEJournals/IT17.doc.

[46] R. Yager (Machine Intelligence Institute, Iona College). Conditional Approach to Possibility-Probability Fusion. 2011. 30p Report No.: MII-3021R.

[47] G. Gross, R. Nagi, and K. Sambhoos, “Soft information, dirty graphs and uncertainty representation/processing for situation understanding,” *The 13th International Conference on Information Fusion*. 2010: Edinburgh, Scotland.

[48] D. Shen, H. Xu, E. Blasch, K. Pham, Z. Wang, H. Ling, G. Chen, “A holistic image segmentation framework for cloud detection and extraction,” To appear in *SPIE Defense, security and sensing*, 2013.

[49] X. Shi, E. Blasch, W. Hu, and H. Ling. "Using Maximum Consistency Context for Multiple Target Association in Wide Area Traffic Scenes", under review.

[50] F. Bunyak, K. Palaniappan, S. K. Nath, and G. Seetharaman,“Flux tensor constrained geodesic active contours with sensorfusion for persistent object tracking,” *J. Multimedia*, vol. 2, no.4, pp. 20–33, August 2007.

[51] K. Palaniappan, F. Bunyak, P. Kumar, I. Ersoy, S. Jaeger, K. Ganguli, A. Haridas, J. Fraser, R. Rao, G. Seetharaman, “Efficient feature extraction and likelihood fusion for vehicle tracking in low frame rate airborne video”, *13thInt. Conf. Information Fusion,* Edinburgh, UK, July 26-29, 2010

[52] K. Palaniappan, I. Ersoy, G. Seetharaman, S.R. Davis, P. Kumar, R. M. Rao, R. Linderman, “Parallel flux tensor analysis for efficient moving object detection”,*14thInt. Conf. Information Fusion*, Chicago, 2011.

[53] R. Pelapur, S. Candemir, M. Poostchi, F. Bunyak, R. Wang, G. Seetharaman, K. Palaniappan, “Persistent target tracking using likelihood fusion in wide-area and full motion video sequences”, *15th Int. Conf. Information Fusion,*Singapore, July 9-12, 2012.

[54] C.R. Shyu, M. Klaric, G. Scott, A. Barb, C. Davis, K. Palaniappan, “GeoIRIS: Geospatial information retrieval and indexing system – Content mining, semantics, modeling, and complex queries”, *IEEE Trans. Geoscience and Remote Sensing*, Vol. 45, No. 4, Apr 2007, pp. 839-852. Special issue on Image Information Mining for Earth Observation Data.

[55] M. Klaric, G. Scott, C.R. Shyu, C. Davis, K. Palaniappan, “A framework for geospatial satellite imagery retrieval systems”, *IEEE Int. Geoscience and Remote Sensing Symposium*, Denver, CO, Jul 31- Aug 4, 2006, pp. 2457-2460.

[56] C. R. Shyu, G. Scott, M. Klaric, C. H. Davis, K. Palaniappan, “Automatic object extraction from full differential morphological profile in urban imagery for efficient object indexing and retrievals”, *3rd Int. Symposium on Remote Sensing and Data Fusion Over Urban Areas* (URBAN 2005), International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences, Vol 36, No 8/W27, Tempe, AZ, Mar 14-16, 2005.

[57] D.Tuia, F. Pacifici, M. Kanevski, and W. J. Emery, Classification of Very High Spatial Resolution Imagery Using Mathematical Morphology and Support Vector Machines, IEEE Trans. Geoscience and Remote Sensing, 47(11): 3866-3879, Nov. 2009.

[58] Z. Wang, F. Wu, and Z. Hu. MSLD: A robust descriptor for line matching. Pattern Recognition, 42: 941-953, 2009.

[59] Beril Sırmaçek, Cem Ünsalan, Urban Area Detection Using Local Feature Points and Spatial Voting*, IEEE Geoscience and Remote Sensing Letters*, 7(1):146 - 150, 2010.

[60] A. Hafiane, G. Seetharaman, K. Palaniappan, B. Zavidovique, “Rotationally invariant hashing of median binary patterns for texture classification”, *Lecture Notes in Computer Science* (ICIAR), Vol. 5112, 2008, pp. 619-629.

[61] Z. Guo, L. Zhang, and D. Zhang, “A completed modeling of local binary pattern operator for texture classification,” *IEEE Trans. Image Processing*, vol. 19, no. 6, pp. 1657–1663, Jun 2010.

[62] S. Liao, M. W. K. Law, and A. C. S. Chung, “Dominant local binary patterns for texture classification,” *IEEE Trans. Image Processing*, vol. 18, no. 5, pp. 1107–1118, May 2009.

[63] S. Lazebnik, C. Schmid, and J. Ponce, “A sparse texture representation using local affine regions,” *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 27, no. 8, pp. 1265–1278, Aug 2005.

[64] M. Varma and A. Zisserman, “A statistical approach to material classification using image patch exemplars.” *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 31, no. 11, pp. 2032–2047, Nov 2009.

[65] J. Chen, S. Shan, C. He, G. Zhao, M. Pietikainen, X. Chen, and W. Gao, “WLD: A robust local image descriptor,” *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 32, no. 9, pp. 1705–1720, Sep 2010.

[66] J. Peng, C. Barbu, G. Seetharaman, F. Wei, X. Wu, K. Palaniappan, “ShareBoost: Boosting for multi-view learning with performance guarantees”, *Lecture Notes in Computer Science* (ECML PKDD European Conf. of Machine Learning and Principles of Knowledge Discovery in Databases), 2011.

[67] S. K. Nath, K. Palaniappan, “Fast graph partitioning active contours for image segmentation using histograms”, *EURASIP Journal on Image and Video Processing,* 9p., 2009.

[68] S. K. Nath, K. Palaniappan, F. Bunyak, “Accurate spatial neighborhood relationships for arbitrarily-shaped objects using Hamilton-Jacobi GVD”, *Lecture Notes in Computer Science* (SCIA), Vol. 4522, 2007, pp. 421-431.

[69] P. Matsakis, J. M. Keller, O. Sjahputera, and J. Marjamaa, “The use of force histograms for affine-invariant relative position description,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 26, no. 1, pp. 1–18, Jan. 2004.

[70] L. Zebedin, J. Bauer, K. Karner, and H. Bischof. Fusion of feature- and area-based information for urban buildings modeling from aerial imagery. In *ECCV*, 2008.

[71] D. Aiger, N. J. Mitra, and D. Cohen-Or. 4-Points congruent sets for robust pairwise surface registration. *ACM Trans. Graphics* (*Proceedings of SIGGRAPH*), 2008.

[72] I. N. Junejo, H. Foroosh, GPS coordinates estimation and camera calibration from solar shadows, *CVIU*, 114:991-1003, 2010.

[73] S. Gu, Y. Zheng and C. Tomasi, Critical Nets and Beta-Stable Features for Image Matching, *ECCV* 2010.

[74] E. Blasch, G. Seetharaman, M. Talbert, K. Palaniappan, H. Ling, Key Elements to Support Layered Sensing Dismount Tracking*, NATO Workshop on Detection and Surveillance of Dismounted Combatants from Airborne Platforms* (SET-178), RTO-IST-086, Sept. 2011.

[75] A. Rice and J. Vasquez, “Context-Aided Tracking with an Adaptive Hyperspectral Sensor,” *Int. Conf. on Info Fusion - Fusion11*, 2011.

[76] Y. Zhao, X. Zhou, K. Palaniappan, X. Zhuang, “Statistical modeling for improved land cover classification”, *SPIE Battlespace Digitization and Network-Centric Warfare II,* Vol. 4741, Ed. Raja Suresh, W.E. Roper, Orlando, FL, Aug 2002, pp. 296-304.

[77] K. Palaniappan, F. Zhu, X. Zhuang, Y. Zhao, and A. Blanchard, “Enhanced binary tree genetic algorithm for automatic land cover classification”, *IEEE 2000 Int. Geoscience and Remote Sensing Symposium (IGARSS)*, Honolulu, Hawaii, July 24-28, 2000, Vol. II, pp. 688-692.

[78] Y. Zhu, Y. Zhao, K. Palaniappan, X. Zhou, X. Zhuang, “Optimal Bayesian classifier for land cover classification using Landsat TM data”, *IEEE 2000 Int. Geoscience and Remote Sensing Symposium (IGARSS)*, Honolulu, Hawaii, July 24-28, 2000, Vol. I, pp. 447-450.

[79] M. Poostchi, K. Palaniappan, F. Bunyak, MichelaBecchi, G.Seetharaman, “Realtime motion detection based on the spatio-temporal median filter using GPU integral histograms”, *ICVGIP*, Bombay, Dec 16-19, 2012.

[80] B. Z. Yao, X. Yang, L. Lin, M. W. Lee, S.C. Zhu, I2T: Image Parsing to Text Description, Proc. *IEEE*, 98(8):1485-1508, 2010.

[81] Junseok Kwon and Kyoung Mu Lee, “Visual Tracking Decomposition,” *IEEE Conference on Computer Vision and Pattern Recognition* (CVPR) 2010.

[82] R. Pelapur, S. Candemir, F. Bunyak, M. Poostchi, G. Seetharaman and K. Palaniappan, "Persistent target tracking using likelihood fusion in wide-area and full motion video sequences", *15th International Conference on Information Fusion (FUSION)*, 2012.

[83] Y. Sheikh, S. Khan, M. Shah, and R. Cannata, “Geodetic alignment of aerial video frames,” in *Video Registration*, Chapter 7, 2003.

[84] G. Zhou, C. Li, and P. Cheng, “Unmanned aerial vehicle (UAV) real-time video registration for forest fire monitoring,” in *IEEE Geoscience and Remote Sensing Symposium*, pp. 1803 – 1806, 2005.

[85] L. G. Brown, “A survey of image registration techniques,” *ACM Comput. Surv.* 24(4), pp. 325–376, 1992.

[86] B. Zitov and J. Flusser, "Image registration methods: A Survey," *Image and Vision Computing* 21(11), pp. 977–1000, 2003.

[87] F. Perlant and D. McKeown, "Scene registration in aerial image analysis," *PhEngRS* 56(4), pp. 481–493, 1990.

[88] C. Shekhar, "Semi-automatic video-to-site registration for aerial monitoring," in ICPR'00: Proceedings of *the International Conference on Pattern Recognition*, pp. 736–739, 2000.

[89] K. Mikolajczyk and et. al , "A comparison of affine region detectors," Int. *J. Computer Vision* 65(1/2), pp. 43–72, 2005.

[90] F. Bunyak, K. Palaniappan, S. Nath, and G. Seetharaman, "Geodesic active contour based fusion of visible and infrared video for persistent object tracking," in 8th *IEEE Workshop on Applications of Computer Vision* (WACV 2007), p. Online, (Austin, TX), Feb. 2007.

[91] F. Bunyak, K. Palaniappan, S. Nath, and G. Seetharaman, "Fux tensor constrained geodesic active contours with sensor fusion for persistent object tracking," *J. Multimedia* 2, pp. 20–33, August 2007.

[92] G. Seetharaman, G. Gasperas, and K. Palaniappan, "A piecewise affine model for image registration in 3-d motion analysis," in *IEEE Int. Conf. On Image Processing*, pp. 561–564, (Vancouver, BC, Canada), Sep. 2000.

[93] L. Zhou, C. Kambhamettu, D. Goldgof, K. Palaniappan, and A. F. Hasler, "Tracking non-rigid motion and structure from 2D satellite cloud images without correspondences," *IEEE Trans. Pattern Analysis and Machine Intelligence* 23, pp. 1330–1336, Nov. 2001.

[94] R. Hartley and A. Zisserman, Multiple View Geometry in Computer Vision. Cambridge University Press, 2000.

[95] H. Wang, D. Mirota, and G. Hager, "A generalized kernel consensus-based robust estimator," *Pattern Analysis and Machine Intelligence, IEEE Transactions* on 32(1), pp. 178–184, 2010.

[96] S. Nath and K. Palaniappan, "Adaptive robust structure tensors for orientation estimation and image segmentation," in *LNCS-3804: Proc. ISVC'05*, pp. 445–453, (Lake Tahoe, Nevada), Dec. 2005.

[97] H. Nagel and A. Gehrke, "Spatiotemporally adaptive estimation and segmentation of OF-Fields," in *LNCS-1407: ECCV98, 2*, pp. 86–102, Springer-Verlag, (Freiburg, Germany), June 1998.

[98] B. Horn and B. Schunck, "Determining optical flow," *Artificial Intelligence* 17, pp. 185–203, Aug. 1981.

[99] K. Palaniappan, H. Jiang, and T. I. Baskin, "Non-rigid motion estimation using the robust tensor method," in *IEEE Comp. Vision and Pattern Recog. Workshop on Articulated and Nonrigid Motion*, (Washington, DC), June 2004.

[100] J. Zhang, J. Gao, and W. Liu, "Image sequence segmentation using 3-D structure tensor and curve evolution," *Circuits and Systems for Video Technology, IEEE Transactions on* 11, pp. 629–641, May 2001.

[101] M. Pravia, O. Babko-Malaya, M. Schneider, J. White, C. Chong and A. Willsky, "Lessons learned in the creation of a data set for hard/soft information fusion," IEEE International Conference on Information Fusion, pp.2114-2121, 2009.

[102] T. Wu and W. Pottenger, "A semi-supervised active learning algorithm for information extraction from textual data: Research Articles," *Journal of the American Society for Information Science and Technology - Intelligence and Security Informatics*, vol. 56, no. 3, pp. 258-271, 2005.

[103] B. Liu, E. Blasch, Y. Chen, D. Shen, G. Chen, “Scalable sentiment classification for Big Data analysis using Naïve Bayes Classifier,” *IEEE International Conference on Big Data*, pp.99-104, 2013.

[104] A. Preece, D. Pizzocaro, D. Braines, D. Mott, G. de Mel and P. Tien, “Integrating hard and soft information sources for D2D using controlled natural language," *International Conference on Information Fusion*, pp.1330-1337, 2012.

[105] E. Blasch, A. Steinberg, S. Das, J. Llinas, C. Chong, O. Kessler, E. Waltz, and F. White, “Revisiting the JDL model for information exploitation,” *International Conference on Information Fusion*, 2013.

[106] E. Blasch, Y. Chen, G. Chen, D. Shen, and R. Kohler, “Information fusion in a cloud-enabled environment,” High Performance Semantic Cloud Auditing, Springer Publishing, 2013.

[107] B. Liu, Y. Chen, E. Blasch, K. Pham, D. Shen, and G. Chen, “A holistic cloud-enabled robotics system for real-time video tracking application,” *International Workshop on Enhanced Cloud Fusion*, in conjunction with Future Information Tech, 2013.

[108] J. Dean and S. Ghemawat, “Mapreduce: simplified data processing on large clusters,” in *Proceedings of the IEEE conference on Symposium on Operating Systems Design & Implementation*, vol.6, 2004 Tech, 2013.

[109] R. I. Hammoud, C. S Sahin, E. P. Blasch, B. J. Rhodes and T. Wang, “Automatic association of chats and video tracks for activity learning and recognition in aerial video surveillance, ” *Sensors*, 2014 Oct 22;14(10):19843-60.

LIST OF ACRONYMS, ABBREVIATIONS, AND SYMBOLS

ACRONYM	**Description**
AGP	Accelerated Proximal Gradient
BReT	Blur Resilient Target Tracking
CLIF	Columbus Large Image Format
CSURF	Clustered set of Structured Uniformly dense Robust Features tracker
CT	Compressive Tracker
E2AT	Enhanced Exploitation and Analysis Tools
FMV	Full Motion Video
GATER	Government Algorithms for Tracking Exploitation Research
GFS	Google File System
GPR	Government Purpose Rights
GUI	Graphical User Interface
HDFS	Hadoop Distributed File System
IFT	Intelligent Fusion Technology, Inc
IMINT	Image Intelligence
LoFT	Likelihood of Features Tracker
MHT	Multiple Hypothesis Tracking
MIL	Multiple Instance Learning
NLP	Natural Language Processing
OAB	Online AdaBoost
OF	Optic Flow
RANSAC	Random sample consensus
TLD	Tracking-Learning-Detection
WAMI	Wide Area Motion Imagery
WFC	Work Flow Controller
UAV	Unmanned Aerial Vehicles
VTD	Visual Tracking Decomposition