Adaptive Stream Mining: A Novel Dynamic Computing Paradigm for Knowledge Extraction

AFOSR DDDAS Program
PI Meeting Presentation

Pls: Shuvra S. Bhattacharyya, University of Maryland
Mihaela van der Schaar, UCLA

Email: ssb@umd.edu, mihaela@ee.ucla.edu

January 29, 2016, Arlington, VA
Talk Outline

- ASMDF Project overview: design and implementation of Adaptive Stream Mining systems using DataFlow methods
- Lightweight dataflow
- Multi-objective design optimization in the *lightweight dataflow for DDDAS environment* (LiD4E)
- Dataflow model detection
- Application area: tracking networks using mobile devices (with T. Damarla, ARL, and W. Stechele, T. U. Munich)
- Application area (emerging work) → Multispectral video processing (with E. Blasch, AFRL)
- Summary
DDDAS Paradigm Applied to ASM

- **Design Space**
  - Classifier topologies
  - Dataflow graph schedules
  - Platform configurations
  - Network attributes

- **Models**
  - Dataflow models for design
  - Classifier models for computation and classification
  - Scheduling models for mapping and distribution
  - Simulation models for behavior prediction and analysis

- **Algorithms**
  - Machine learning algorithms
  - Scheduling algorithms
  - Signal processing algorithms

- **Applications**
  - Multimedia processing
    - Surveillance
    - Cyber-Security
    - Intelligent traffic control
    - Seismic monitoring
    - Online Financial analysis
Dataflow-based Design for Embedded Systems

- A variety of development environments is based on **dataflow models of computation**.
  - Applications are designed in terms of stream processing block diagrams.
- By using these design tools, an application designer can
  - Develop complete functional specifications of model-based components.
  - Verify functional correctness through model-based simulation and verification.
  - Implement the designs on embedded platforms through supported platform-specific flows.
DSP-Oriented Dataflow Modeling

- Motivated by the diversity and increasing relevance of model-based design tools for embedded signal and information processing, our research emphasizes
  - Abstraction of relevant models and methods
  - Experimentation with and optimization of new model-based methods in the context of relevant stream mining applications
- Signal flow diagrams as dataflow graphs
- Emphasis on characterization of production and consumption rates
  - Static constants $\rightarrow$ synchronous dataflow
  - Constant periodic patterns $\rightarrow$ cyclo-static dataflow
  - Port-controlled dynamic behavior $\rightarrow$ Boolean dataflow
  - Dynamically parameterized rates $\rightarrow$ parameterized dataflow
  - Mode-based dynamic behavior $\rightarrow$ core functional dataflow (and many others)
- Large library of algorithms for graph analysis and graph-based design optimization ("transformations")
- Co-design of dataflow models and transformations
Design Component (Actor)
Design in Lightweight Dataflow

- Actor design in terms of statically or dynamically determined transitions through (parametric) synchronous dataflow modes
- System design in terms of FSM/dataflow compositions

(a) Example of a CFDF graph.
(b) Dataflow table for actor X.
(c) Dataflow table for actor Y.
(d) Mode transition graph for actor X.
(e) Mode transition graph for actor Y.
Lightweight Dataflow APIs for Actor Implementation

- **Construct** and **Terminate** functions → instantiate and remove actors in a dataflow graph
- **Enable** function:
  - Returns a Boolean value indicating whether or not the given actor can be executed (“fired”) in its *next mode*
  - → checks for sufficient data on the input edges, and sufficient empty space
- **Invoke** function: executes an actor according to its designated next mode
  - Produces/consumes data from incident edges
  - Does so without any blocking reads or blocking writes
  - Updates the next mode of the actor
- It is *not* always necessary to call the enable function before the invoke function
  - Calls can be “bypassed” at run-time if the corresponding conditions are guaranteed through other forms of analysis
  - Various methods for static, dynamic, and hybrid static/analysis can be applied for streamlining use of the enable function
boolean lide_c_inner_prod_enable(
    lide_c_inner_prod_context_type *context) {
    boolean result = FALSE;

    switch (context->mode) {
    case LIDE_C_INNER_PROD_MODE_STORE_LENGTH:
        result = lide_c_fifo_population(context->m) >= 1;
        break;
    case LIDE_C_INNER_PROD_MODE_PROCESS:
        result = (lide_c_fifo_population(context->x) >=
                  context->length) &&
                  (lide_c_fifo_population(context->y) >=
                  context->length) &&
                  ((lide_c_fifo_population(context->out) <
                    lide_c_fifo_capacity(context->out)));
        break;
    default:
        result = FALSE;
        break;
    }
    return result;
Some Useful Features of Lightweight Dataflow

- **Abstract, “lightweight’’ APIs** that can be retargeted across different platform-oriented languages (e.g., C, C++, CUDA, OpenCL, Verilog, VHDL, …) to provide a unified, cross-platform framework for model-based design

- **Orthogonalization** across system-level design concerns (e.g., dataflow graph scheduling and memory management), and actor implementation

- Natural connection to many application areas of stream mining and signal & information processing

- Capability to naturally express and efficiently exploit coarse grain parallelism

- Facilitates investigation of dataflow graph transformations for system level optimization
Talk Outline

• ASMDF Project overview: design and implementation of Adaptive Stream Mining systems using DataFlow methods
• Lightweight dataflow
• Multi-objective design optimization in the lightweight dataflow for DDDAS environment (LiD4E)
• Dataflow model detection
• Application area: tracking networks using mobile devices (with T. Damarla, ARL, and W. Stechele, T. U. Munich)
• Application area (emerging work) → Multispectral video processing (with E. Blasch, AFRL)
• Summary
ASM Multiobjective Design Optimization (AMDO) Framework

• Motivated by complex multidimensional design evaluation spaces
  • Real-time performance: e.g., latency and throughput
  • Stream mining quality: e.g., accuracy and false positive rate
  • Energy efficiency: e.g., peak and average power consumption

• ASM multiobjective design optimization (AMDO) framework
  • Model-based design approach for data-driven multi-mode (MM) system design
  • Provides capabilities for exploring multidimensional design evaluation spaces in ASM system implementation
  • Inherits (from our earlier work in the project) design process in terms of adaptation state machine $S_{MM}$
  • Introduces parameterization of $S_{MM}$
    • AMDO design space parameter set $P = (p_1, p_2, \ldots, p_K)$
  • Different parameter configurations for $P$ lead to different ways in which data-driven adaptation is controlled, and
  • … in which multidimensional design evaluation metrics are traded off throughout the execution process
AMDO System Design Model

- System designed as a set of mutually exclusive application modes
  \[ S_M = \{\mu_1, \mu_2, \ldots, \mu_N\} \]
  - \( \mu_i \): set of application systems active during a corresponding mode of operation
  - Actor-, application-, and schedule-level parameter configurations are associated with \( \mu_i \)
- Set of measurements, \( M = m_1, m_2, \ldots, m_k \)
  - From I/O, platform, operating environment, …
  - \( m_i \): a distinct metric \( \rightarrow \) instantaneous power consumption, remaining battery capacity, etc.,
- Measurement vectors: \( m_1(i), m_2(i), \ldots, m_k(i) \) from application level instrumentation
  - Drive the multi-mode (MM) state machine \( S_{MM} \)
- Functionality of specific application modes is represented using dataflow models of computation — i.e., FSM/dataflow compositions in the form of “HCFDF”
- AMDO system modeled as a tuple: \( \alpha = (S_{MM}, P, T) \)
  - State machine, parameterization, performance assessment actor (PAA) set
- State machine parameterization \( \rightarrow \) Alternative parameterizations provide for static configuration or data-driven adaptation across multidimensional design evaluation metrics (different regions of the design space)
Example of FSM Parameterization

- FSM parameterization vector, $P = \{p_1, p_2, p_3, p_4, p_5\}$
- $p_1$: deadline for processing each image;
- $p_2$: deadline miss tolerance: the percentage of deadlines that can be missed before the system is considered to be “underperforming”;
- $p_3$: execution time tolerance factor: overperformance if average execution time is less than $p_1 \times p_3$;
- $p_4$: threshold for overperformance with respect to battery capacity (%);
- $p_5$: threshold for underperformance with respect to battery capacity (%).
AMDO-Integrated Design and Implementation

AMDO

Multi-mode (MM) system design
  - Auxiliary components
  - Algorithms
  - Application

Dataflow modeling (HCFDF)

LiD4E

Design environment

Optimization/simulation environment

Parameterization

Instrumentation

PAA Set

S_{MM}

User specified objectives, design requirements, platform specifications

Analyze Pareto optimized design configurations; provide feedback to refine parameters, instrumentation, and objectives
Case Study: Multi-class Vehicle Classification

Android Nexus 7

- Multi-class classifier 1
- Multi-class classifier 2
- Multi-class classifier 3

Profiling data from target platform

PC-based AMDO Simulation Tool

Simulated environment implemented with ASM multi-objective design optimization (AMDO) framework

Buses

Cars

Vans
Pareto Analysis

- Multiobjective Pareto analysis
  - Complex systems are difficult to optimize across the entire objective space
  - Conventionally, some objectives are fixed (static) and the system is optimized for a single objective
  - A Pareto optimal design (among some set of “candidate designs”) is one such that improvement in one dimension results in degradation in one or more other dimensions

- Pareto analysis using the AMDO framework
  - Run-time selection of the most strategic operational point for the present operational scenario
  - Dynamic selection from within the Pareto optimal set of designs based on the relevant operational constraints and objectives
  - Flexible framework for adapting constraints and objectives while the system is running
The AMDO approach achieves competitive solutions at extremes, while allowing for intensive exploration of “in-between” points.

LID4E provides a systematic framework for system design and implementation based on the AMDO approach.
Example of FSM Parameterization

- FSM parameterization vector, \( P = \{p_1, p_2, p_3, p_4, p_5\} \)
- \( p_1 \): deadline for processing each image;
- \( p_2 \): deadline miss tolerance: the percentage of deadlines that can be missed before the system is considered to be “underperforming”;
- \( p_3 \): execution time tolerance factor: overperformance if average execution time is less than \( p_1 \times p_3 \);
- \( p_4 \): threshold for overperformance with respect to battery capacity (\%);
- \( p_5 \): threshold for underperformance with respect to battery capacity (\%).
Talk Outline

• ASMDF Project overview: design and implementation of Adaptive Stream Mining systems using DataFlow methods
• Lightweight dataflow
• Multi-objective design optimization in the lightweight dataflow for DDDAS environment (LiD4E)
• Dataflow model detection
• Application area: tracking networks using mobile devices (with T. Damarla, ARL, and W. Stechele, T. U. Munich)
• Application area (emerging work) → Multispectral video processing (with E. Blasch, AFRL)
• Summary
Recap: DSP-Oriented Dataflow Modeling

• Motivated by the diversity and increasing relevance of model-based design tools for embedded signal and information processing, our research emphasizes
  • Abstraction of relevant models and methods
  • Experimentation with and optimization of new model-based methods in the context of relevant stream mining applications
• Signal flow diagrams as dataflow graphs
• Emphasis on characterization of production and consumption rates
  • Static constants → synchronous dataflow
  • Constant periodic patterns → cyclo-static dataflow
  • Port-controlled dynamic behavior → Boolean dataflow
  • Dynamically parameterized rates → parameterized dataflow
  • Mode-based dynamic behavior → core functional dataflow (and many others)
• Large library of algorithms for graph analysis and graph-based design optimization ("transformations")
• Co-design of dataflow models and transformations
Dataflow Model Detection

- Set of all actors in a dataflow based design (DBD)
  \[ \mathcal{A} = \{a_1, a_2, \ldots, a_n\} \]  \hspace{1cm} (1)

- Set of available models.
  \[ \mathcal{M} = \{m_1, m_2, \ldots, m_z\} \]  \hspace{1cm} (2)
  where \( m_i \)'s are ordered in increasing generality (\( m_a \) is more restrictive compared to \( m_b \) whenever \( a < b \)).

- \( \mathcal{A}_k \subset \mathcal{A} \): set of actors in \( \mathcal{A} \) that conform to model \( m_k \).

- Because \( \mathcal{M} \) is ordered in increasing generality that leads to \( \mathcal{A}_k \subset \mathcal{A}_l \) for \( k < l \).

- Most specialized model (MSM) for an actor \( a \):
  \[ \text{MSM}(a) = \min\{i \mid m_i \in \mathcal{R}(a)\}. \]  \hspace{1cm} (3)
  where \( \mathcal{R}(a) \) is the set of all models in \( \mathcal{M} \) that actor \( a \) conforms to.
Transforming Legacy Code into LIDE

- Model detection works on LIDE dataflow graphs
- Converting legacy code to LIDE-compatible format
  - Add FIFO for each input
  - Create `enable()` function that returns TRUE when FIFOs have sufficient numbers of tokens, FALSE otherwise
  - Set `invoke()` function to call the original legacy function

```
Original function

fnc(in1,in2,out1,out2)
{
  ...
  ...
  ...
}
```

```
Inputs/outputs replaced with buffers

fnc(in1,in2,out1,out2)
{
  ...
  ...
  ...
}
```

```
LIDE block

Set `invoke()` to `fnc()`

`enable()` returns TRUE if input buffers contain enough tokens to fire
```
Generate input to achieve greatest code coverage.

Analyze consumption/production rates using pattern matching algorithms to determine dataflow model.
CMS Level 1 Trigger System

Compact Muon Solenoid (CMS) is a large particle physics detector built on the Large Hadron Collider (LHC) at CERN (CMS Collaboration, 2000).

<table>
<thead>
<tr>
<th>Mode</th>
<th>In[0]</th>
<th>Out[0]</th>
<th>Out[1]</th>
<th>Actor</th>
<th># Inputs</th>
<th># Outputs</th>
<th>Cons./Prod Rates</th>
<th>Model Detected</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>-64</td>
<td>1</td>
<td>1</td>
<td>Jet Reconstruction</td>
<td>1</td>
<td>2</td>
<td>64/1</td>
<td>SDF</td>
</tr>
<tr>
<td>1</td>
<td>-64</td>
<td>1</td>
<td>1</td>
<td>Cluster Threshold</td>
<td>12</td>
<td>12</td>
<td>1/1</td>
<td>HSDF</td>
</tr>
<tr>
<td>1</td>
<td>-64</td>
<td>1</td>
<td>1</td>
<td>Cluster Compute</td>
<td>12</td>
<td>6</td>
<td>1/1</td>
<td>HSDF</td>
</tr>
<tr>
<td>1</td>
<td>-64</td>
<td>1</td>
<td>1</td>
<td>Cluster Isolation</td>
<td>1</td>
<td>2</td>
<td>64/8</td>
<td>SDF</td>
</tr>
<tr>
<td>SDF</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>HSDF</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Iterative Module Partitioning

- Detect actor’s dataflow model:
  - If found, then done
  - Else, partition the original actor into sub-functions
  - Continue until a model has been found for each actor or no further partitioning is possible
Evaluation Parameters

- Model detection algorithm was applied to 20+ actors from the LIDE library of actors.
- The MSM for each actor was obtained a priori through manual inspection and was used to verify the correctness of the automatic dataflow model identification algorithm.
- The DSPCAD command line environment (DICE) was used to execute the experiments and collect results.
- The experiments were conducted on a machine with a 2.6 GHz Core2Duo processor and 2 GB of RAM.
- Each experiment was repeated 100 times.
Performance Results

Model detection algorithm was able to correctly identify 22 out 23 actors.

<table>
<thead>
<tr>
<th>Model Detected</th>
<th>Number of Actors</th>
</tr>
</thead>
<tbody>
<tr>
<td>HSDF</td>
<td>6</td>
</tr>
<tr>
<td>SDF</td>
<td>7</td>
</tr>
<tr>
<td>CSDF</td>
<td>2</td>
</tr>
<tr>
<td>BDF</td>
<td>2</td>
</tr>
<tr>
<td>EIDF</td>
<td>6</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Model Detected</th>
<th>Complexity</th>
</tr>
</thead>
<tbody>
<tr>
<td>HSDF</td>
<td>$O(n)$</td>
</tr>
<tr>
<td>SDF</td>
<td>$O(n)$</td>
</tr>
<tr>
<td>CSDF</td>
<td>$O(n \log n)$</td>
</tr>
<tr>
<td>BDF</td>
<td>$O(kn \log n)$</td>
</tr>
<tr>
<td>IDF</td>
<td>$O(ckn \log n)$</td>
</tr>
<tr>
<td>EIDF</td>
<td>$O(ckn \log n)$</td>
</tr>
</tbody>
</table>

Each point is mean of 100 runs.
ASMDF Project overview: design and implementation of Adaptive Stream Mining systems using DataFlow methods

Lightweight dataflow

Multi-objective design optimization in the lightweight dataflow for DDDAS environment (LiD4E)

Dataflow model detection

Application area: tracking networks using mobile devices (with T. Damarla, ARL, and W. Stechele, T. U. Munich)

Application area (emerging work) \(\rightarrow\) Multispectral video processing (with E. Blasch, AFRL)

Summary
Motivation

• People and vehicle tracking in wilderness → important for border security applications

• Mobile devices are attractive to use as prototypes for disposable sensor node platforms
  – Low cost
  – Disposability
  – Integration of advanced communications, sensing, and processing features
  – Capability for interfacing with more advanced external sensors
  – Flexible demonstration and design iteration before committing resources to custom sensor node implementation
Problem Description

• Data-driven tracking system integrating computational and measurement processes
  - Optimized operation on mobile devices
  - Understanding system design trade-offs under resource constraints
• Dataflow-based design of an optimized tracking application
• Multidimensional constraints
  - Tracking accuracy
  - Real-time performance
  - Energy consumption

⇒ DDDAS-enabled Tracking System for Mobile Devices (DTSMD)
  - Selects efficient tracking algorithm configurations in terms of trade-offs among accuracy, energy efficiency, and real-time performance.
  - System architecture that facilitates multi-objective design optimization
Design Flow

- Input: acoustic signal
- 3 output classes:
  - Person
  - Vehicle
  - Noise
Feature Extraction Actor

- **Cadence analysis**
  - Means (DC offset) removal and signal normalization
  - FFT computation of the signal envelope and extraction of the first $p_f$ FFT samples

- **Mutual information based feature extraction**
  - Means (DC offset) removal and signal normalization
  - FFT computation of the signal envelope and extraction of $p_f$ features using mutual information

- **Cepstral analysis**
  - Means (DC offset) removal and signal normalization
  - Computation of the cepstral coefficients and extraction of the first $p_f$ coefficients

- **DDDAS-based integration**
  - Instrumentation for dynamic SNR assessment
  - Adaptation of feature extraction mode based on SNR threshold
System-Level Dataflow Model

Input Data
\[ e_1 \]
Target Detection
\[ e_2 \]
Feature Extraction
\[ e_3 \]
Classification
\[ e_4 \]
Results
\[ e_6 \]
Output

Training parameters
\[ e_5 \]

Binary Classifier
\[ e_7, e_8, e_9 \]
Voting
Execution Time Comparison of SVM Classifiers Employed

- Each classifier implementation was executed 100 times on the tablet
Adaptive Tracking Solution

- System adapts among different classification and feature extraction algorithms depending on existing operational conditions.
- 2 constraints considered
  - Remaining battery capacity
  - SNR level of the detected signal
- Energy-saving modes
  - Executed when the battery level is low
- 3 parameters define each operating mode:
  - $p_d$: Target interval
  - $p_{fem}$: Feature extraction mode
  - $p_{cm}$: Classifier mode
- $T_s$ = threshold value of the SNR level
- $T_{b1}$ and $T_{b2}$ = thresholds of the battery level
  - Gradual shut-down
## Adaptive Tracking Solution

<table>
<thead>
<tr>
<th>States</th>
<th>Target interval length, $p_d$ (sec)</th>
<th>Feature extraction mode, $p_{fem}$</th>
<th>Classifier mode, $p_{cm}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$S_1$</td>
<td>6</td>
<td>Cepstral analysis</td>
<td>SVM – rbf</td>
</tr>
<tr>
<td>$S_2$</td>
<td>4</td>
<td>Cadence analysis</td>
<td>SVM – linear</td>
</tr>
<tr>
<td>$S_3$</td>
<td>4</td>
<td>Cadence analysis</td>
<td>LDA</td>
</tr>
<tr>
<td>$S_4$</td>
<td>3</td>
<td>Mutual information-based feature extraction</td>
<td>SVM – rbf</td>
</tr>
</tbody>
</table>

Decision actor determines the values of $p_d$, $p_{fem}$, and $p_{cm}$, and thus, the modes in which the classification and feature extraction actors will be executed.
Evaluation of Adaptation Approach

- 3 solutions considered:
  - Solution 1: The system is configured statically under the settings of state $S_1$ (MFCC - SVM rbf - 6 sec)
  - Solution 2: The system is configured statically under the settings of state $S_4$ (Mutual information - SVM rbf - 3 sec)
  - Adaptive solution: The system is configured dynamically using the adaptive approach, without considering the energy saving modes.

<table>
<thead>
<tr>
<th>Solutions</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Solution 1</td>
<td>84.21 %</td>
</tr>
<tr>
<td>Solution 2</td>
<td>81.82 %</td>
</tr>
<tr>
<td>Adaptive solution</td>
<td>91.39 %</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>States</th>
<th>Voltage level (V)</th>
<th>Discharge (mAh)</th>
<th>Consumed energy per execution (J)</th>
</tr>
</thead>
<tbody>
<tr>
<td>$S_1$</td>
<td>3.571</td>
<td>0.3604</td>
<td>4.68</td>
</tr>
<tr>
<td>$S_2$</td>
<td>3.658</td>
<td>0.2883</td>
<td>3.96</td>
</tr>
<tr>
<td>$S_3$</td>
<td>3.607</td>
<td>0.2703</td>
<td>3.51</td>
</tr>
<tr>
<td>$S_4$</td>
<td>3.632</td>
<td>0.2163</td>
<td>2.82</td>
</tr>
</tbody>
</table>
Adaptive Tracking on Mobile Platforms: Summary and Ongoing Work

- Design and implementation of an adaptive system for detecting and tracking human footsteps and vehicles from mobile devices.
- System adapts among different classification and feature extraction algorithms depending on current operational conditions.
- Experiments on an Android-based implementation.
- Analysis of the experimental results in terms of tracking accuracy and energy efficiency.

Ongoing work:
- Interfacing with high quality external sensors
- Investigation of networked mobile sensor nodes, including distribution of tracking system processing across the network
- Extension of the adaptive, mobile-device-based tracking system to apply multiple sensing modalities (e.g., seismic sensor data in conjunction with acoustic data).
Talk Outline

• ASMDF Project overview: design and implementation of Adaptive Stream Mining systems using DataFlow methods
• Lightweight dataflow
• Multi-objective design optimization in the lightweight dataflow for DDDAS environment (LiD4E)
• Dataflow model detection
• Application area: tracking networks using mobile devices (with T. Damarla, ARL, and W. Stechele, T. U. Munich)
  • Application area (emerging work) → Multispectral video processing (with E. Blasch, AFRL)
• Summary
Background

• With the advances in video acquisition technology, multispectral video processing is attracting increasing interest.

• Multispectral video offers better spectral resolution compared to monochromatic video.

→ New opportunities and challenges for applying the paradigm of DDDAS to design and implementation of video analytics systems;

→ subset of available multispectral bands to store/communicate/process as a key system design parameter.
First Version Testbed

• Novel data set from U. de Bourgogne (Benezeth et al.) that provides the first publicly available collection of annotated multispectral video sequences

• Target application: background subtraction

• GMM applied to individual bands for feature-level fusion

• Lightweight dataflow employed for system level design and prototyping on PC and Android platforms

• OpenCV applied for specialized image processing functions
  – large third-party library of software components for computer vision
Summary

- This project addresses the need for structured design methodologies, graphical models, and software tools for dynamic, data-driven, adaptive stream mining (ASM) systems.
- We have further developed and applied our recently-developed tool: Lightweight Dataflow for Dynamic, Data-Driven Application Systems Environment (LiD4E).
- We have introduced new system design methodologies in the ASM multi-objective design optimization (AMDO) framework.
- We have introduced model detection methods to automate the derivation of most specialized models for actors in LiD4E.
- We have developed a mobile (Android-based) testbed for experimentation with embedded stream mining systems.
- We have developed a novel system for adaptively tracking people and vehicles on this testbed using LiD4E (with T. Damarla, ARL).
- We are exploring the application of our methods and tools to multispectral video processing (with E. Blasch, AFRL).