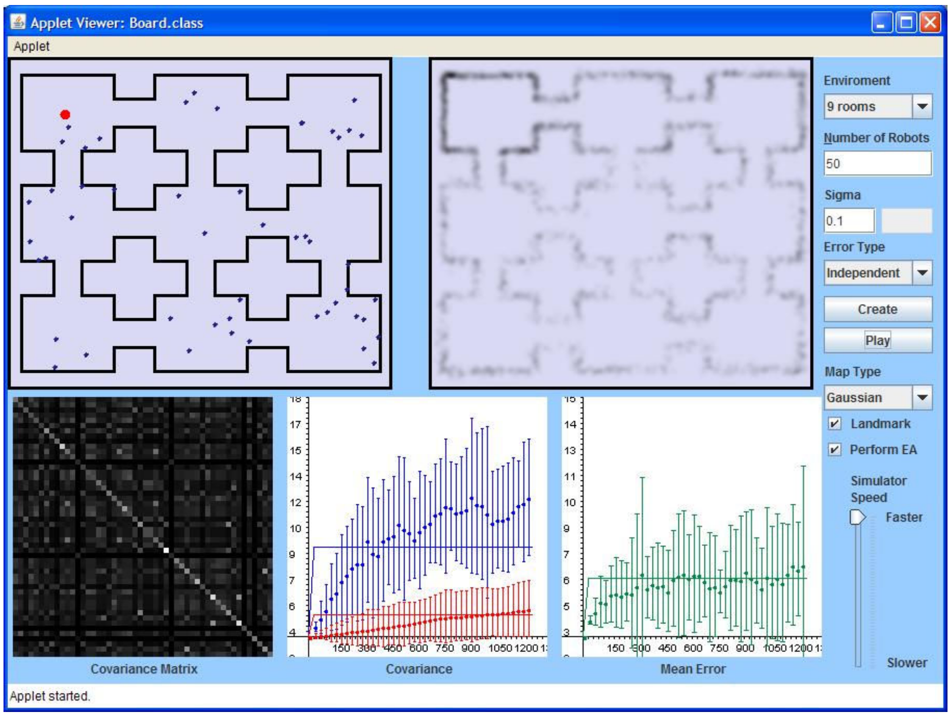
Intelligent Systems Simulation Project (236754)

A thermodynamic Approach to the Analysis of Multi-Robot Cooperative Localization



Applet Viewer: Board.class

Enviroment: 9 rooms

Number of Robots: 50

Sigma: 0.1

Error Type: Independent

Map Type: Gaussian

☑ Landmark

☑ Perform EA

Applet started.

Submitted by:

Amir Keren

Nimrod Eldan

<u>Under the guidance of</u> Yotam Elor

<u>Under the supervision of</u> Prof. Alfred M. Bruckstein

March, 2011

Table of Contents

Introduction

Our project presents a simulator based on a new approach to the simultaneous cooperative localization of a group of robots capable of sensing their own motion and the relative position of nearby robots. In addition, this project is an enhancement of a project which was handed by Oshrat Bonker and Ariella Avramovich on August 2010.This version includes new features and some bug fixes.

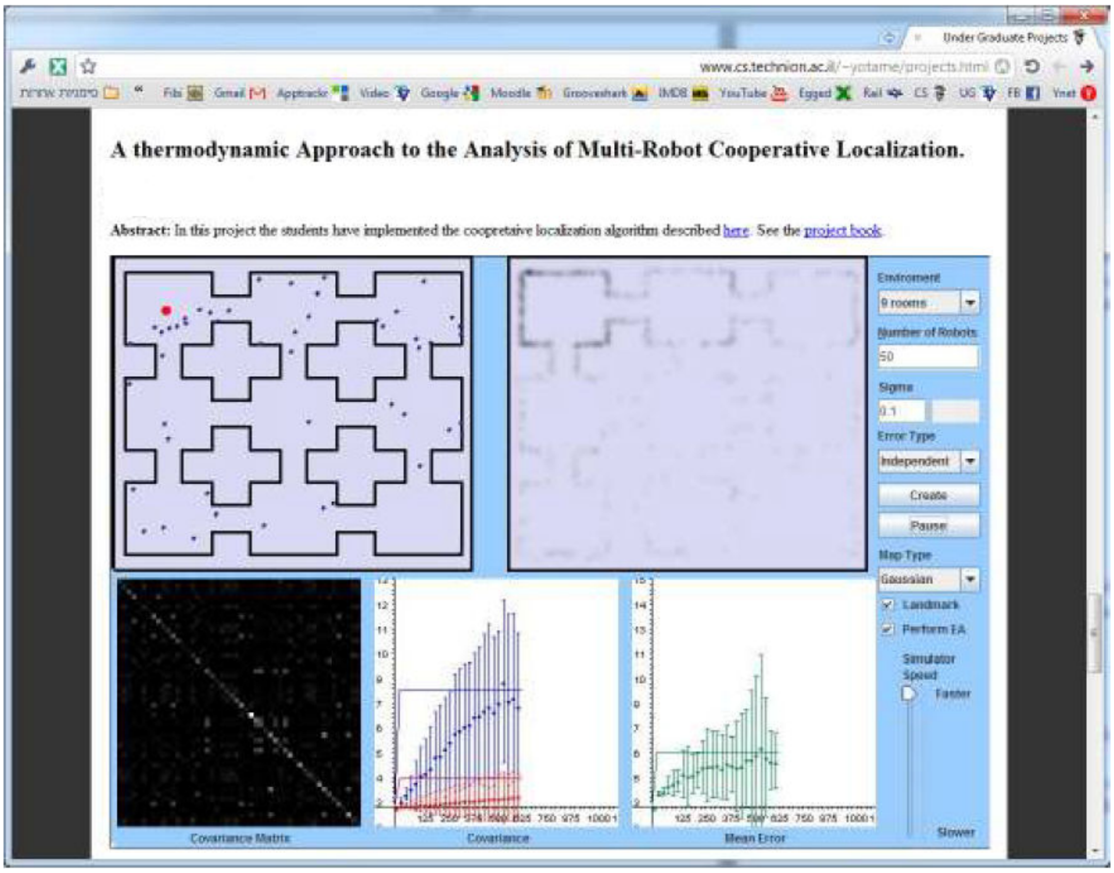
The simulator provides a solution based on the paper "A Thermodynamic Approach to the Analysis of Multi-Robot Cooperative Localization under Independent Errors" by Yotam Elor and Alfred M. Bruckstein.

Localization is the task of estimating the robot location and has been identified as one of the key problems in robotics. The localization problem can be roughly divided into two variants: In the first variant the robots can estimate their location by sensing their surroundings and comparing it with information they possess about the environment, while in the second variant, the robots have no such capabilities. Instead, every robot knows its initial location and updates its location estimation [1].

In this project we are focusing on the second variant. In the second variant of the localization problem it is assumed that, initially, every robot knows its location and uses audometry in order to track it (dead-reckoning) [1].

However, due to noisy sensor readings, in time, the estimation diverges from the robot real location. When a group of robots perform localization, the localization error can be reduced by sharing information between them [1].

The goal is not to derive hard lower or upper bounds, but rather to characterize the robots expected behavior and in particular, to predict the expected localization error [1].



A thermodynamic Approach to the Analysis of Multi-Robot Cooperative Localization.

Abstract: In this project the students have implemented the coopretaive localization algorithm described here. See the project book.

Environment: 9 rooms
Number of Robots: 50
Sigma: 0.1
Error Type: Independent
Map Type: Gaussian
✓ Landmark
✓ Perform EA

A preview of how the simulator looks embedded in a web page

[1] "A Thermodynamic Approach to the Analysis of Multi-Robot Cooperative Localization Under Independent Errors", Yotam Elor and Alfred M. Bruckstein, Faculty of Computer Science and the Goldstein UAV and Satellite Center.

New Features and Fixes

As mentioned in the introduction, our project is an enhancement of a project which was handed by Oshrat Bonker and Ariella Avramovich on August 2010, and it includes new features and bug fixes.

New features

New map type – in addition to the regular map type, the user can choose to display the robots' false location map as a Gaussian map. (More on the Gaussian map in the Algorithms section).

Two new error modeling types – in addition to the independent model, the user can choose between compass model and full audometry model. (More on compass model and full audometry model in the Algorithms section).

New Environment - the user can choose torus environment. This environment has no walls.

Landmark feature – enables a spot on the map which upon arriving to that spot causes the robots to initialize their location calculation mistakes.

Bug fixes

Display refresh problem – when running the project as a java applet, minimizing and maximizing the program would have caused display problems.

Mean error graph resize bug – now performs resize correctly.

Panel buttons reorganized - buttons are now organized in a more user friendly way.

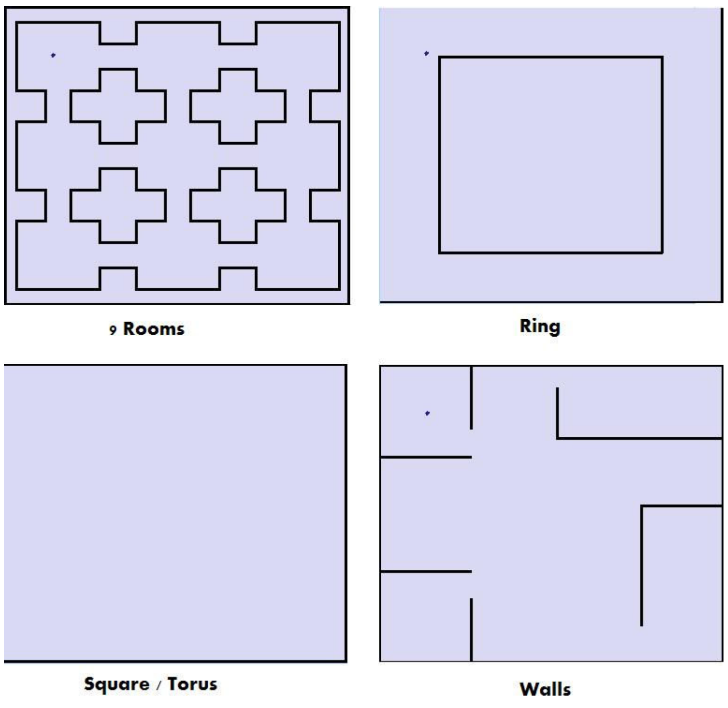
Simulator Description

The simulator allows the user to choose one of several different environments and to set different parameters such as number of robots, map type, the covariance of the localization error, animation speed control and others. According to the user preferences, the robots will patrol the selected environment and map how the area is constructed, but due to noisy sensor readings the mapping has errors and deviation.

The user is able to reduce these errors according to three different error models.

Whenever two robots are within sensing and communication range, they average their location estimations and by that reducing their errors.

In addition, there is a graph of the covariance of the localization error of any two robots, and a graph of the robots mean error, both will be explained later.



Some of the environments of the simulator

Implementation Design

Main classes

The class *CreateGraph* contains all the information about the environment the robots are in and the environment mapping, and is also responsible for displaying it correctly on the screen. The class contains an array of robots and three bi-dimensional arrays, one for the environment itself, one for the mapped environment and one for the Gaussian map. This class is responsible to invoke all the robots to make a single step and also to invoke the EA algorithm. In addition, the *Chart2* class uses some of *CreateGraph* information to draw the mean error graph.

The class *Robot* contains for each robot information about its location. The information includes the real location of the robot and the false location. The real location is the actual location the robot is at, and the false location is the location the robot thinks it's at, as a result of the errors of the sensor that are accumulated to the real location. The real and false locations are calculated according to algorithms that are described in the following section.

The class *covarianceMatrix* includes all the information about the covariance matrix. It contains a bi-dimensional array which represents the covariance matrix. With each step of the robots, we add $\sigma_0{}^2$ to the main diagonal of the matrix, and if the EA algorithm is applied, then every time two robots meet, we average their rows and columns by updating this matrix. When we draw this matrix on the screen, we draw every cell of the matrix in a color between black and white. The more white the cell[i][j] is, the more dependent robots i and j are. The *Chart* class uses this information to draw the covariance graph.

Algorithms

All algorithms are based on Yotam Elor and Alfred M. Bruckstein article: "A Thermodynamic Approach to the Analysis of Multi-Robot Cooperative Localization Under Independent Errors", Faculty of Computer Science and the Goldstein UAV and Satellite Center.

Robots movement - In the class *CreateGraph* we initiate for each robot its preliminary location, meaning its X_i and Y_i coordinates and the initial angle ϕ_i. Robots speed is set to be V_0. With every step, the robots' real location is updated according to the following:

$$x_i(t+1) = x_{i(t)} + \cos(\phi_i(t)) \cdot v_0$$
$$y_i(t+1) = y_{i(t)} + \sin(\phi_i(t)) \cdot v_0$$

If a robot collides with a wall we randomize a new angle for it.

Due to noisy sensor readings there is an error in each robot's location. Thus, we also calculate the "false" location, meaning the location where a robot thinks it is at. We mark the false location $X_i^{'}$ and $Y_i^{'}$
Our simulator presents three different error models:

<u>Independent Model</u>

The user sets the parameter $\sigma_0^{\ 2}$ which affect z.
$z \sim N(0,\sigma_0^{\ 2})$ implies that z is a random variable distributed normally in one-dimension with zero mean and variance of $\sigma_0^{\ 2}$.

$$x_i\,'(t+1) = x_i\,'(t) + \cos(\phi_i(t)) \cdot v_0 + z_{i_x}$$
$$y_i\,'(t+1) = y_i\,'(t) + \cos(\phi_i(t)) \cdot v_0 + z_{i_x}$$

Where $z_i \sim N(0,\sigma_0^{\ 2})$

<u>Compass Model</u>

The user sets the parameters $\sigma_v^{\ 2}$ which affect z and $\sigma_c^{\ 2}$ which affect w.

$z \sim N(0,\sigma_v^2)$ and $w \sim N(0,\sigma_c^2)$ implies that z,w are random variables distributed normally in one-dimension with zero mean and variance of σ_v^2, σ_c^2 accordingly.

$$x_i'(t+1) = x_i'(t) + \cos(\phi_i(t)) \cdot v_0 + z_{i_x}$$
$$y_i'(t+1) = y_i'(t) + \cos(\phi_i(t)) \cdot v_0 + z_{i_x}$$
Where $z_i \sim N(0,\sigma_0^2)$
$$\phi'_i(t+1) = \phi'_{i(t)} + w_i$$
Where $w_i \sim N(0,\sigma_c^2)$

Full Audometry Model

The user sets the parameters σ_v^2 which affect z and σ_t^2 which affect w. $z \sim N(0,\sigma_v^2)$ and $w \sim N(0,\sigma_c^2)$ implies that z,w are random variables distributed normally in one-dimension with zero mean and variance of σ_v^2, σ_t^2 accordingly.

$$x_i'(t+1) = x_i'(t) + \cos(\phi_i(t)) \cdot v_0 + z_{i_x}$$
$$y_i'(t+1) = y_i'(t) + \cos(\phi_i(t)) \cdot v_0 + z_{i_x}$$
Where $z_i \sim N(0,\sigma_0^2)$
$$\phi'_i(t+1) = \phi'_{i(t)} + w_i + \frac{d\phi_i}{dt}$$
Where $w_i \sim N(0,\sigma_t^2)$

Error Averaging (EA)

We are interested in the error (e_i), which is the distance between the robot's real location to the robot's false location.

$$\tilde{x} = x' - x$$
$$\tilde{y} = y' - y$$

Therefore the error is calculated as follows: $e_i = \sqrt{\tilde{x}_i(t)^2 + \tilde{y}_i(t)^2}$.

EA allows us to fix the errors of the robots. Each two robots that are within a radius r average their errors and update their false location accordingly:

$$x'_{robot_i} = x'_{robot_i} + \left\{ \frac{(x'_{robot_i} - x_{robot_i}) + (x'_{robot_j} - x_{robot_j})}{2} \right\}$$

$$x'_{robot_j} = x'_{robot_j} + \left\{ \frac{(x'_{robot_j} - x_{robot_j}) + (x'_{robot_i} - x_{robot_i})}{2} \right\}$$

The same calculation for y.

Covariance Matrix

In the class *covarianceMatrix* we handle a bi-dimensional matrix. This matrix is the covariance matrix of the localization errors of the robots at time t. At start, the matrix is initiated to zero.
When no correction mechanisms are applied, the covariance of the localization error of any two robots is zero. To be precise, for any $i \neq j \ \ \sigma_{ij}(t) = 0$.
The components on the main diagonal of the matrix grows linearly in time for any i, $\sigma_i^2(t) = t \cdot \sigma_0^2$ .When error correction mechanisms are applied, the process of updating the covariance matrix as a result of a meeting between robot i and robot j can be carried out by averaging rows i and j of the matrix and averaging columns i and j.
The evolution of the covariance matrix, when the robots follow EA, can be described as follows. Initially: covariance matrix = 0. Then, as the robots move, the values on the main diagonal of the matrix start growing. Due to meetings between robots, error from the main diagonal spread to the rest of the matrix. Finally, a semi-steady state is achieved in which the rate of removing error from the main diagonal almost equals the rate of adding error to it.

Graphs

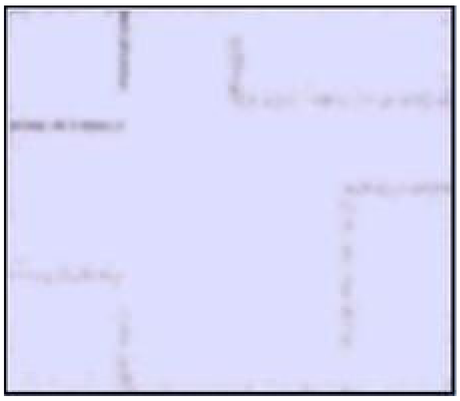
The simulator presents two statistical graphs. In the two graphs the X axis represents the time. In the covariance graph the blue dots represent the average of the main diagonal of the covariance matrix while the red dots represent the average of the whole matrix.

The sum value of the main diagonal is divided it by number of robots. Let's call it *diagonalAvg*. We calculate now the main diagonal deviation. For each robot sum up $(matrix_{i,j} - diagonalAvg)^2$. Finally we divide it with the number of robots and square root the result. This is the deviation of the main diagonal. We do the same thing for the whole matrix. We sum up all the components of the matrix. Let's call it *sum*. Then we calculate $matrixAvg = \frac{sum}{(number\_of\_robots)^2}$ which is the matrix average.
In order to calculate the deviation of the whole matrix, for each cell in the matrix we add $(matrix_{i,j} - diagonalAvg)^2$ to *graphSum.* The deviation of the graph will be $\sqrt{\frac{graphSum}{(number\_of\_robots)^2}}$.

In the mean error graph the green dots represent the mean error of the robots, and the calculating of the average error and its standard deviation is done the same as described above.

<u>Gaussian map</u>

In addition to the regular display, a Gaussian map has been added to give a different perspective of the map created by the robots as they collide with the obstacles laid out for them.



The Gaussian map is created using 10x10 matrices which normalize the values of the regular map using a Gaussian distributed variable, thus creating a "smeared" look of the hits to emphasize the locations hit more frequently than others.

Implementation Platform

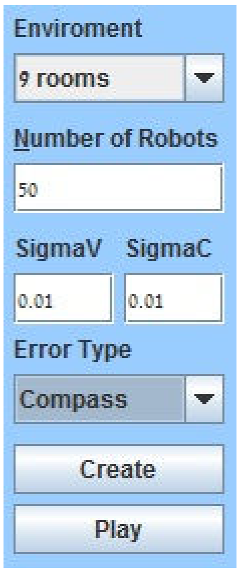
The project was implemented on Java Applet platform, using Eclipse IDE for Java Developers.

Using the simulator

In the right panel of the applet there are buttons and parameters that enable the user to change and control the simulation.

Parameters and Buttons

Environment - change the environment the robots are in. Available environments are: a square room, a room in the shape of a ring, a room consists of 9 rooms, a room with walls and a Taurus room. (This change will be applied when the after press the *Create* button).


Enviroment: 9 rooms
Number of Robots: 50
SigmaV: 0.01
SigmaC: 0.01
Error Type: Compass

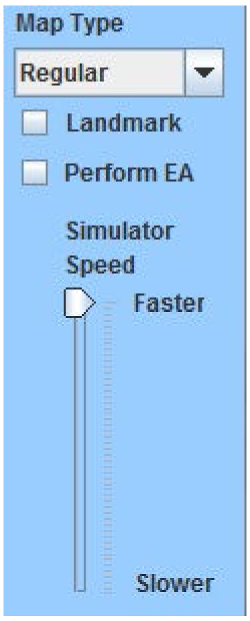
Number of Robots - choose between 1 to 250 robots. (This change will be applied when the after press the *Create* button).

Sigma, sigmaV, sigmaC, SigmaT – according to the error model, the user can change the mean for the robots' location and\or angle errors which are random variables distributed normally with a zero mean and variance of *Sigma, sigmaV, sigmaC, SigmaT* according to the error model. (This change will be applied when the after press the *Create* button).

Error Type – switch between the three different error models: Independent, Compass and Full Audometry.

Create – After setting the parameters, pressing this button creates a new simulation.

Play/Pause – After pressing the *Create* button, pressing the *Play* button starts the simulation. Pressing again pause the simulation and so forth.



Map Type - switch the display between normal and Gaussian.

Perform EA –choose whether to apply the algorithm to average the robots' errors or not. (This change will be applied immediately).

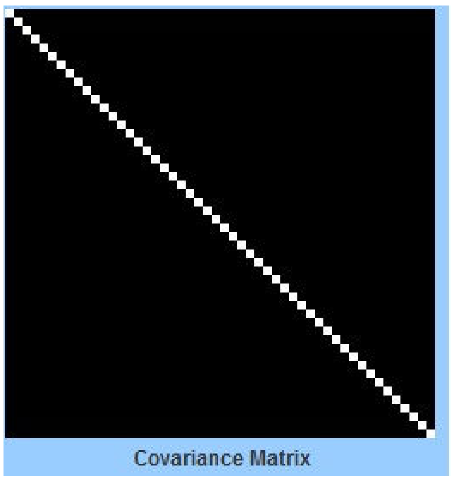
Landmark – activates and displays a fixed point on the map that initializes the error averaging history for every robot within a certain radius from it. (This change will be applied immediately and can only be activated with EA on).

Simulator Speed –control the animation speed (This change will be applied immediately).

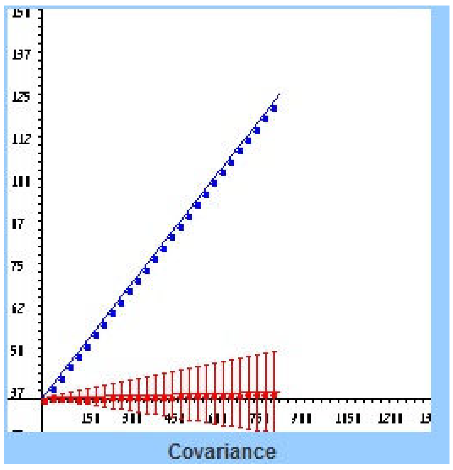
Matrix and Graphs

Beneath the maps there are the covariance matrix, the covariance graph and the mean error graph.

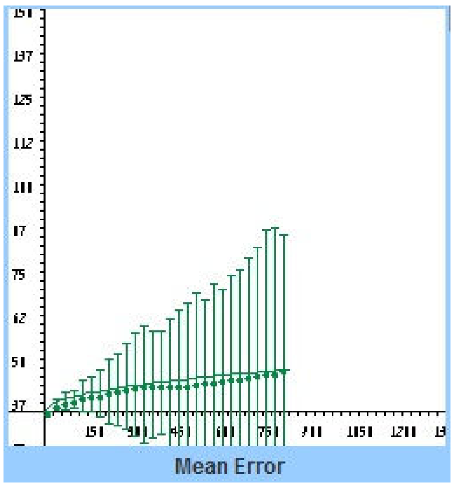
Covariance Matrix – The matrix is of size m*m, where m is the number of robots. Every cell of the matrix is shown in a color between black and white, while the more white the cell[i][j] is, the more dependent robots i and j are.



Covariance Graph - The X axis stands for the number of cycles passed since the start of the simulation. The Y axis stands for the covariance.
The blue points represent the average of the main diagonal of the covariance matrix, and the blue lines represent its standard deviation.
The red points represent the average of the rest of the covariance matrix, and the red lines represent its standard deviation.

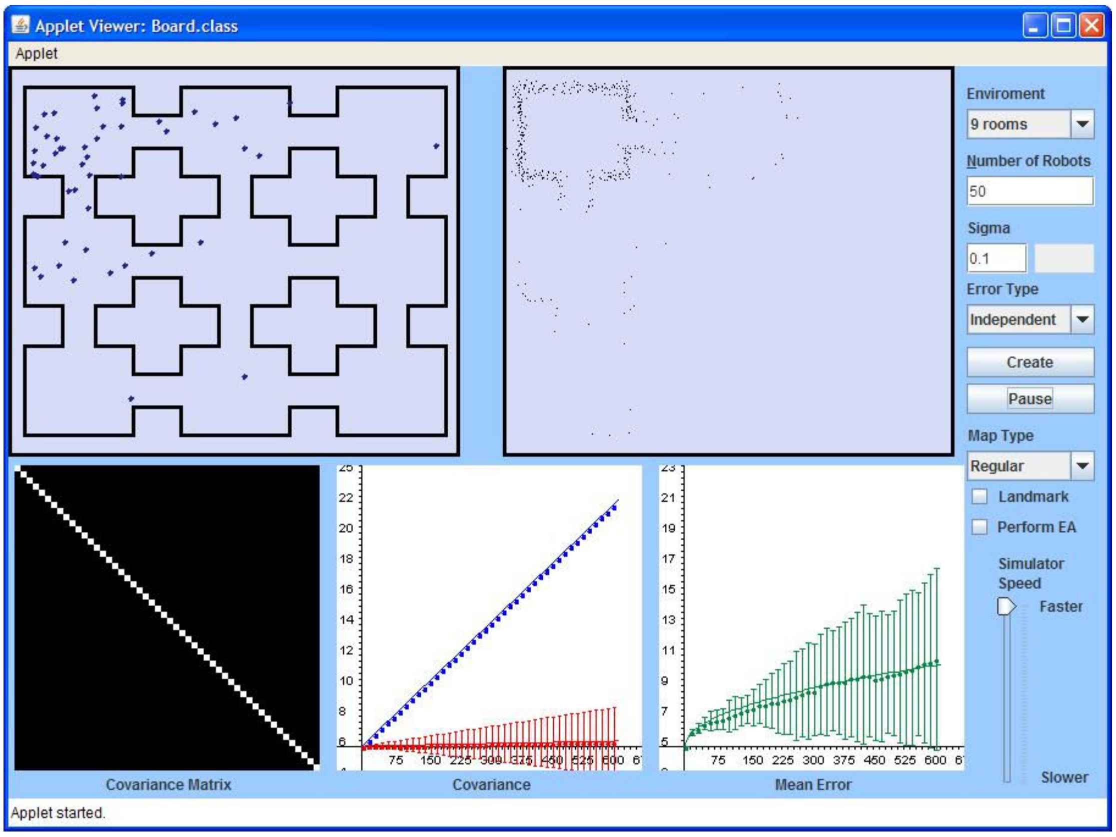


Mean Error Graph - The X axis stands for the number of cycles passed since the start of the simulation. The Y axis stands for the robots error.
The green dots represent the mean error of the robots while the green lines represent its standard deviation.



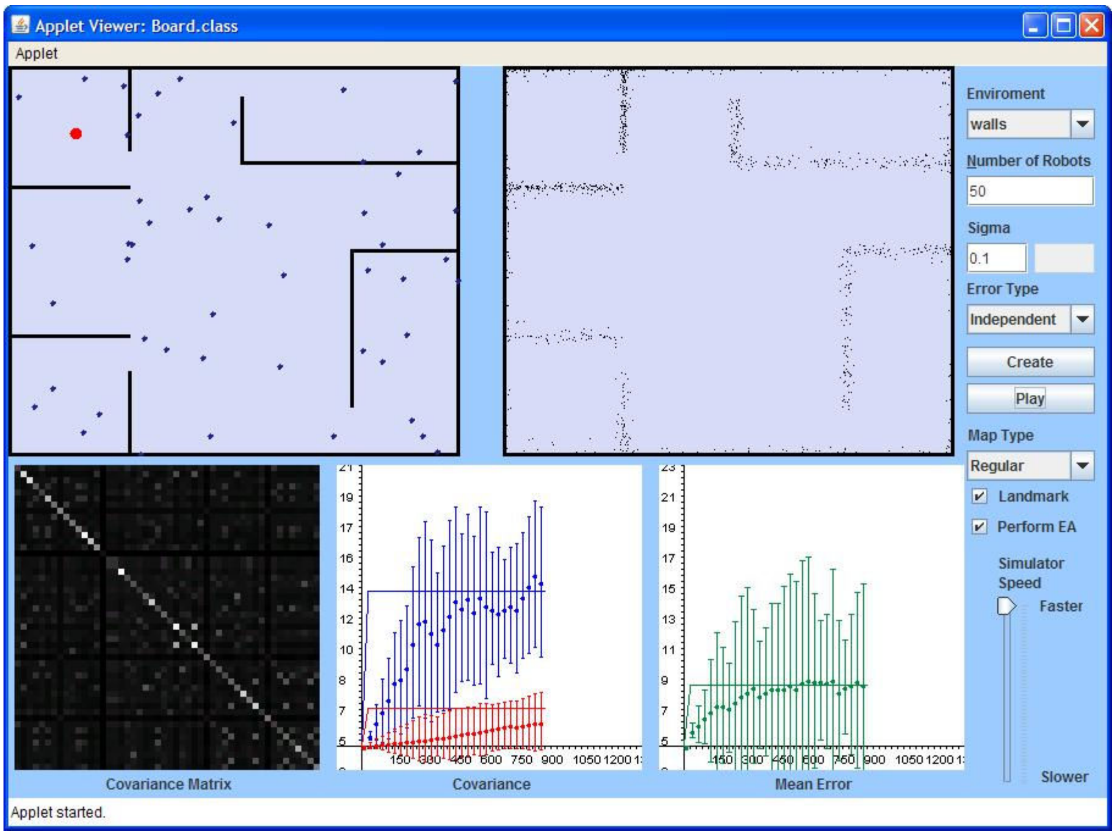
Screenshots

Running the simulator in *9 rooms* environment without EA:



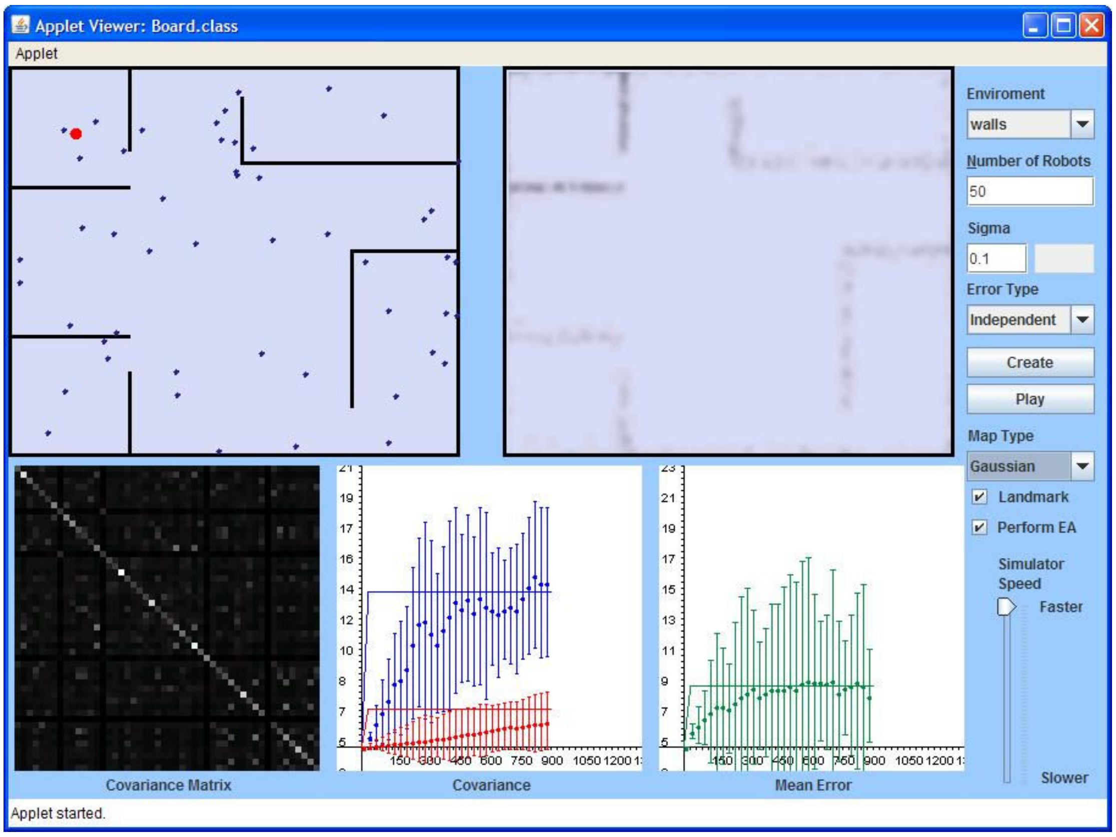
Enviroment: 9 rooms
Number of Robots: 50
Sigma: 0.1
Error Type: Independent
Map Type: Regular
Landmark: ☐
Perform EA: ☐

Running the simulator in *Walls* environment with EA:



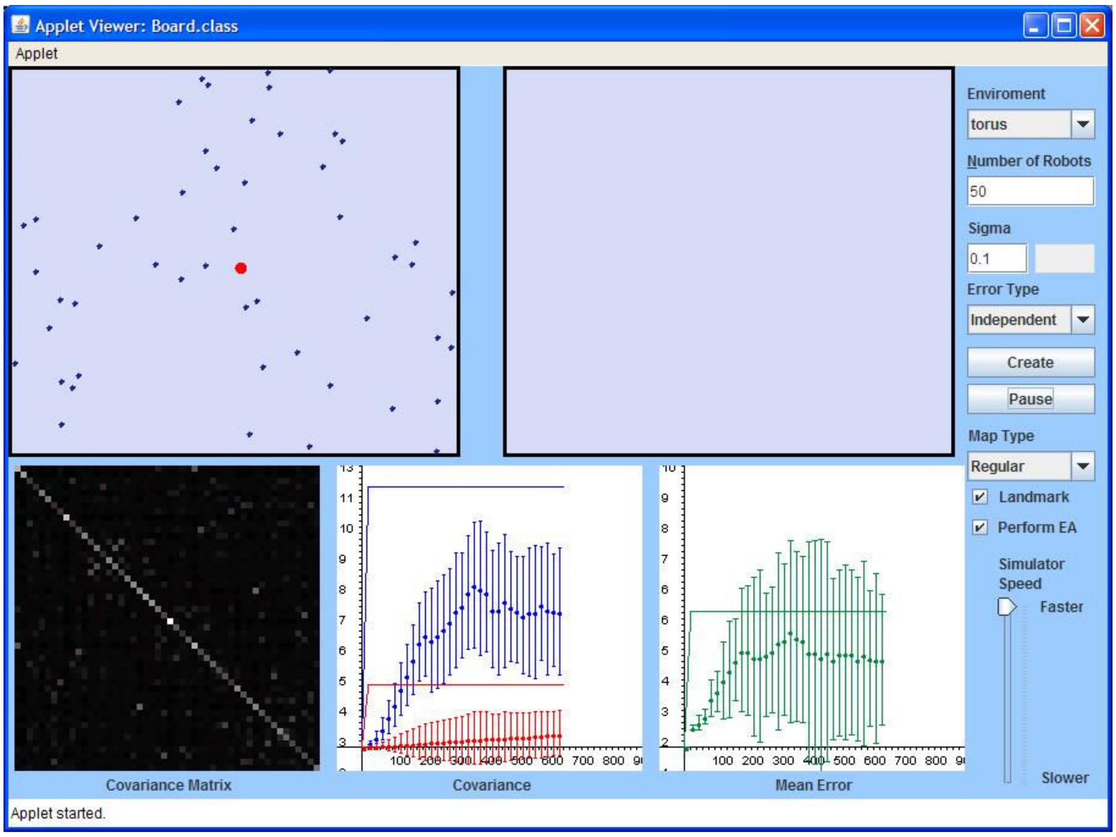
Enviroment: walls
Number of Robots: 50
Sigma: 0.1
Error Type: Independent
Map Type: Regular
Landmark: ✓
Perform EA: ✓

Running the simulator in *Walls* environment with EA, Landmark and Gaussian map:



Enviroment: walls
Number of Robots: 50
Sigma: 0.1
Error Type: Independent
Map Type: Gaussian
✓ Landmark
✓ Perform EA
Applet started.

Running the simulator in *Torus* environment with EA and Landmark:



Enviroment: torus
Number of Robots: 50
Sigma: 0.1
Error Type: Independent
Map Type: Regular
✓ Landmark
✓ Perform EA
Applet started.