Building compositional learning and optimization applications for mobile sensor networks with PILOT

Ilge Akkaya, Shuhei Emoto, Edward A. Lee

Eleventh Biennial Ptolemy Miniconference
University of California, Berkeley

October 16, 2015
Table of Contents

1 Introduction

2 PILOT Workflow
   - PILOT Actor Library
   - State-space Modeling

3 Case Studies
   - Designing Cooperative Robotic Applications using PILOT
   - Real-time Streaming Learning Applications

4 Conclusion
Motivation

Distributed system design is prone to errors:

- time and concurrency often not addressed by programming abstractions
- algorithm design is not compositional

In a mobile sensor network setting, design requirements are even more complex

Existing software abstractions based on imperative code fall short on providing

- Scalability
- Structured, repeatable code for deterministic behavior
- Flexible interfaces for variable computational resources
Introduction

- PILOT (Ptolemy Inference, Learning, and Optimization Toolkit) is an actor library for structured design of robotic sensor network applications
- **Goal:** Designing sensor-to-actuator *streaming* learning and control applications
- Reusable, **actor-oriented component abstractions** that are less error-prone and can be deployed on variable network resources
- Use **aspects** to enable separation-of-concerns
Table of Contents

1. Introduction

2. PILOT Workflow
   - PILOT Actor Library
   - State-space Modeling

3. Case Studies
   - Designing Cooperative Robotic Applications using PILOT
   - Real-time Streaming Learning Applications

4. Conclusion
PILOT Library

- Hidden Markov Model parameter estimation and decoding
- Particle Filtering
- Kalman Filtering [Emoto]
- Aspect-oriented state-space modeling, dynamics simulation, prediction
- Sensing and dynamics models
Focus: Accessibility via Specialized Components

PILOT Workflow
- ParticleFilter
- HMMGaussianEstimator
- ParticleFilterRange
- HMMExponentialEstimator
- CollaborativeRangeParticleFilter
- HMMMultinomialEstimator

Introduction
- PILOT Library
- State-space Modeling

Case Studies
- Cooperative Apps
- Streaming ML

Conclusion
State-Space Modeling in PILOT

A general Bayesian state-space model is given by

\[ x_0 \sim \pi_X(x_0) \quad \text{(prior)} \]
\[ z_t | x_t \sim g(x_t, u_t, t) \quad \text{(measurement model)} \]
\[ x_{t+1} | x_t \sim f(x_t, u_t, t) \quad \text{(state dynamics)} \]

Example:

\[
X_t = \begin{bmatrix} x_t; y_t \end{bmatrix}
\]
\[ x_0, y_0 \sim \text{Uniform}(-100, 100) \]
\[ X_{t+1} = X_t + \eta_t \]
\[ z_t = \sqrt{x_t^2 + y_t^2} + \omega_t \]
\[ \omega_t \sim \mathcal{N}(0, \sigma^2) \]
\[ \eta_t \sim \mathcal{N}(\begin{bmatrix} 0 \\ 0 \end{bmatrix}, \Sigma) \]
System Architecture

Figure: State-Space Aware System Architecture in PILOT
Table of Contents

1 Introduction

2 PILOT Workflow
   - PILOT Actor Library
   - State-space Modeling

3 Case Studies
   - Designing Cooperative Robotic Applications using PILOT
   - Real-time Streaming Learning Applications

4 Conclusion
Case Study: Cooperative Robotic Control

- A network of mobile sensor nodes
- Range-only sensors used to sense the position of mobile target(s)
- A cloud based application (centralized or decentralized) which takes range measurements and computes future robot trajectories to achieve a control goal
  - Localize target as fast as possible
  - Pursue the target
  - A multi-objective control goal
- Subject to environmental constraints
  - Obstacle/Collision avoidance
  - Speed and acceleration constraints
Case Study: Cooperative Robotic Control

Figure: Top-Level Model for Range-Only Target Localization
PILOT Control Workflow

Introduction
PILOT Workflow
PILOT Library
State-space Modeling
Case Studies
Cooperative Apps
Streaming ML
Conclusion

UC Berkeley
Case Studies
Ilge Akkaya

PILOT
Ilge Akkaya, Shuhei Emoto, Edward A. Lee
Eleventh Biennial Ptolemy Miniconference
University of California, Berkeley

RangeSensor2RangeSensor1
RangeSensor4RangeSensor3
RangeSensor2RangeSensor1
RangeSensor4RangeSensor3

Figure: PILOT Model for Target State Estimation and Trajectory Optimization
State-Space Oriented Learning Models

Figure: Sample particle output for state estimation using range-only sensing
\[ x^* = \min_{x \in \mathbb{R}^n} f(x, q_1, q_2) \]
subject to \[ g(x, q_1, q_2) \geq 0, \]

**Algorithm**

**CompositeOptimizer**

**Input:** \( Q \leftarrow Q_i \)

**Output:** \( x^* \) that is a local optimum of \( f(\cdot) \)

define \( P \): An actor that implements SDF, has inputs: \( x, Q \) and outputs: \( f, g \)

while \( k < k_{MAX} \) \& !\( \text{CompositeOptimizer.converged()} \)
do
\( x^{(k)} \leftarrow \text{OptimizerDirector.getNextX}() \);
P.readInputs(\( x \leftarrow x^{(k)}, Q \leftarrow Q_i \));
P.execute();
P.writeOutput(\( f(x^{(k)}), Q_i \Rightarrow f^{(k)}, g(x^{(k)}, Q_i) \Rightarrow g^{(k)} \));
OptimizerDirector.computeNextX(\( f^{(k)}, g^{(k)} \));
end while

\( x^* \leftarrow \text{CompositeOptimizer.getOptimalX()} \)
Experiments: MI Maximization

Figure: Sample trajectory for the MI Maximization Control Policy

\[
\mathbf{u}_t^* = \arg \max_{\mathbf{u}_t \in \mathcal{U}^M} I(\mathbf{z}_{t+1}; x_{t+1}) \\
\text{s.t.} \|u_t^{(i)}\| \leq V_{\text{max}}, \ i = 1, 2, \ldots, M
\]
Experiments: MI Maximization + Pursuit

\[
\mathbf{u}_t^{(i)} = \begin{cases} 
\arg \min_{\mathbf{u}_t^{(i)} \in \mathcal{U}} \| \mathbf{R}_{t+1} - x_{t+1} \| \\
\text{s.t.} \| \mathbf{u}_t^{(i)} \| \leq V_{\max}, \ i = 1, 2, \ldots, M 
\end{cases}
\]

\[
\text{if } d_t^{(i)} < d_t^{(j)}, \ \forall j \neq i
\]

\[
\arg \max_{\mathbf{u}_t^{(i)} \in \mathcal{U}} I(z_t^{(i)}; x_{t+1}) \\
\text{s.t.} \| \mathbf{u}_t^{(i)} \| \leq V_{\max}, \ i = 1, 2, \ldots, M
\]

otherwise

\[
d_t^{(i)} := \| \mathbf{R}_{i,t} - x_t \|, \ i \in \{1, 2, \ldots, M\}
\]

Figure: Sample trajectory for MI Maximization with Single Pursuer
Composing Robotics Applications using PILOT Aspects

PILOT Workflow

PILOT Library

State-space Modeling

Case Studies

Cooperative Apps

Streaming ML

Conclusion

UC Berkeley

Case Studies

Ilge Akkaya
Most inference and optimization application require constraint definitions

- In the IoT context, constraints are often dynamic
- Map constraints can be added to dynamics simulations and state estimation actors such as the ParticleFilter
Real-time Feature Extraction and Streaming GMTK Classification

GMTK: Graphical Models Toolkit [Bilmes et al]
- WebSocket based streaming communication
- Audio Sampling Rate: 48 kHz
- Feature Frame Rate: 15 fps (Window Duration= 133 ms)
- 18-Channel Gammatone Filterbank (Overlap-add FFT)
- Average Feature Extraction Delay: 60 ms
Enabling Real-time Streaming Learning Applications: A Case Study on Applause Detection

Gammatone Filterbank
- 25 Hz - 10 kHz
- 18 Channels

Raw Audio Capture

Gammatone Filtering

Channel Selection and Envelope Detection

Modulation Band FFT Feature Selection

GMTK Training

\begin{align*}
p(x_t | h_t) &= 
\begin{cases}
0 & \text{No Applause} \\
1 & \text{Applause}
\end{cases}
\end{align*}

Sensor Readings

Sensor Feature Vectors

Hidden Variables

Target Variables

Feature \( i \) and \( j \):
- \(-4 \rightarrow -3 \rightarrow -2 \rightarrow -1 \rightarrow 0 \rightarrow 1 \rightarrow 2 \rightarrow 3 \rightarrow 4\)
# Table of Contents

1. **Introduction**
2. **PILOT Workflow**
   - PILOT Actor Library
   - State-space Modeling
3. **Case Studies**
   - Designing Cooperative Robotic Applications using PILOT
   - Real-time Streaming Learning Applications
4. **Conclusion**
Looking Ahead

- Introduced PILOT, an actor library that enables composing swarm applications, prototyping behaviors, and streaming models compositionally
- Enhancing connections to standard frameworks for robotics and machine learning:
  - Better integration with GMTK to provide user-friendly interfaces to graphical model training and inference
  - State-space and sensor modeling in sync with ROS
- Enhancing usability
PILOT is shipped as part of the open-source Ptolemy II Project

RSN control demos can be accessed online at:

http://ptolemy.eecs.berkeley.edu