



TerraSwarm

A Machine Learning and Optimization Toolkit for the Swarm

Ilge Akkaya, Shuhei Emoto, Edward A. Lee

University of California, Berkeley

*TerraSwarm Tools Telecon
17 November 2014*



Sponsored by the TerraSwarm Research Center, one of six centers administered by the STARnet phase of the Focus Center Research Program (FCRP) a Semiconductor Research Corporation program sponsored by MARCO and DARPA.

Report Documentation Page				Form Approved OMB No. 0704-0188	
Public reporting burden for the 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 Washington Headquarters Services, Directorate for Information Operations and Reports, 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 a penalty for failing to comply with a collection of information if it does not display a currently valid OMB control number.					
1. REPORT DATE 17 NOV 2014		2. REPORT TYPE		3. DATES COVERED 00-00-2014 to 00-00-2014	
4. TITLE AND SUBTITLE A Machine Learning and Optimization Toolkit for the Swarm				5a. CONTRACT NUMBER	
				5b. GRANT NUMBER	
				5c. PROGRAM ELEMENT NUMBER	
6. AUTHOR(S)				5d. PROJECT NUMBER	
				5e. TASK NUMBER	
				5f. WORK UNIT NUMBER	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) University of California, Berkeley, Department of Electrical Engineering and Computer Sciences, Berkeley, CA, 94720				8. PERFORMING ORGANIZATION REPORT NUMBER	
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)				10. SPONSOR/MONITOR'S ACRONYM(S)	
				11. SPONSOR/MONITOR'S REPORT NUMBER(S)	
12. DISTRIBUTION/AVAILABILITY STATEMENT Approved for public release; distribution unlimited					
13. SUPPLEMENTARY NOTES					
14. ABSTRACT					
15. SUBJECT TERMS					
16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT Same as Report (SAR)	18. NUMBER OF PAGES 40	19a. NAME OF RESPONSIBLE PERSON
a. REPORT unclassified	b. ABSTRACT unclassified	c. THIS PAGE unclassified			



Overview

1. Motivation
2. Overview of Current ML Toolkit Capabilities
3. Case Study: Cooperative Robot Localization and Control
 - State Estimation: Particle Filtering
 - Path Planning: Information Based Methods for Robot Trajectory Optimization
 - Actor-oriented Design for State Space Dynamics and Measurements
4. Future Directions & Conclusions



Motivation

ML technology in programming languages:

- MATLAB, Python, Octave, Julia, R ...

And in the form of toolkits:

- **GMTK, StreamLab, SHOGUN, Weka,...**

The state-of-the-art tools traditionally interact with data and present no native way of incorporating system aspects

Goal: to make the ML aspects a native part of the system design by

- Exploiting component-level interactions in the swarm
- Restoring the **system level roots of machine learning methodologies** by providing the **right interfaces** between machine learning tools and CPS design aspects.



Motivation

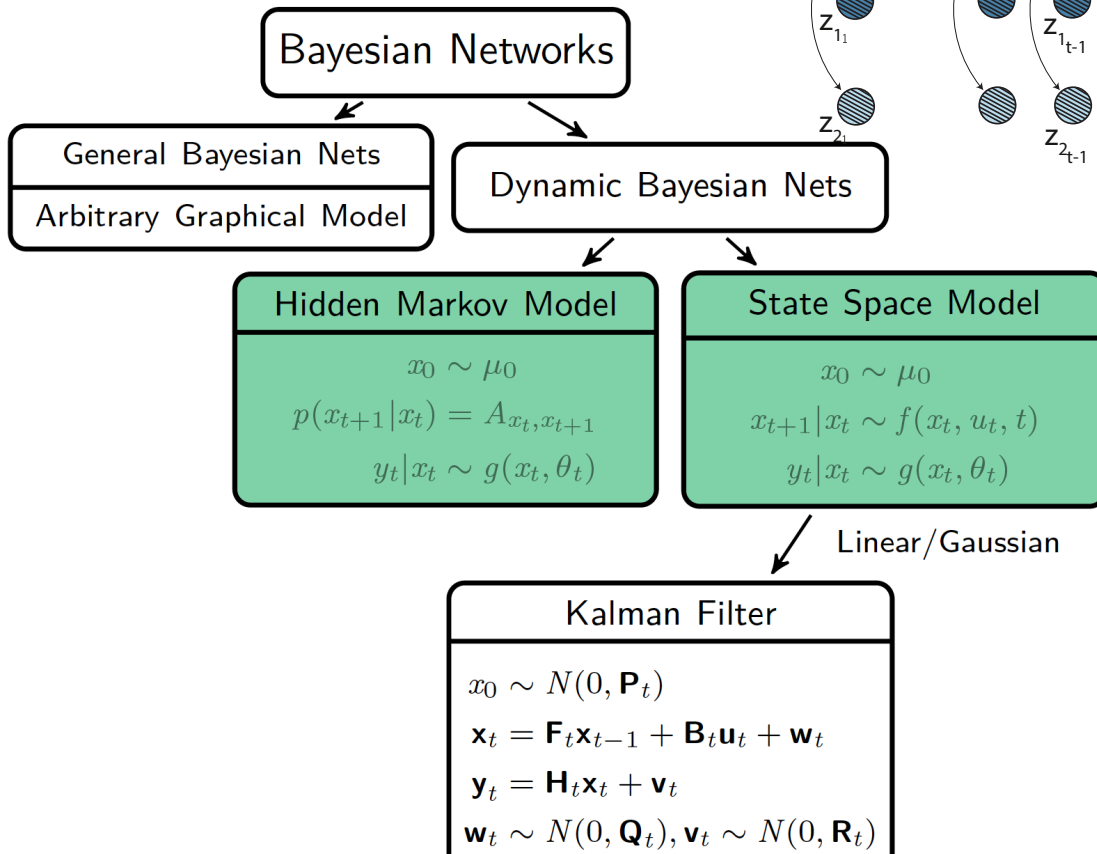
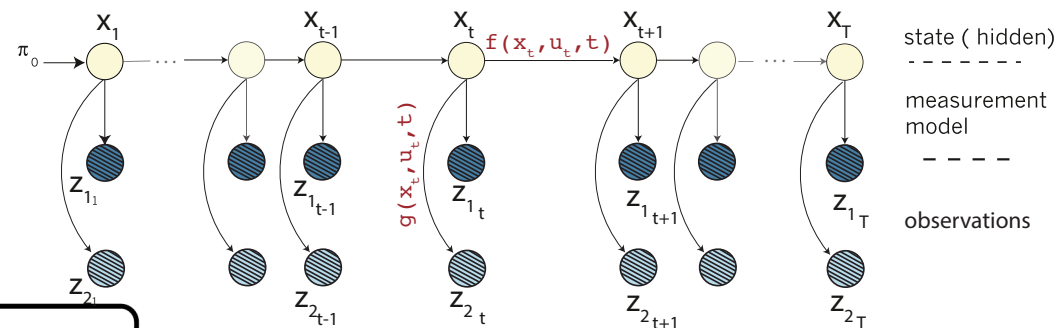
We present an actor-oriented machine learning toolkit that focuses on

- Applications of ML Algorithms to **streaming data**
- Enabling ML techniques to be natively integrated into system design
- Context-aware parameterization of a rich set of ML algorithms
- Library of **easy-to-use tools** for developers who are not ML experts
- Enhancing **programmability** of *swarmlets*



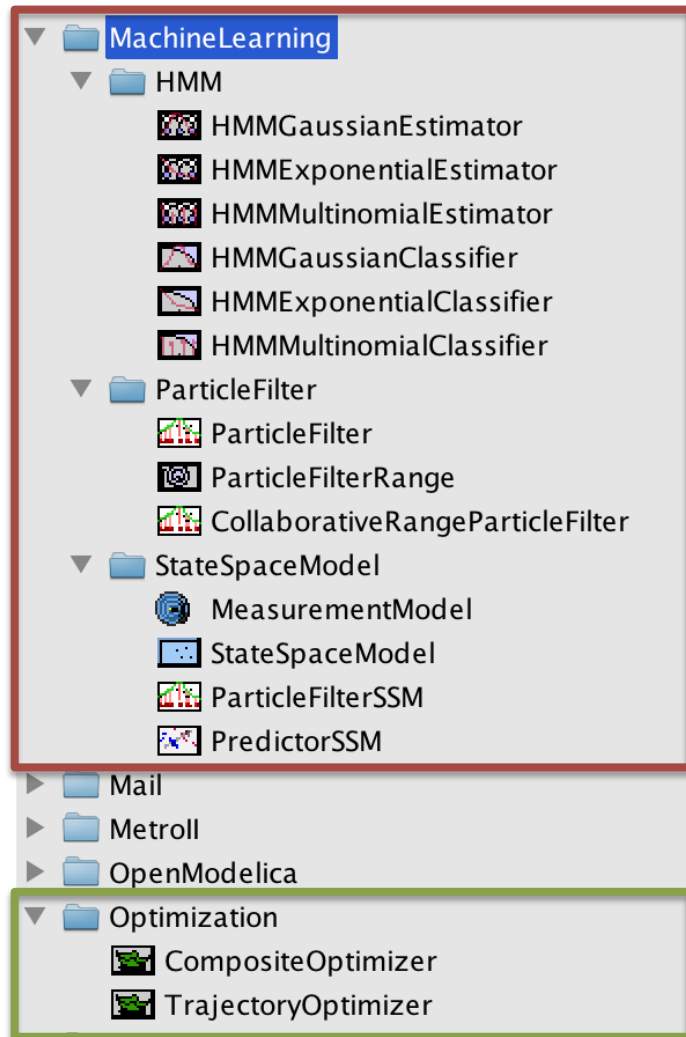
Inference for Streaming Data

Goal : Inference on data that is evolving in *time*



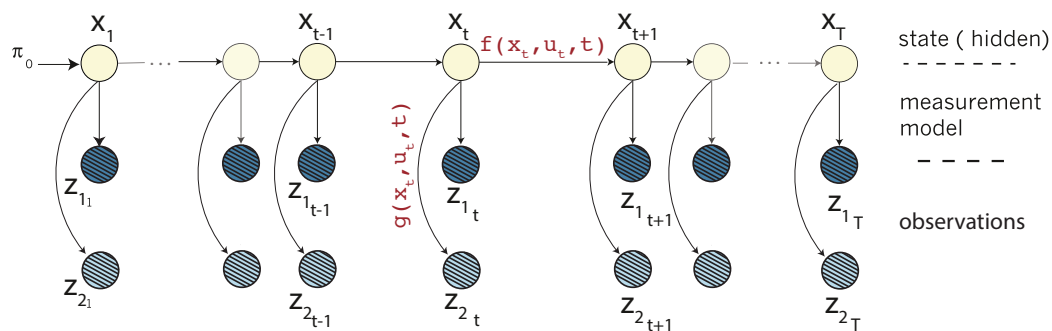


The Machine Learning Toolkit in Ptolemy II



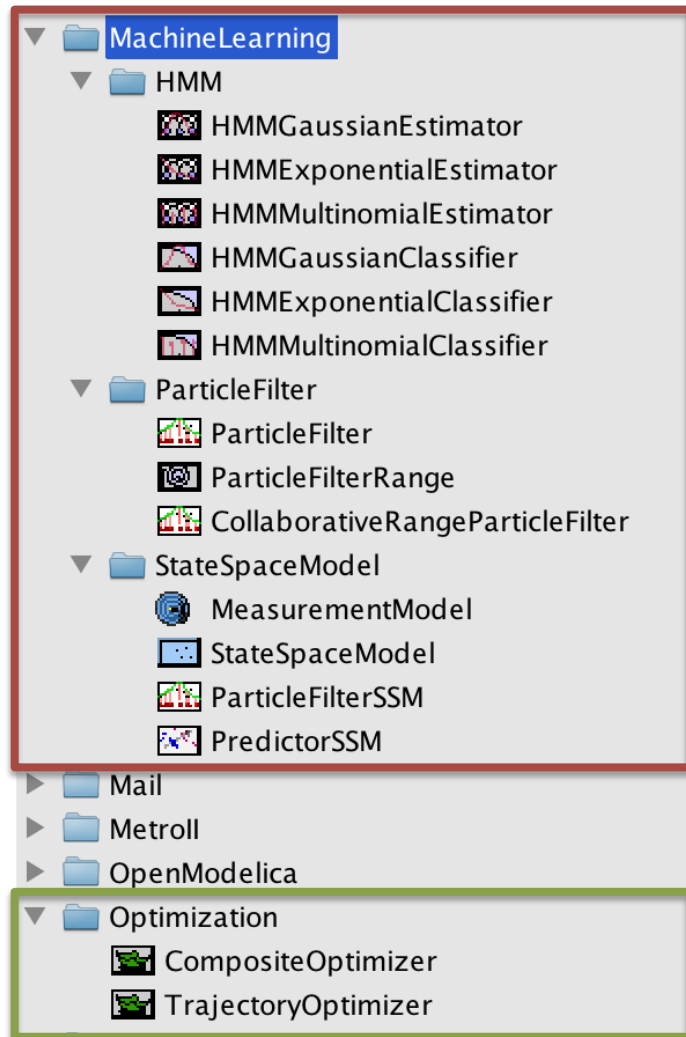
Machine Learning:

1. Hidden Markov Models (HMM)
 2. Gaussian Mixture Models (GMM)
- Parameter Estimation
 - Classification



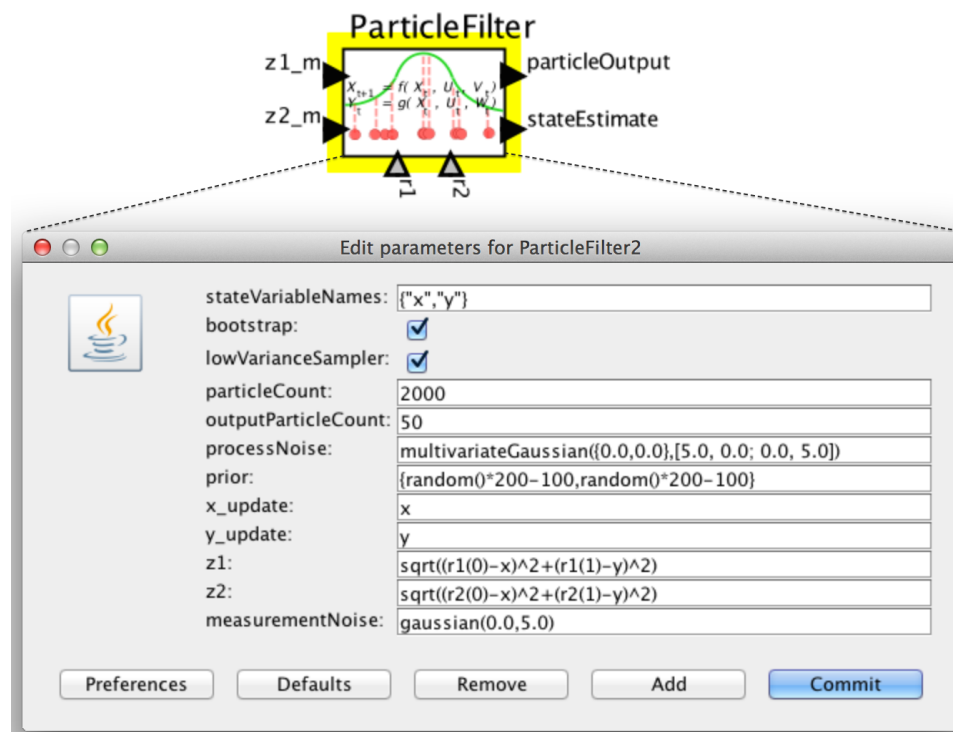


The Machine Learning Toolkit in Ptolemy II



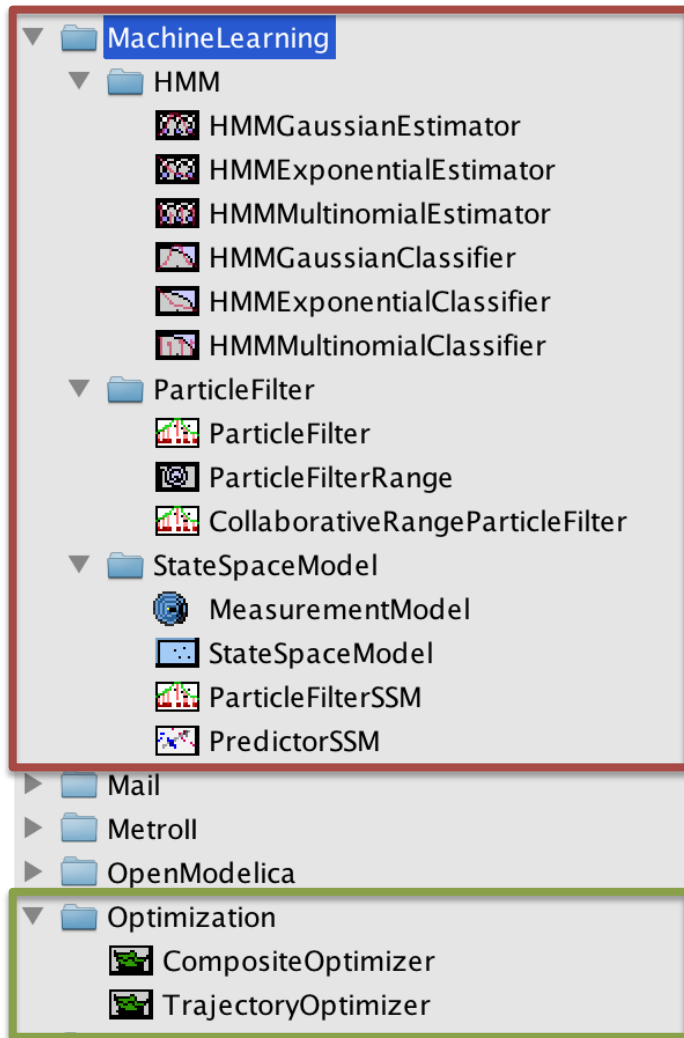
State Estimation:

- Particle Filtering



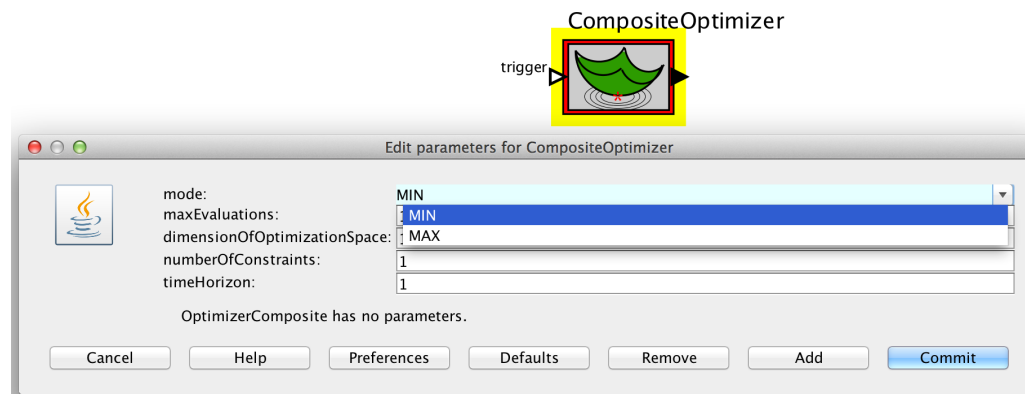


The Machine Learning Toolkit in Ptolemy II



Optimization:

- CompositeOptimizer: An actor-oriented gradient-descent solver





Application: Swarmlets for Cooperative Robot Control

Problem Definition: A team of robots, tracking/pursuing a target.

Model: *State Space Model of target dynamics*

Observations: Robot sensor measurements (generally nonlinear functions of target position + noise)

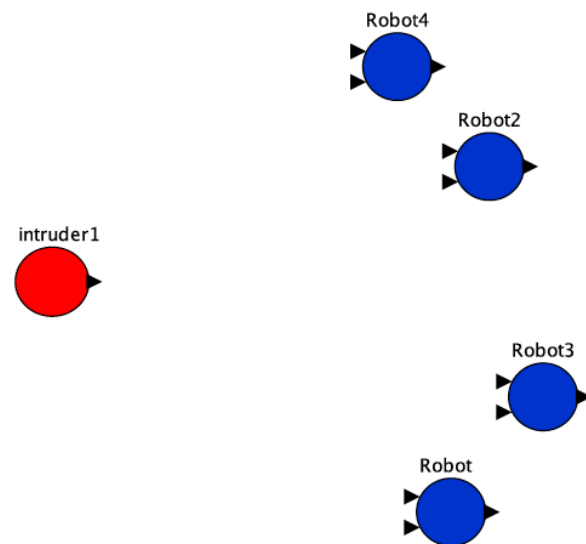
Tasks:

- *Target State Estimation*
- *Robot Path Planning: Multiple Objectives*
Collision/Obstacle Avoidance, Pursuit, SLAM, Fast Localization, Minimal Uncertainty, ...



Cooperative Robot Control : Challenges

- *Cooperation between robots*
- *Complex measurement/noise models*
 - *Range Measurements (e.g., RSSI)*
 - *Bearings Measurement (e.g., Cameras)*
- *Nonlinear robot dynamics*
- *Unknown Environment*





Cooperative Robot Localization: State Space Models

$$\theta_t = \begin{bmatrix} x_t \\ y_t \end{bmatrix} \quad \text{Target state (position)}$$

$$x_0 \sim \text{Uniform}([-100, 100])$$

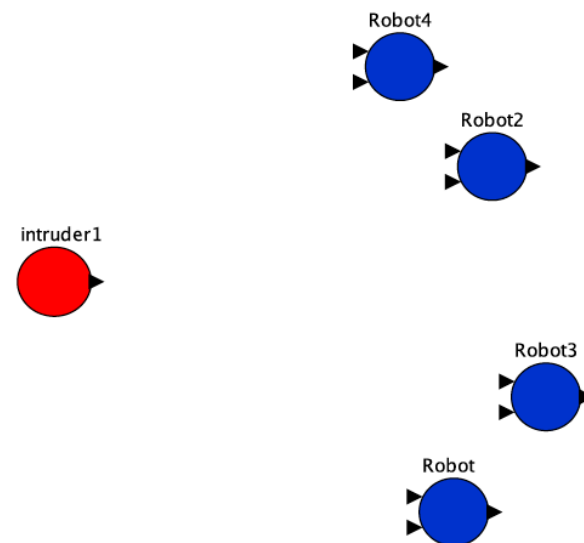
$$y_0 \sim \text{Uniform}([-100, 100])$$

$$\mathbf{z}_t = \begin{bmatrix} z_1 \\ z_2 \end{bmatrix} \quad \text{Range Measurements}$$

$$z_{it} = \|r_{it} - \theta_t\| + \omega_t, \quad i = 1, 2 \quad \text{Measurement model}$$

$$\omega_t \sim \mathcal{N}(0, \sigma^2), \sigma^2 = 5.0$$

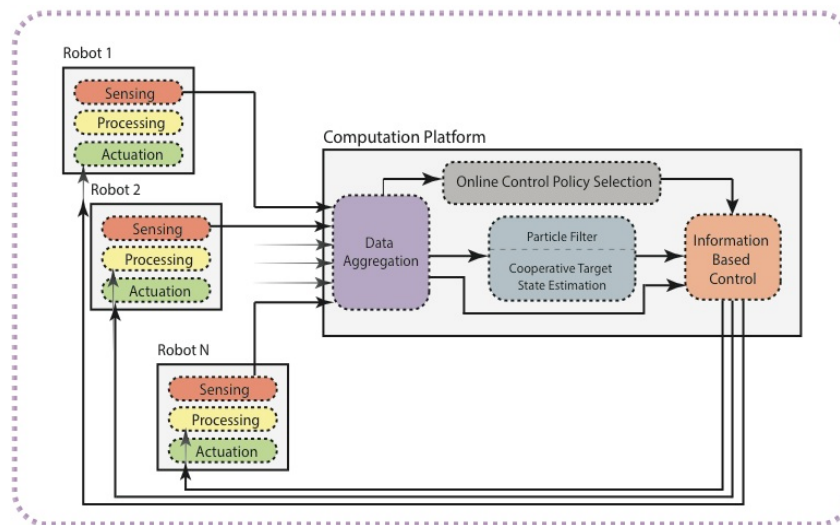
$$\theta_{t+1} = \theta_t + \nu_t, \quad \nu_t \sim \mathcal{N}\left(\begin{bmatrix} 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 5.0 & 0.0 \\ 0.0 & 5.0 \end{bmatrix}\right) \quad \text{Target state dynamics}$$





Algorithm Workflow

1. Robots make independent range measurements
2. A centralized (or local) cooperative state estimation algorithm estimates **target position** given measurements
3. Robot trajectories are **optimized** w.r.t. some objective function based on the estimated target position
4. Robots move according to the planned path



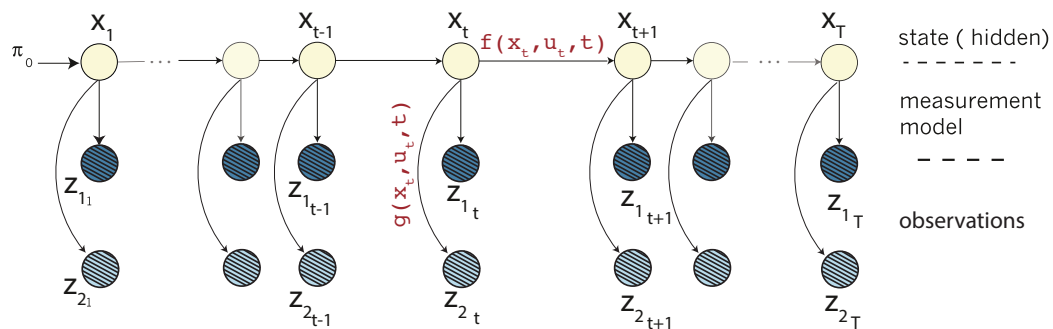


Target State Estimation

$$x_0 \sim \pi_X(x_0)$$

$$\mathbf{z}_t | x_t \sim g(x_t, u_t, t)$$

$$x_{t+1} | x_t \sim f(x_t, u_t, t)$$



- Given z_t , $t=1, \dots, T$: noisy measurements of a target state x_t ,
- Estimate $p(x_T | z_{1:T})$: Posterior density of the target state

$$\hat{p}(x_t) = p(x_t | \mathbf{z}_{1:t}) = \sum_{i=1}^N w_t^i \delta(x_t - \tilde{x}_t^i)$$

- Particle filtering is a popular Bayesian Filtering technique to solve this problem: Provides a density estimate of x_T as a particle set



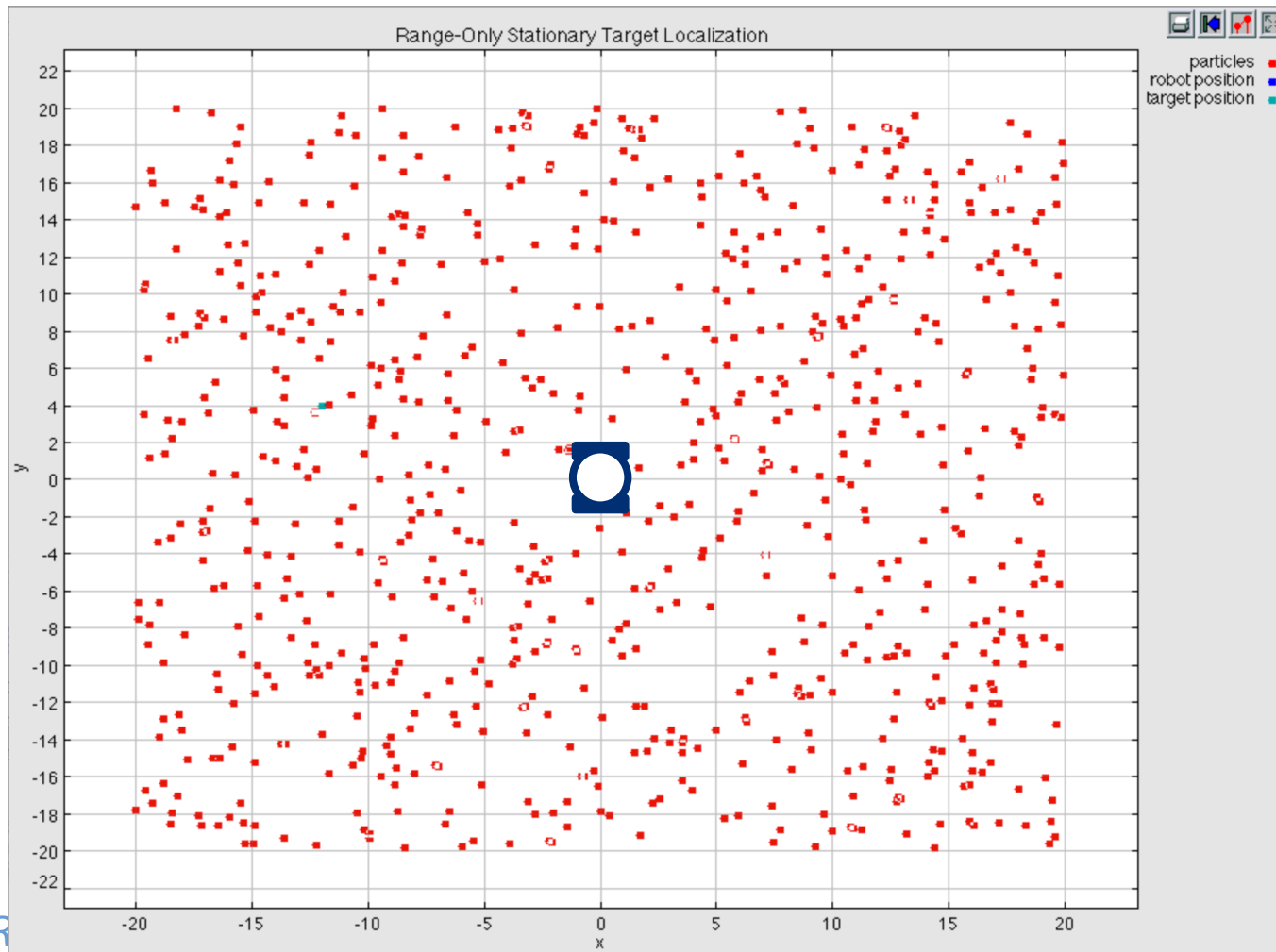
The Particle Filter

- Introducing the particle filter:
 - Sequential Monte Carlo methods as a general family
 - A Bayesian filter that performs maximum-likelihood state estimation for state-space models with
 - nonlinear dynamics and non-Gaussian noise, in the general case
 - A stochastic (and often better performing) alternative of the Kalman filter (which is only optimal for the linear Gaussian case)



Particle Filter: Operation

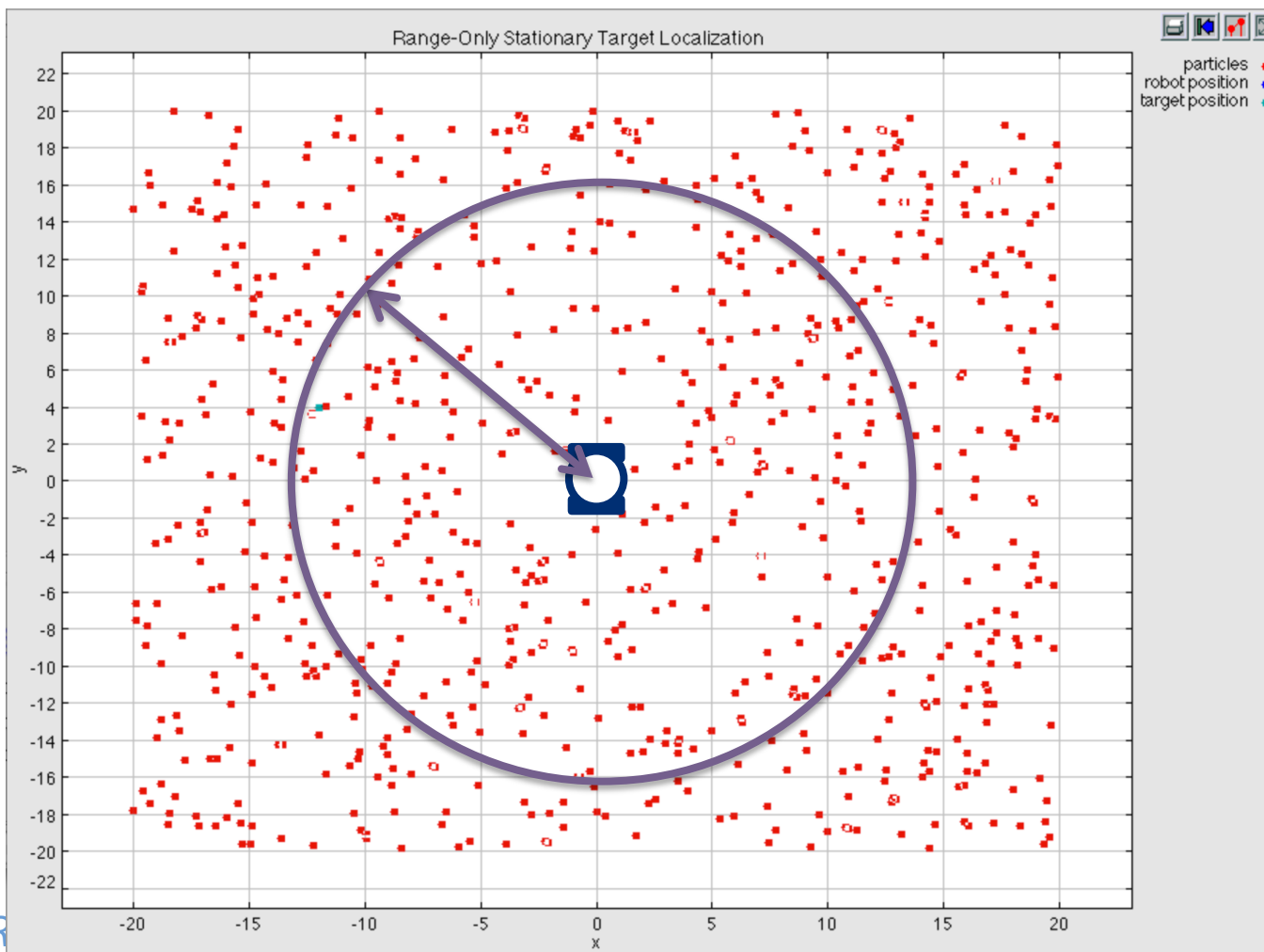
- Establish a prior belief of the state, represented as a set of particles
- Each particle is a candidate “state”, which is the intruder position in this particular application





Particle Filter: Operation

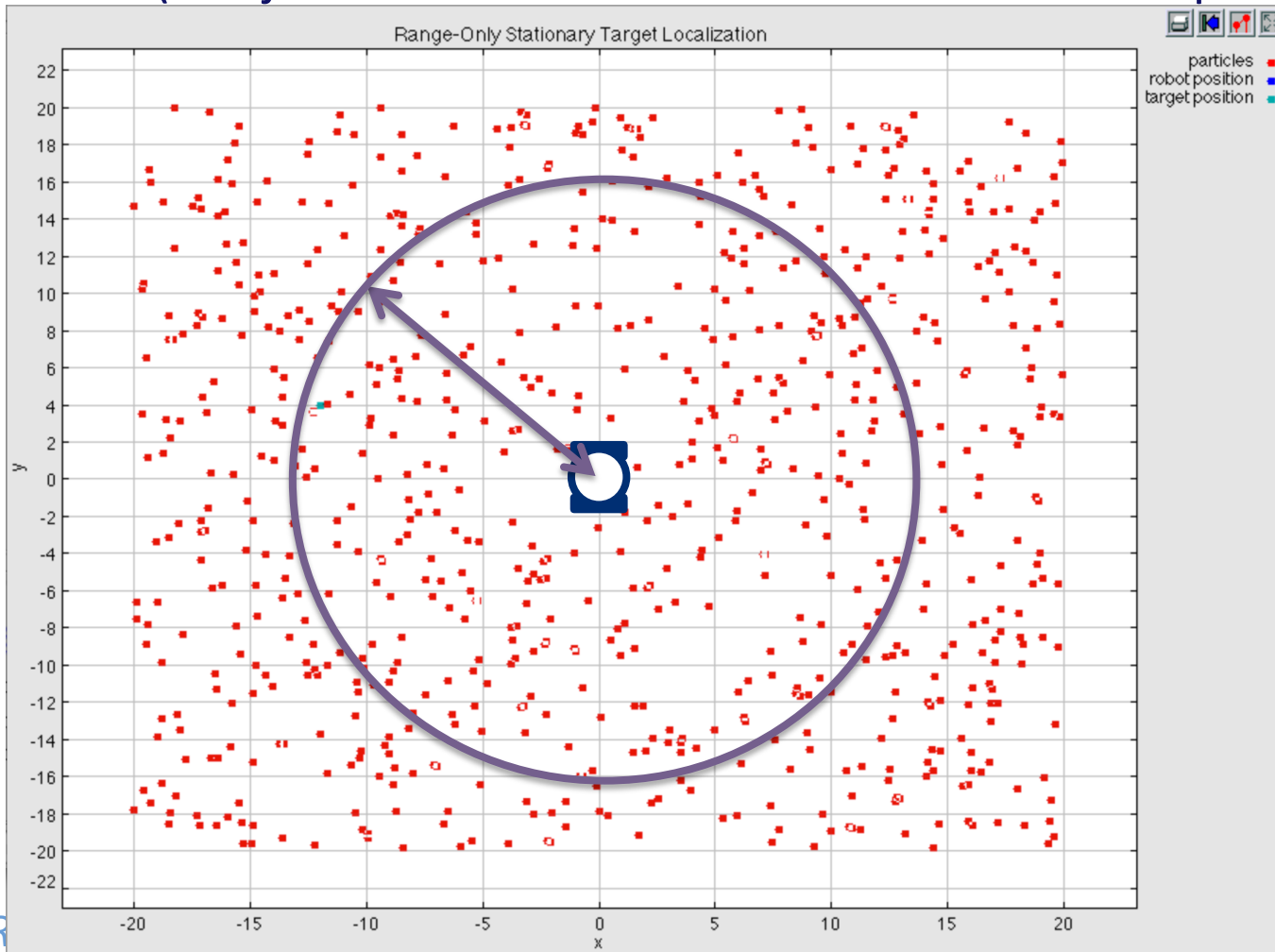
- Make a measurement





Particle Filter: Assigning Weights

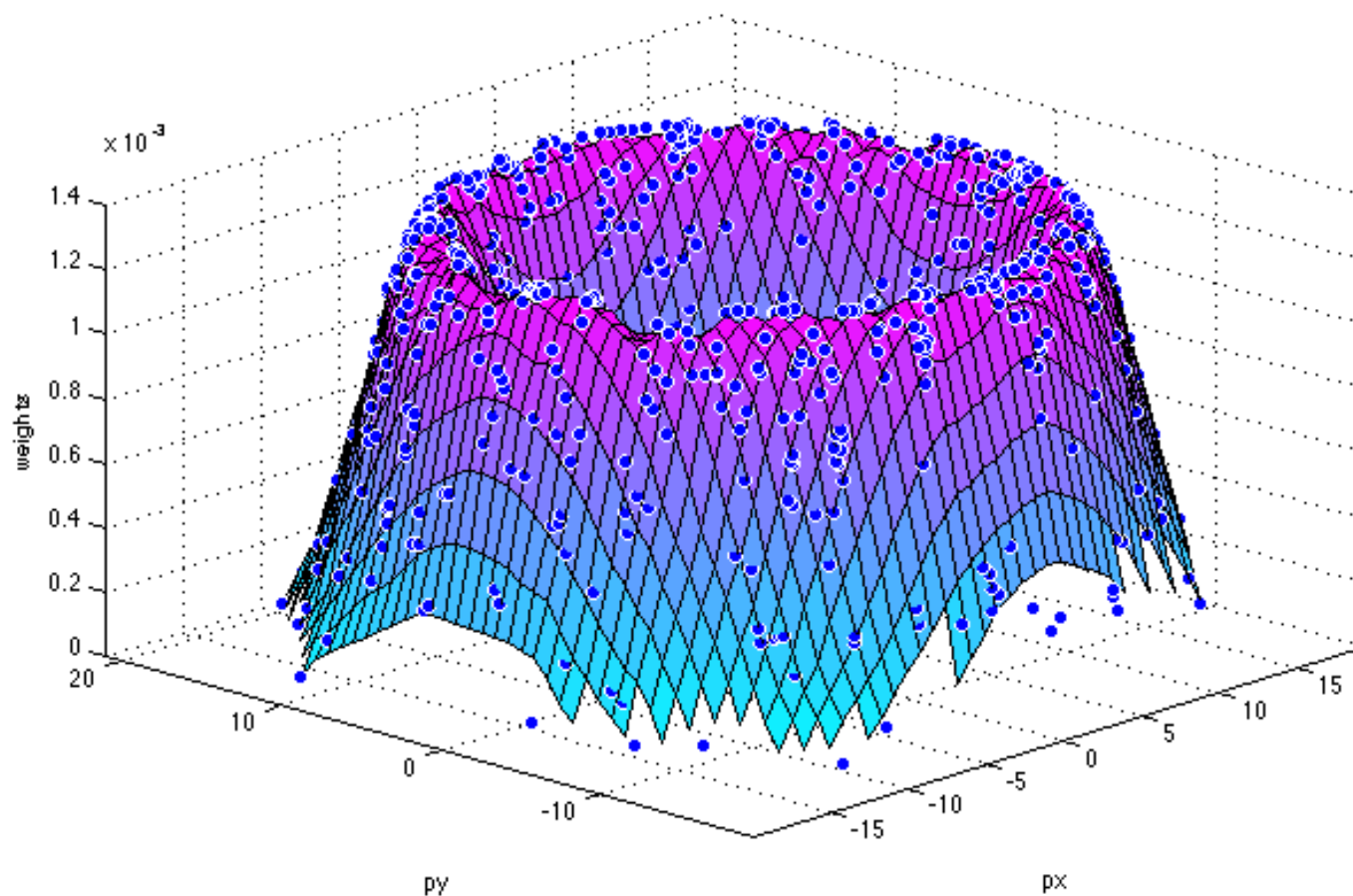
- **Assign weights** to each particle according to how well it explains the measurement (subject to a measurement model and noise specification)





Particle Filter: Assigning Weights

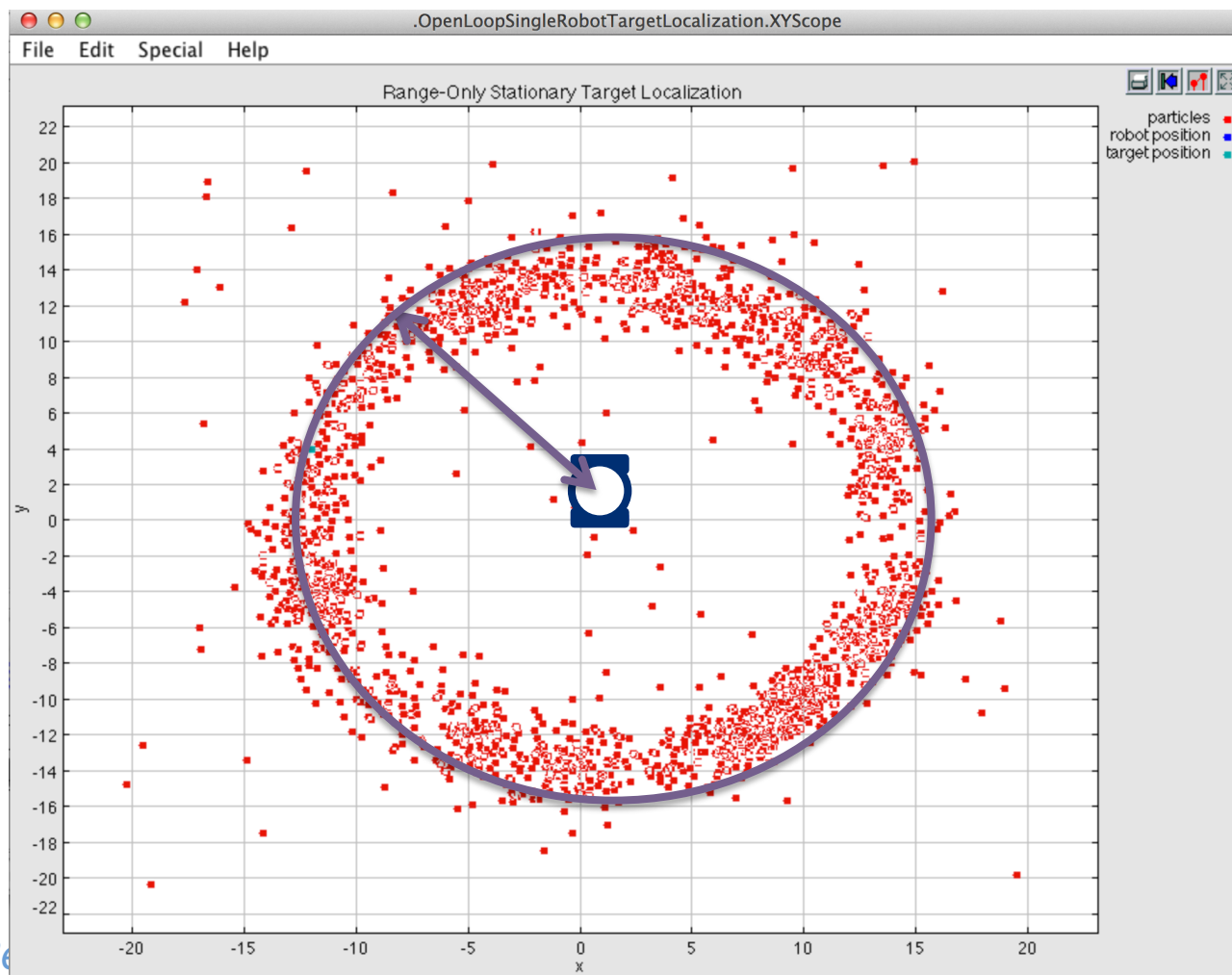
- The particle weights (under Gaussian noise) would look like the following:





Particle Filter: Resampling

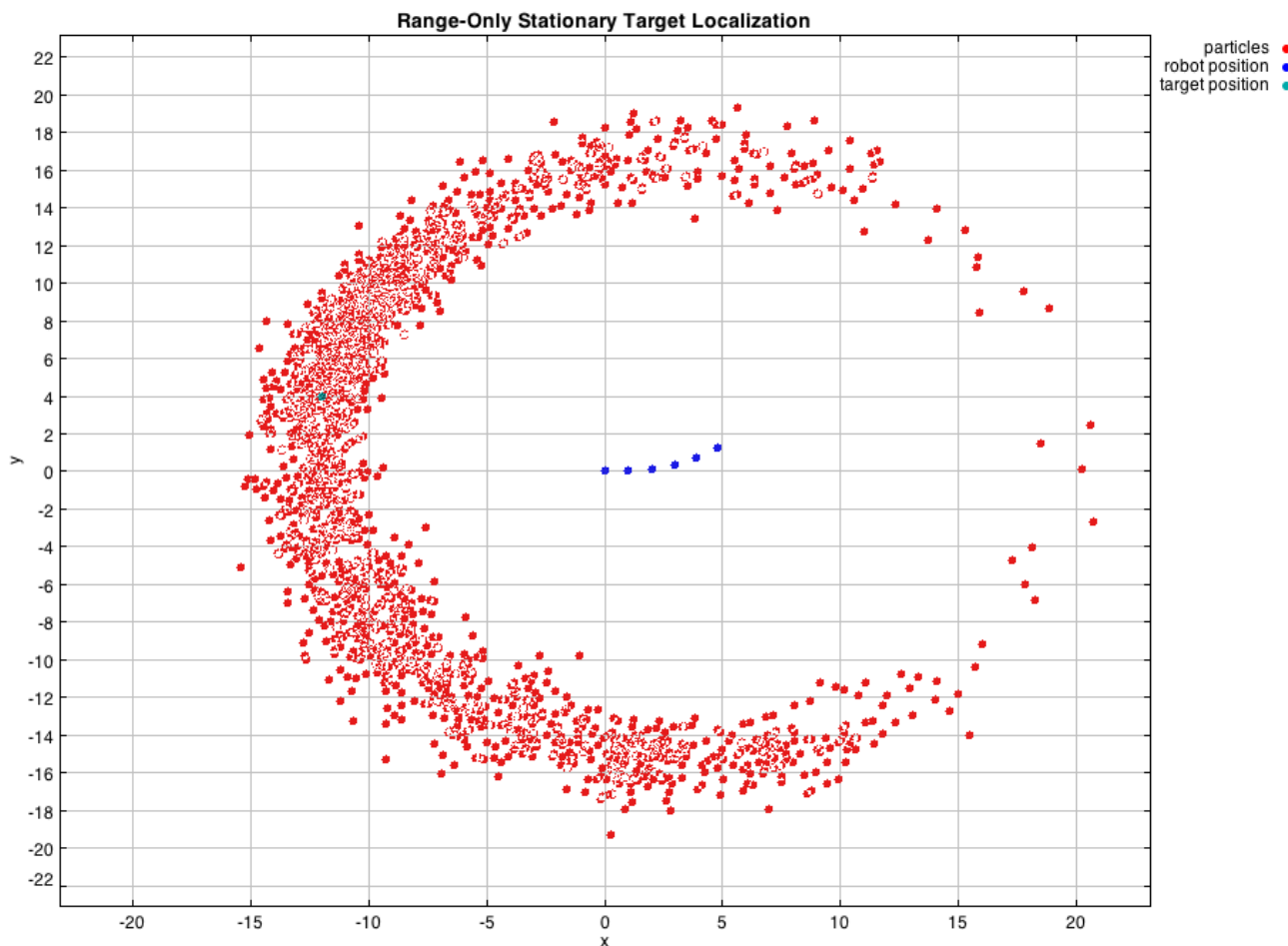
- The resulting set of particles would look like:





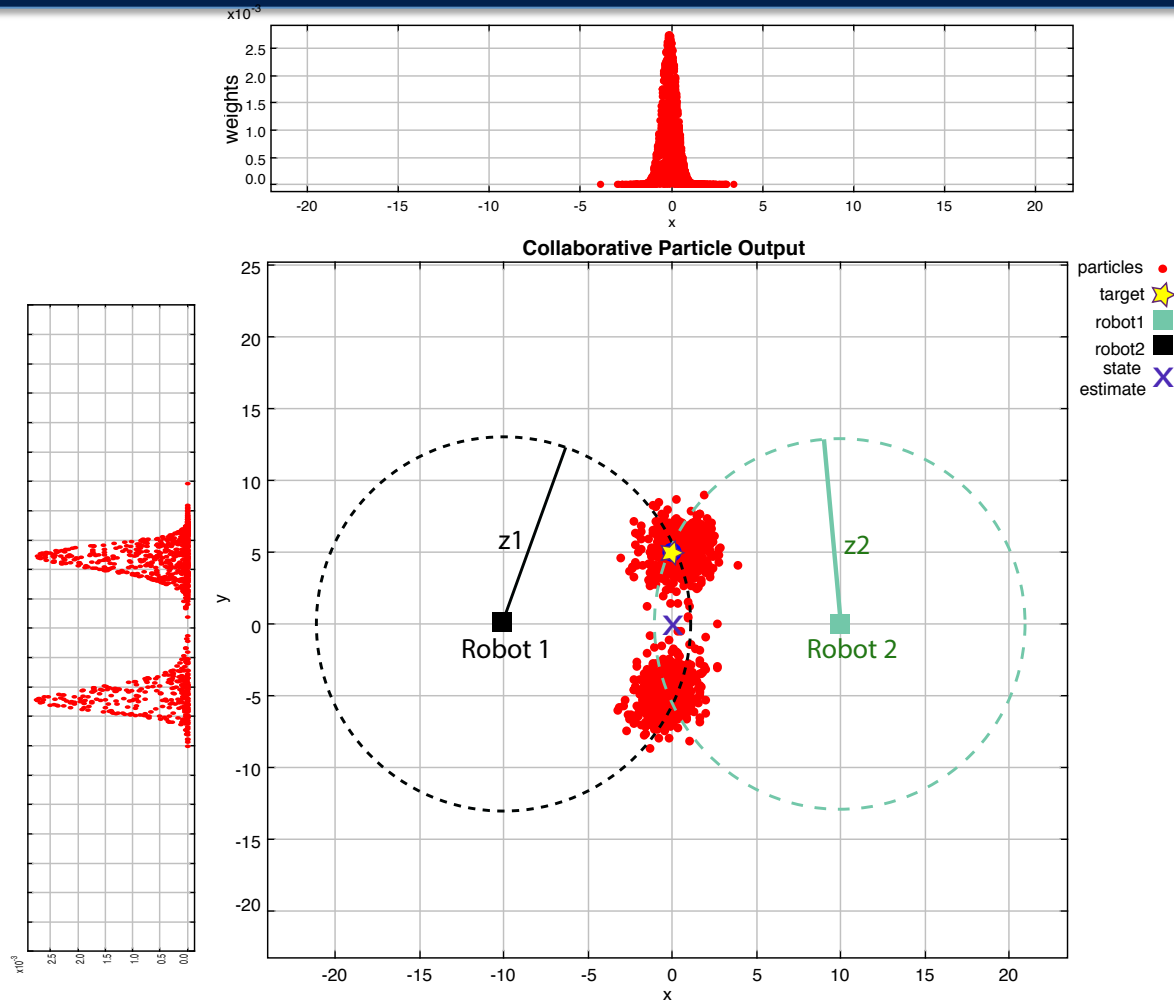
Particle Filter: Propagation

- Propagate resulting particles according to dynamics model





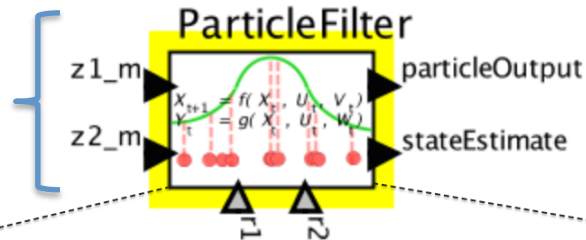
Particle Filtering with Range Sensors





Two-Observer Particle Filter

Measurement
Input



State Space
Model

Edit parameters for ParticleFilter2

stateVariableNames:	["x", "y"]
bootstrap:	☒
lowVarianceSampler:	☒
particleCount:	2000
outputParticleCount:	50
processNoise:	multivariateGaussian([0.0, 0.0], [5.0, 0.0; 0.0, 5.0])
prior:	{random()*200-100, random()*200-100}
x_update:	x
y_update:	y
z1:	sqrt((r1(0)-x)^2+(r1(1)-y)^2)
z2:	sqrt((r2(0)-x)^2+(r2(1)-y)^2)
measurementNoise:	gaussian(0.0, 5.0)

$$\theta_t = \begin{bmatrix} x_t \\ y_t \end{bmatrix}$$

$$x_0 \sim \text{Uniform}([-100, 100])$$

$$y_0 \sim \text{Uniform}([-100, 100])$$

$$\mathbf{z}_t = \begin{bmatrix} z_1 \\ z_2 \end{bmatrix}$$

$$z_{it} = \|r_{it} - \theta_t\| + \omega_t, \quad i = 1, 2$$

$$\omega_t \sim \mathcal{N}(0, \sigma^2), \quad \sigma^2 = 5.0$$

$$\theta_{t+1} = \theta_t + \nu_t, \quad \nu_t \sim \mathcal{N}\left(\begin{bmatrix} 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 5.0 & 0.0 \\ 0.0 & 5.0 \end{bmatrix}\right)$$



Path Planning

- One candidate metric to be used for online trajectory optimization: Information based methods: Mutual Information
 - A **particle** set is a good probabilistic measure of the *uncertainty* in a state variable
 - Size of particle set can be used to tune approximation bounds
- Optimization Goal: **Maximize** Mutual Information between measurements and particle set:
 - Locate intruder as precisely as possible, with fewest steps
- Can equivalently be formulated as: **Minimize** uncertainty in estimated intruder location

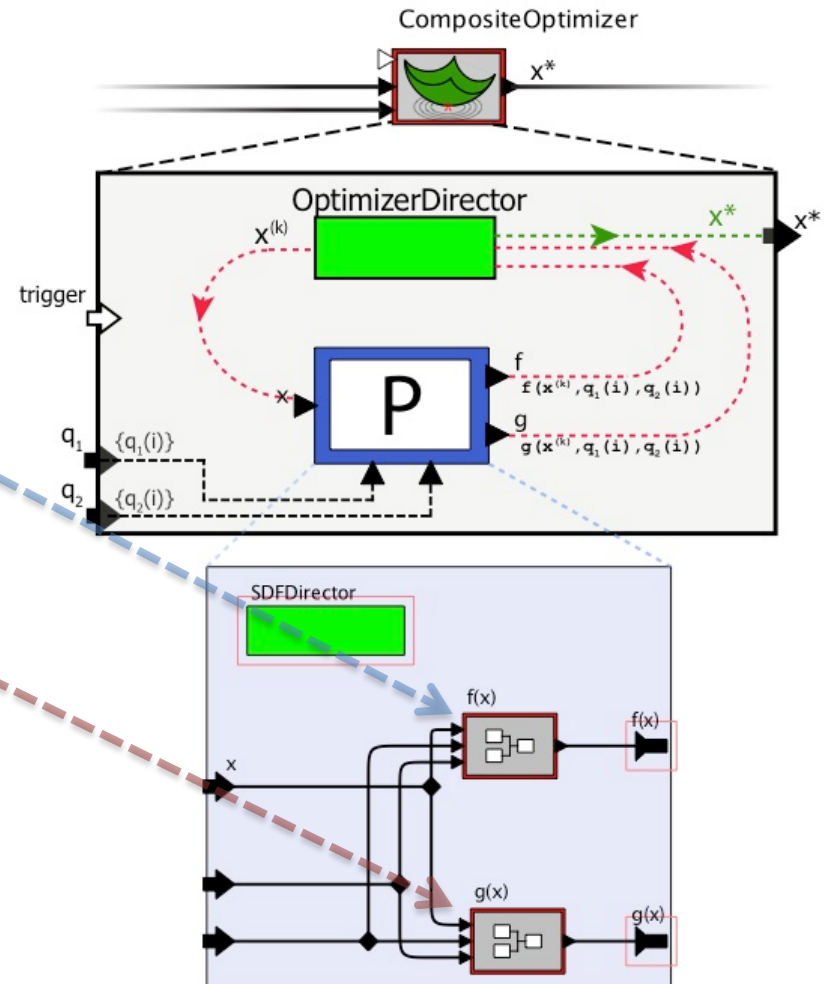


An Actor-oriented Optimizer

Consider the general constrained optimization problem of the form:

$$\begin{aligned} &\underset{\mathbf{x} \in \mathbb{R}^n}{\text{minimize}} && f(\mathbf{x}, \mathcal{Q}) \\ &\text{subject to} && g(\mathbf{x}, \mathcal{Q}) \geq 0 \end{aligned}$$

Currently supports: COBYLA,
a gradient-descent constrained
optimization solver



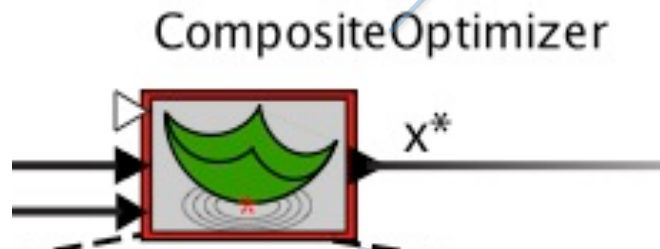


Cost Functions for Path Planning: Mutual Information

Optimization Goal: **Maximize** Mutual Information between future measurements and predicted particle set:

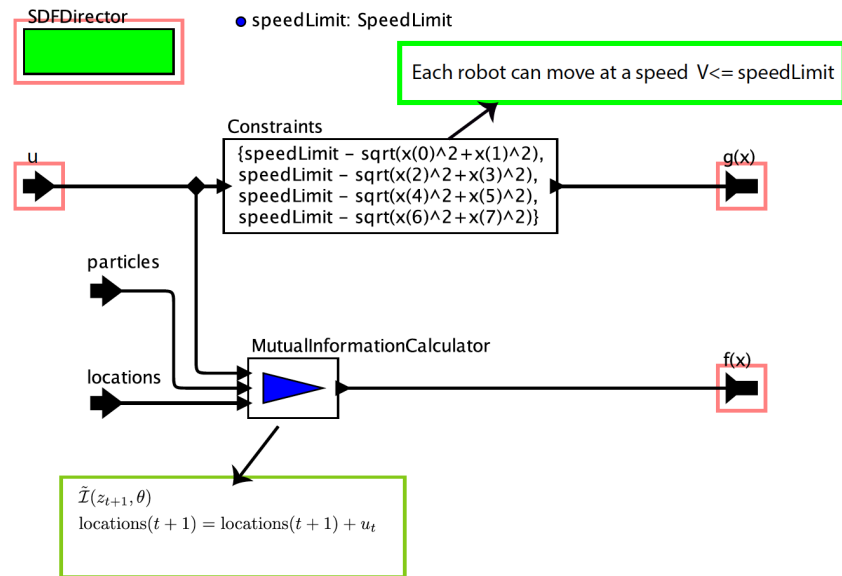
- Locate intruder as precisely as possible, with fewest steps

This can equivalently be formulated as: **Minimizing** the uncertainty in estimated intruder location. One-step optimal trajectories:



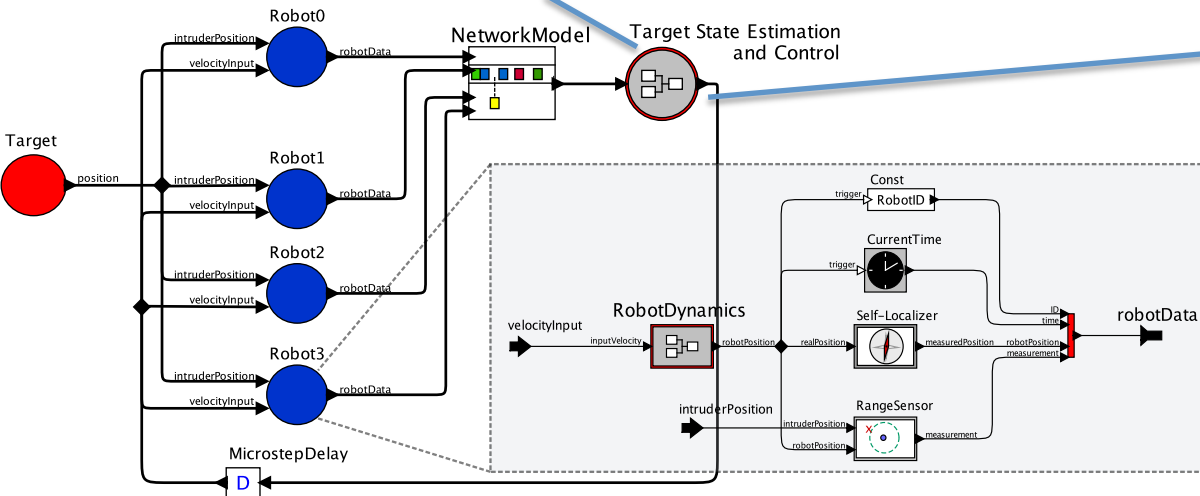
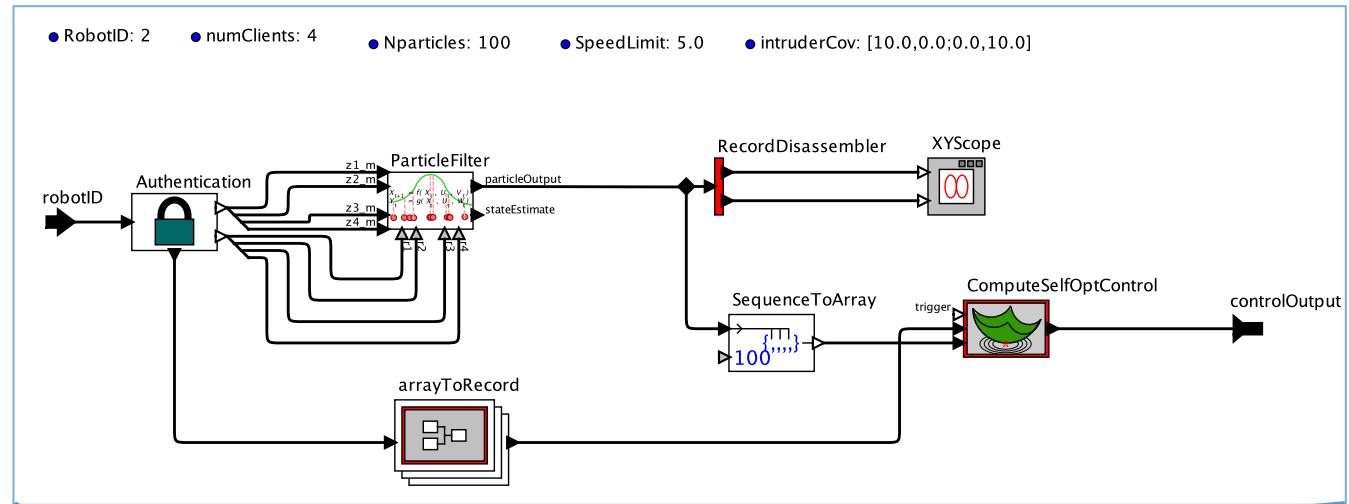
$$\mathbf{u}_t^* = \arg \max_{\mathbf{u}_t} I(z_{t+1}, x_{t+1})$$

$$\text{s.t } \|u_t^{(i)}\| \leq V_{max}, \quad i = 1, 2, \dots, M$$



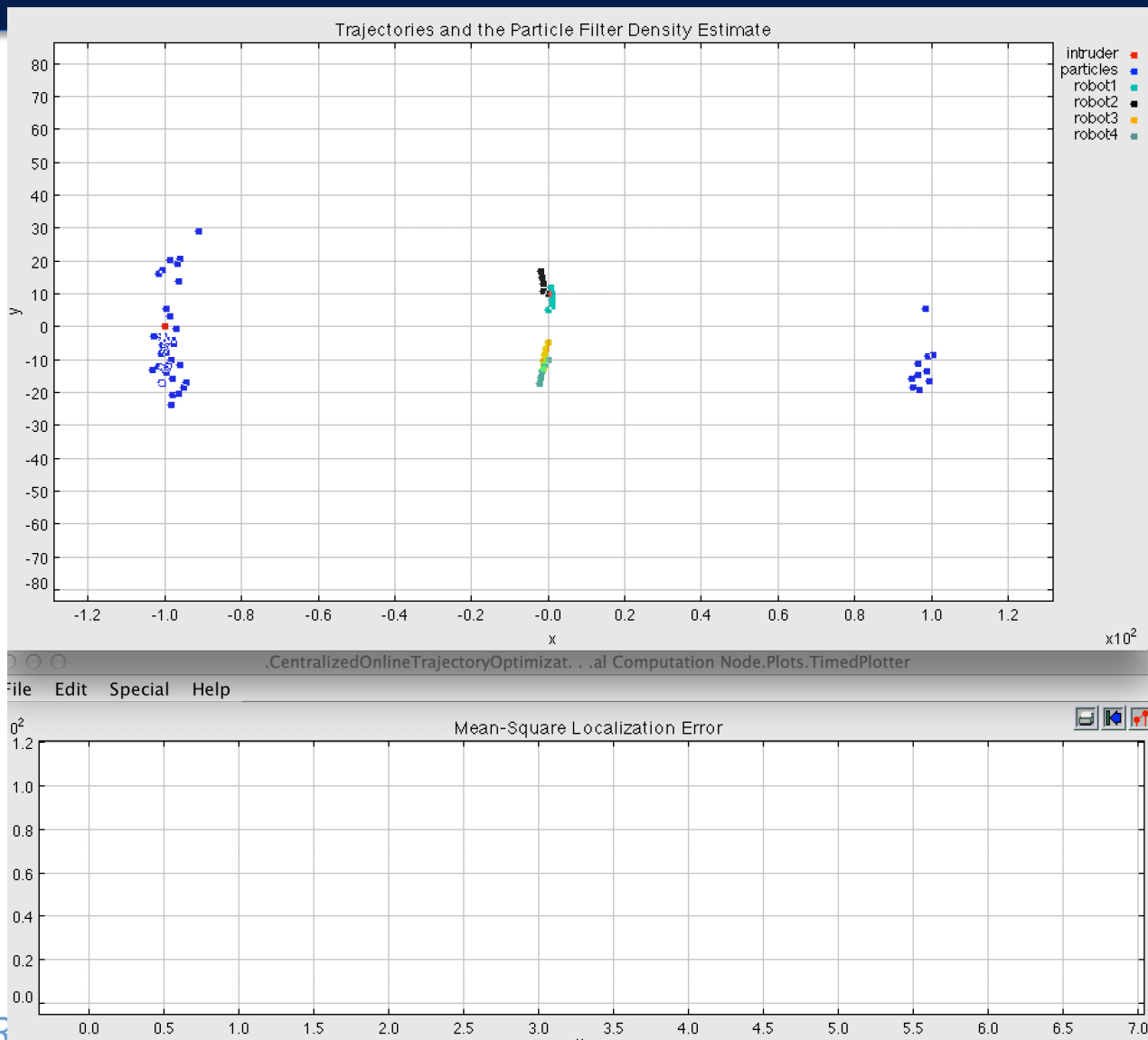


Cooperative Target Localization: Models



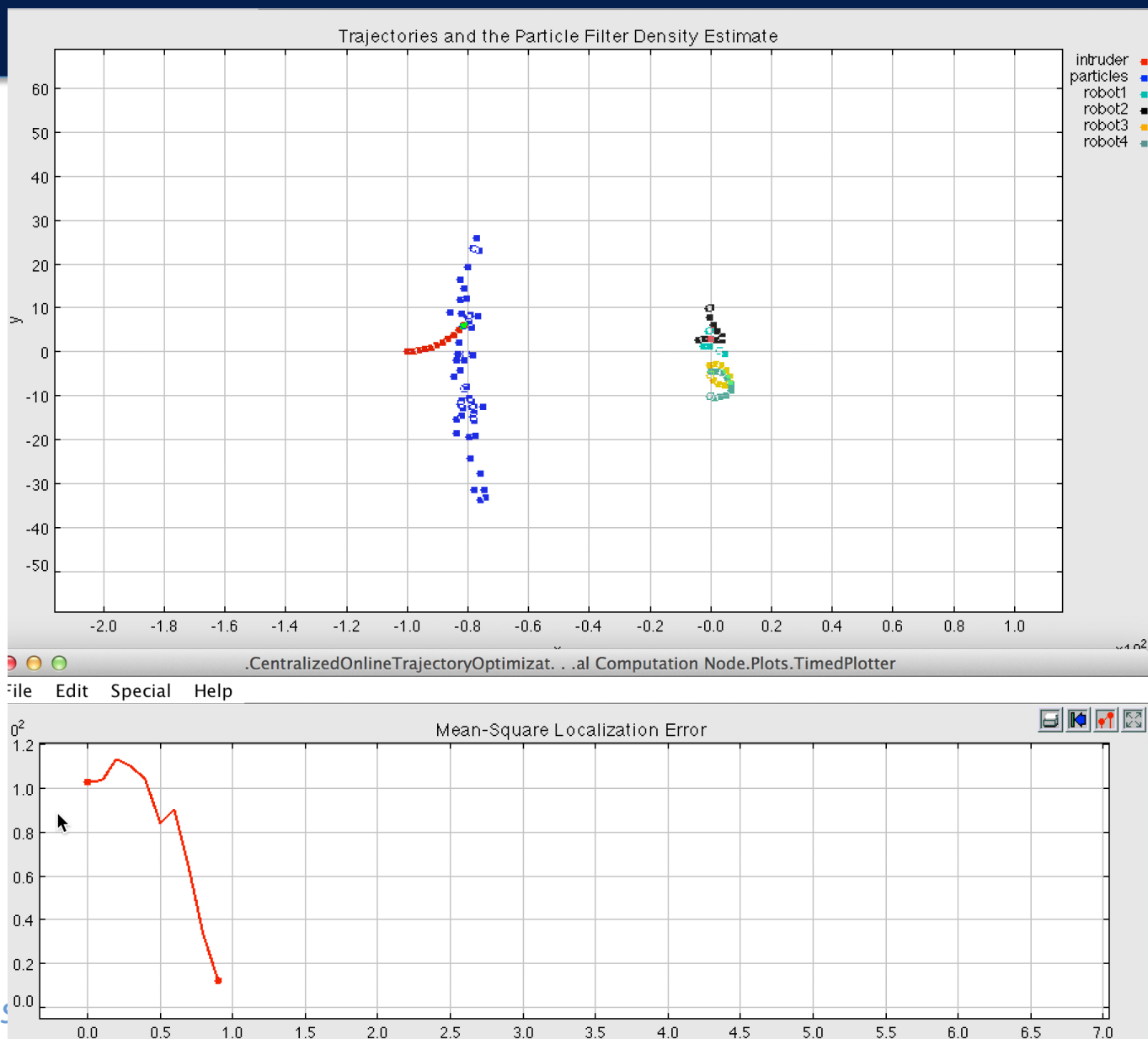


Demo: MI Maximization



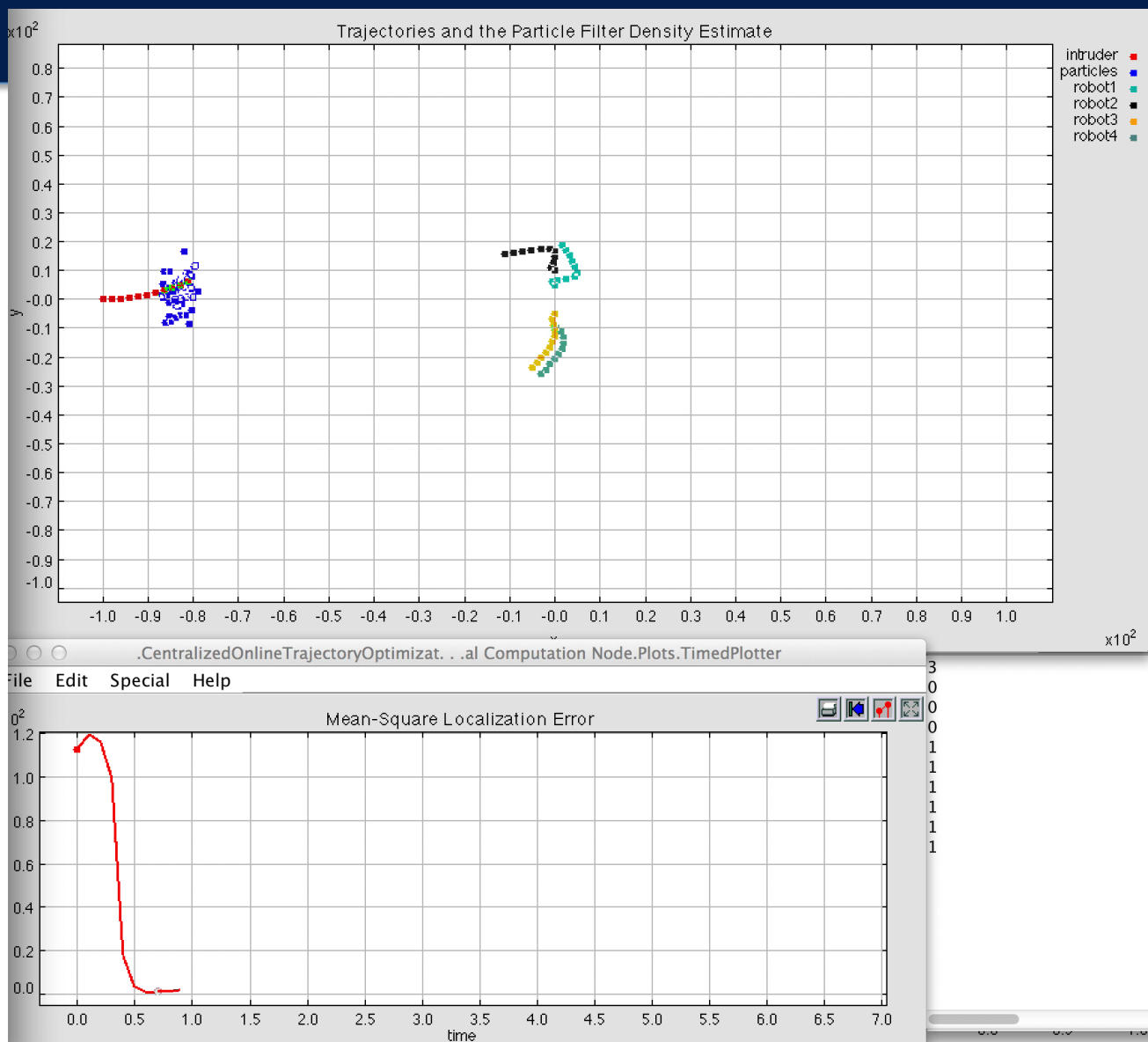


Demo: Direct Pursuit





Demo: Hybrid Approach - 1 Follower





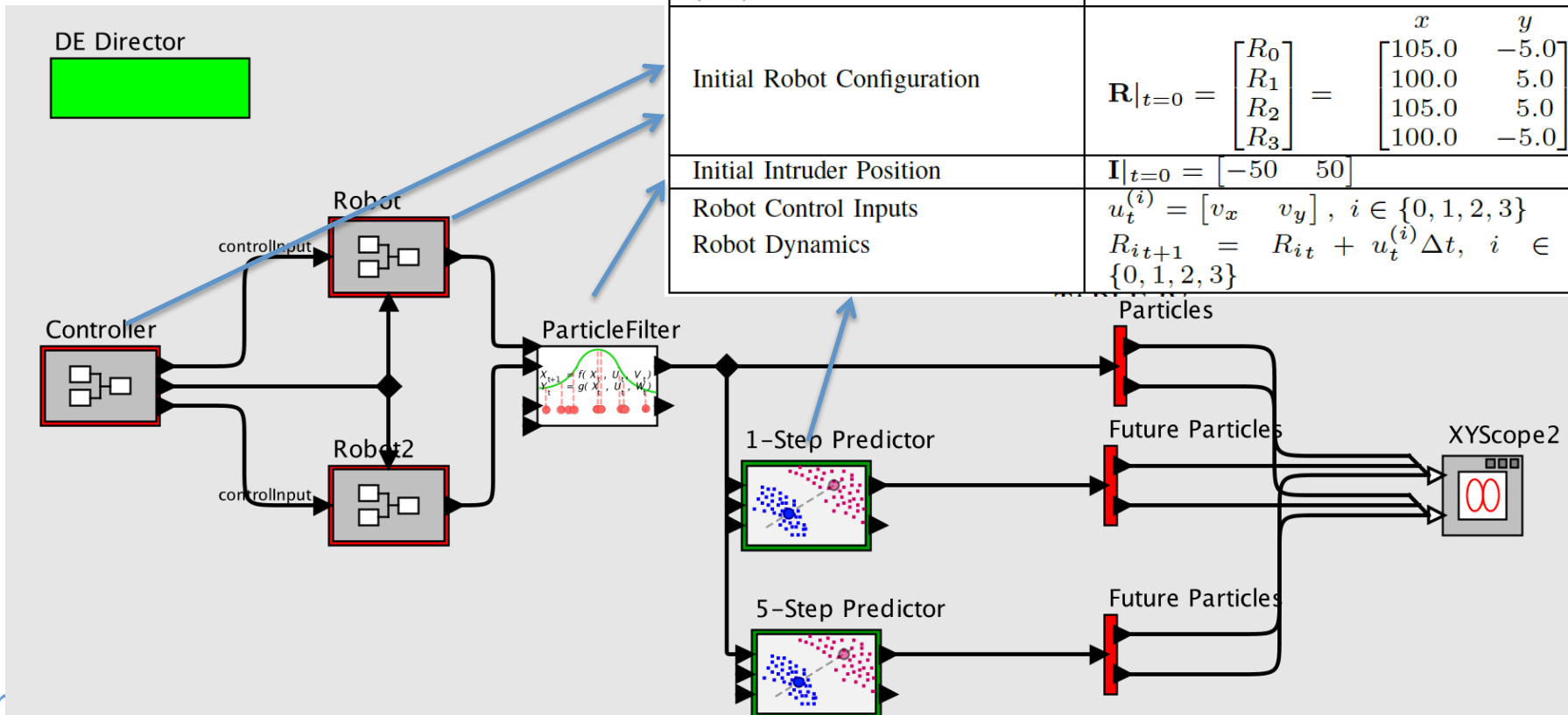
Bridging Actor-Oriented Modeling and ML Algorithms

- Goal: ML Algorithms that are aware of the system models
- Methodology: Implement measurement models and system dynamics as decorator actors in the system model
 - Easy to share, consistent models of underlying system models
 - Scalable and unambiguous ML algorithm design for non-experts



Shared State Space Models for Model Predictive Control

Simulation Parameter	Value
Size of Robot Team, M	4
Search space	200x200 units
Sensor measurement noise	$\nu_t \sim \mathcal{N}(0, 5.0)$
Maximum Robot Speed	20 units/s
Iteration Frequency	10 Hz
Target Dynamics	Circular Motion with $\omega = \pi/5$
Prior Belief on Target Position (π_X)	Uniform over search space
Initial Robot Configuration	$\mathbf{R} _{t=0} = \begin{bmatrix} R_0 \\ R_1 \\ R_2 \\ R_3 \end{bmatrix} = \begin{bmatrix} x & y \\ 105.0 & -5.0 \\ 100.0 & 5.0 \\ 105.0 & 5.0 \\ 100.0 & -5.0 \end{bmatrix}$
Initial Intruder Position	$\mathbf{I} _{t=0} = [-50 \quad 50]$
Robot Control Inputs	$u_t^{(i)} = [v_x \quad v_y], i \in \{0, 1, 2, 3\}$
Robot Dynamics	$R_{i,t+1} = R_{i,t} + u_t^{(i)} \Delta t, i \in \{0, 1, 2, 3\}$





Measurement Models and Dynamics as Decorators

DE Director



• Vtarget: 1

• Nout: 300

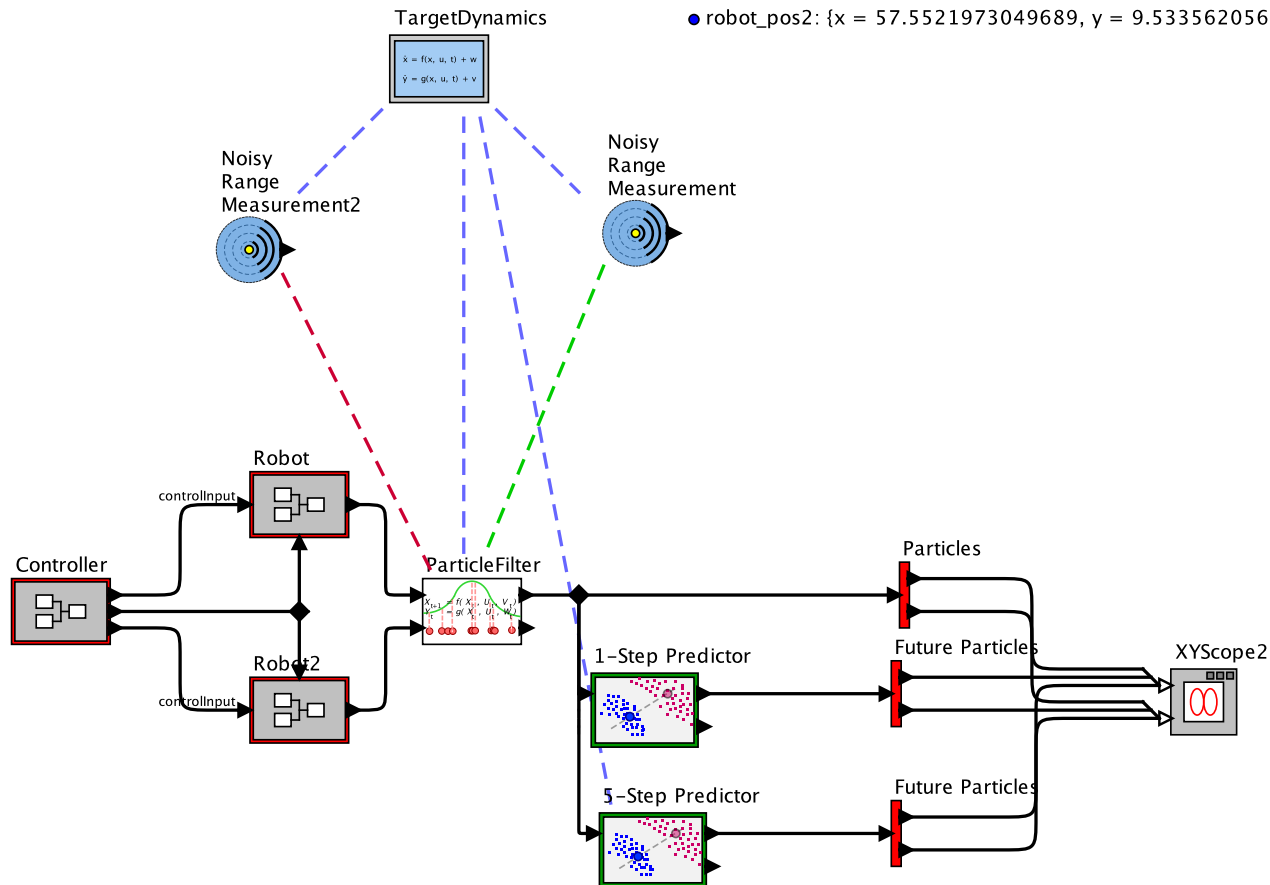
• Nparticles: 2000

Author: Ilge Akkaya

• target_pos: {x = 0.178875011368, y = 0.4636026512614}

• robot_pos: {x = 9.5335620567504, y = -2.4478026950311}

• robot_pos2: {x = 57.5521973049689, y = 9.5335620567504}





Measurement Models and Dynamics as Decorators

DE Director

• Vtarget: 1 • Nout: 300 • Nparticles: 2000

Author: Ilge Ak

• target_pos: {x = 0.1
• robot_pos: {x = 9.53
• robot_pos2: {x = 57



```
stateVariableNames: {"x","y"}  
prior: {random()*200-100,random()*200-100}  
processNoise: multivariateGaussian({0.0,0.0},{1.0,0.4;0.4,1.2})  
x_update: x  
y_update: y
```

TargetDynamics

$\dot{x} = f(x, u, t) + w$
 $\dot{y} = g(x, u, t) + v$

Noisy Range Measurement2

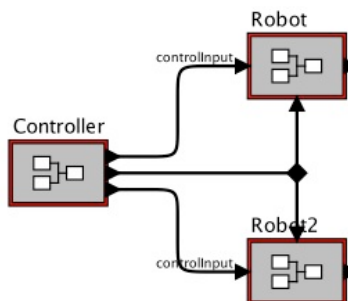
Noisy Range Measurement



Noisy Range Measurement Model

StateSpaceModel

```
z: sqrt((x-robot_pos.x)^2+(y-robot_pos.y)^2)  
noiseCovariance: [5.0]
```



ParticleFilter

1-Step Predictor

5-Step Predictor

Future Particles

Future Particles

ParticleFilter

RangeSensor2

StateSpaceModel

RangeSensor1

```
bootstrap: ☒  
lowVarianceSampler: ☒  
particleCount: Nparticles  
outputParticleCount: Nout
```

XYScope2



Target Localization: Adding a new Sensor

DE Director



• Vtarget: 1

• Nout: 300

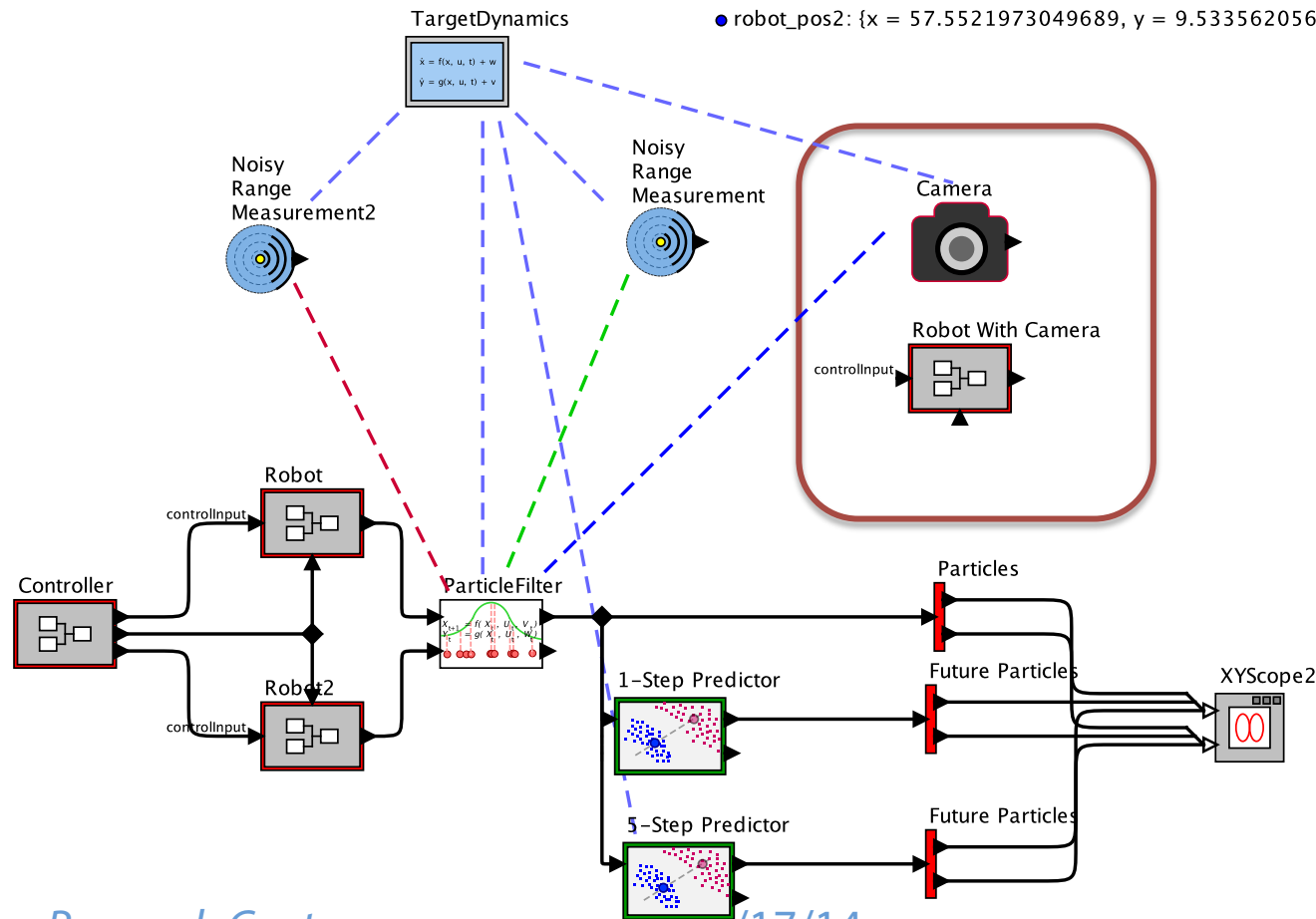
• Nparticles: 2000

Author: Ilge Akkaya

• target_pos: {x = 0.178875011368, y = 0.4636026512614}

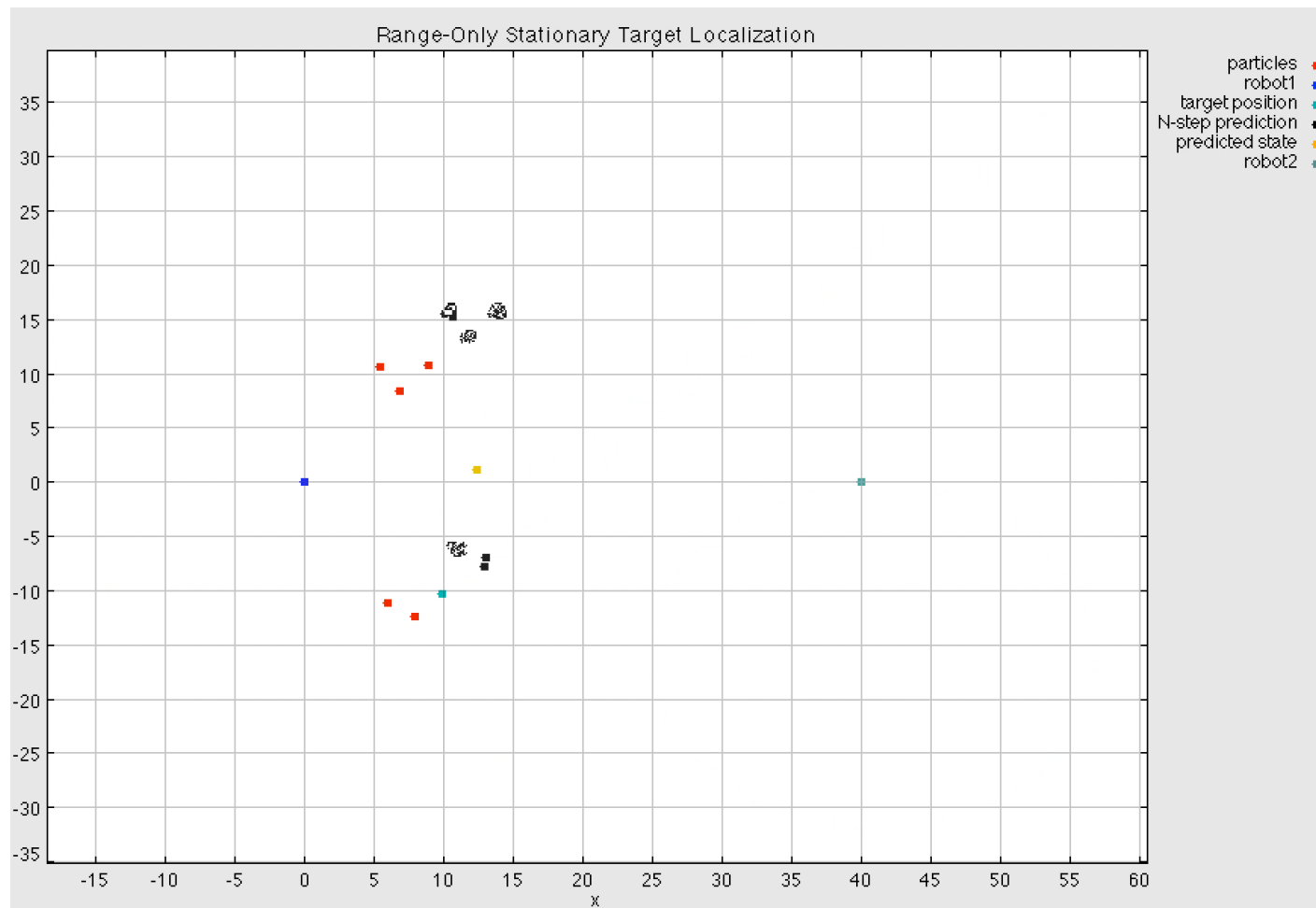
• robot_pos: {x = 9.5335620567504, y = -2.4478026950311}

• robot_pos2: {x = 57.5521973049689, y = 9.5335620567504}



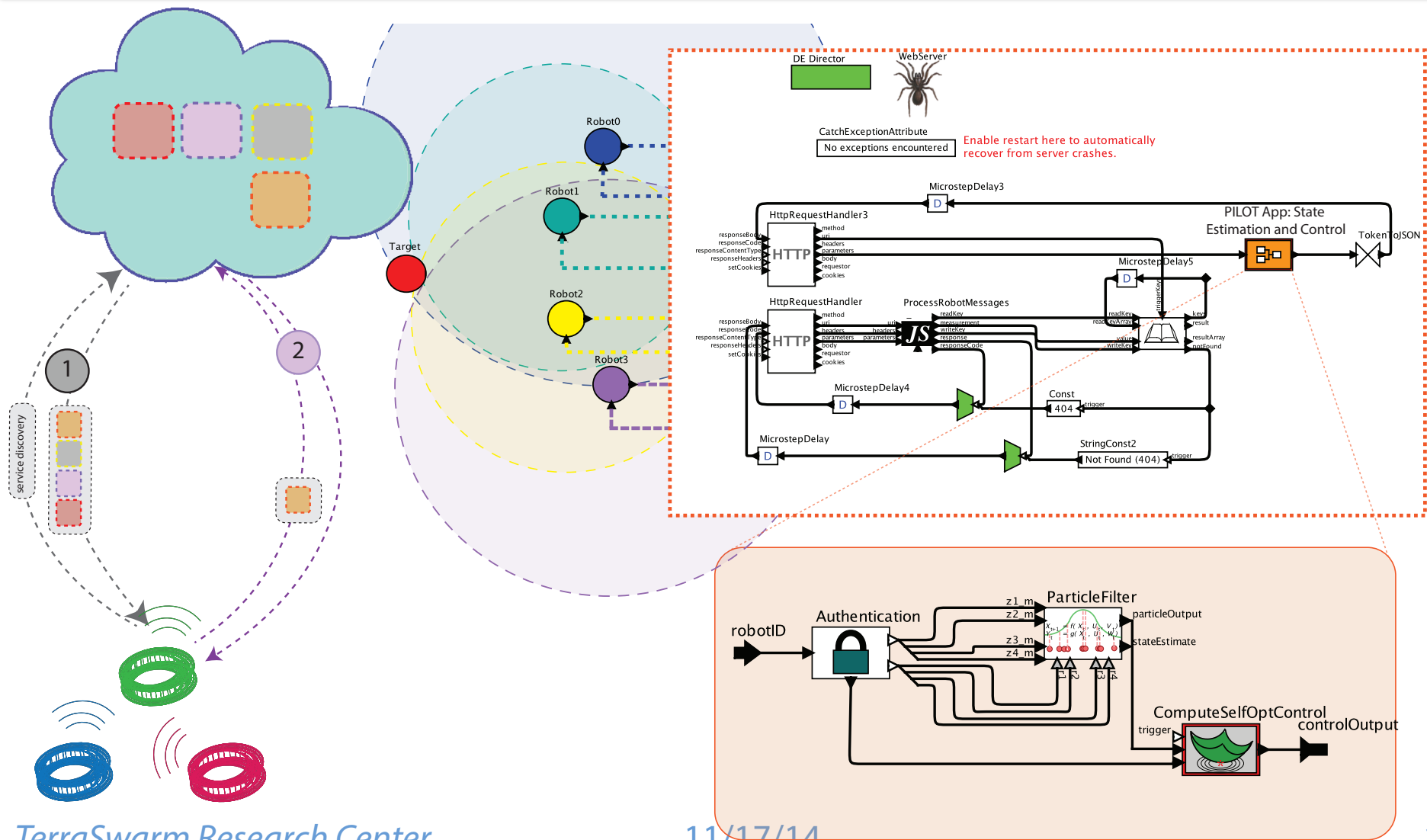


Demo: Prediction





ML and Optimization: Swarmlets

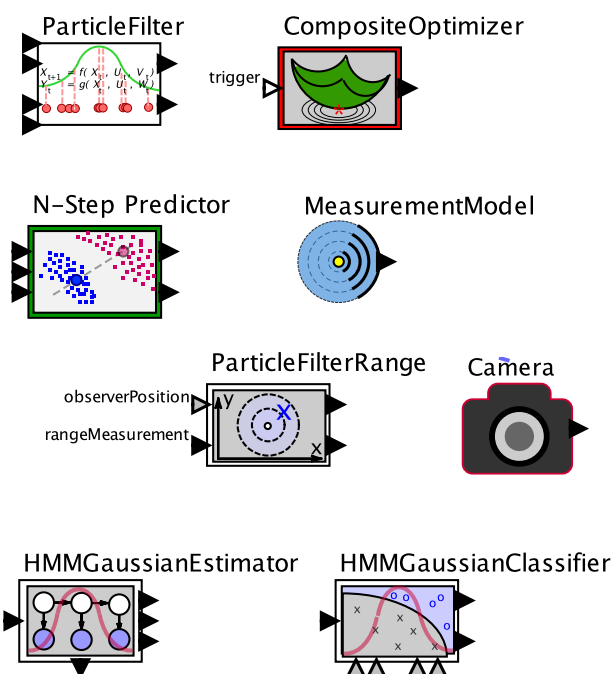




Conclusions

Presented an actor-oriented machine learning toolkit that is designed for

- ML and Optimization applications on **streaming data**
- Enhancing **programmability** of *swarmlets*
- *Actor libraries for common state-space dynamics and sensor models*





Looking Ahead

- Enhancing ML capabilities:
 - *Discrete Optimization Solvers*
 - *(Mixed) Integer Programming*
 - *Tool Integration: e.g., GMTK*
- *Developing Swarmlets: Providing Services to TerraSwarm Application Developers*
 - *More case studies*
 - *Anomaly detection*
 - *Multi-sensor fusion*



Demos: Available in Ptolemy II

Optimization and Machine Learning

Control Improvisation

- [Jazz Improvisation](#)

<http://chess.eecs.berkeley.edu/ptexternal/>

Optimization

- [Constrained Simple Linear Regression](#)
- [Simple Function Minimization](#)

Particle Filter

- [Multi Robot Intruder Tracking](#)
- [Online Robot Trajectory Optimization](#)
- [Online Robot Trajectory Optimization - Distributed Computation](#)
- [Open-Loop Target Localization - Single Robot](#)
- [Open-Loop Target Localization - Two Robots](#)
- [Multi-Observer Particle Filtering](#)
- [Particle Filter Range](#)

Probabilistic Models

- [Channel Fault Model](#)
- [Communication Anomaly Detection Using HMM Estimation](#)
- [Gaussian Mixture Model](#)
- [Gaussian Mixture Model Parameter Estimation](#)
- [Hidden Markov Model](#)
- [Hidden Markov Model Analysis](#)
- [Discrete-Time Markov Chain](#)



Thank You !

Questions?
Comments?