SensorNet - Report

COMP 3203 – Principles of Computer Networks

Professor Evangelos Kranakis

December 13th, 2016

Chantal Forget - 100886390

Uri Schoijett - 100771318

Contents

[Operational Details 3](#_Toc468892264)

[Trials 3](#_Toc468892265)

[Single Trial 3](#_Toc468892266)

[Multi Trial 5](#_Toc468892267)

[Class Diagram 5](#_Toc468892268)

[Simple Coverage 6](#_Toc468892269)

[Implementation Details 6](#_Toc468892270)

[Issues 7](#_Toc468892271)

[Rigid Coverage 8](#_Toc468892272)

[Implementation details 8](#_Toc468892273)

[Issues 9](#_Toc468892274)

[Graphs 9](#_Toc468892275)

[Future Improvements 10](#_Toc468892276)

[Appendix 11](#_Toc468892277)

[Figure 1. Blank application. 11](#_Toc468892278)

[Figure 2**.** Result of a single trial of the Rigid Coverage algorithm. 12](#_Toc468892279)

[Figure 3. Result of a single trial of the Simple Coverage algorithm. 13](#_Toc468892280)

[Figure 4. Result of multiple trials of the Simple Coverage algorithm. 14](#_Toc468892281)

# Operational Details

As depicted in Fig. 1, the application accepts user input of numbers greater than 0 for both the number of sensors and the radius. Though the application accepts numbers greater than 1 for the radius, in order for the algorithm to best be demonstrated it is recommended to choose a radius between 0 and 1.

As depicted in Fig. 1,the application expects the user choose the correct algorithm to run. If the user wishes to have multiple trials run, they can select the "Multi Trial" radio button and input a positive number for the number of trials.

Once the user has inputted all correct values needed to run the algorithm, the application creates a list of sensors, assigning each a radius and a randomly selected location between 0 and 1. The sensors are then ordered in ascending order by their location.

# Trials

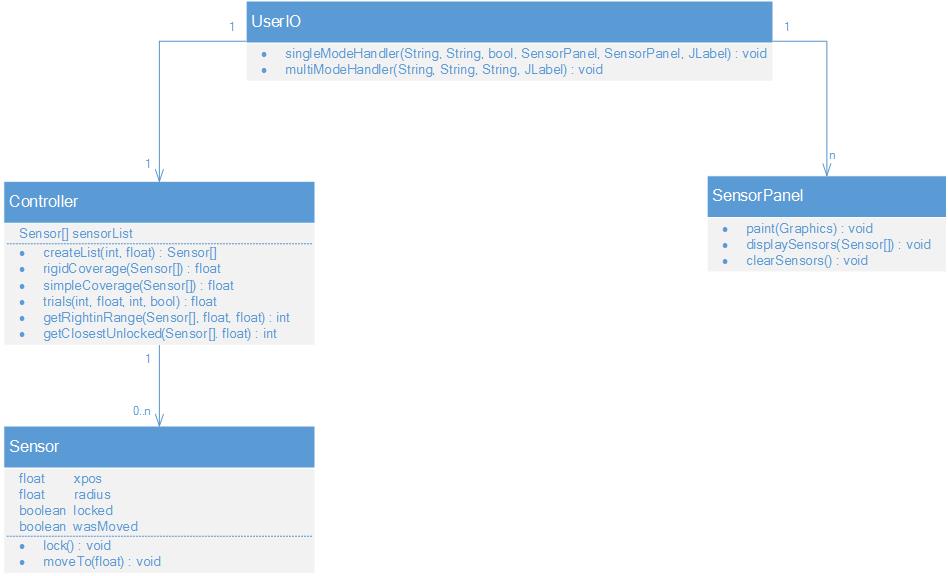
## Single Trial

When the user selects single trial, the algorithm selected is run only once. The initial randomized positions for the sensors are shown on the Graphical User Interface (GUI), as shown in Fig. 2 and Fig. 3**.** Once the algorithm completes, the user is shown the results, with the sensors that have been moved depicted in red, and the sensors that have been locked with a grey circle around them. The total movement for the single trial is shown at the bottom of the GUI, beneath the graphs depicting the sensors final positions.

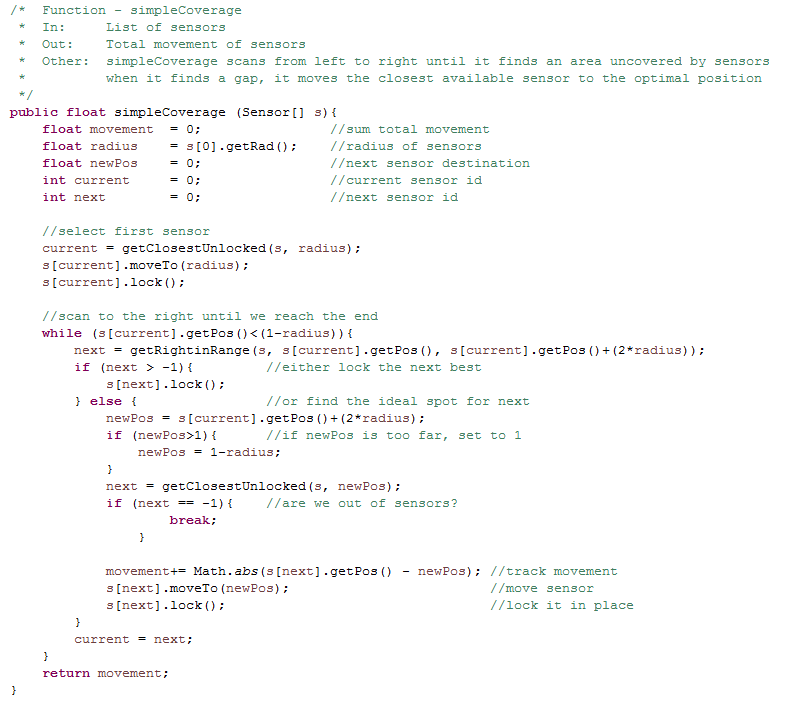
## Multi Trial

In multi trial runs, the user must specify the number of trials they wish to execute. Once the algorithm desired in selected, the user is simply shown the average movement of the sensors for the number of trials run. Before and after sections of the GUI are left blank when multiple trials are performed, as the information displayed would be useless to the user.

# Class Diagram



# Simple Coverage



## Implementation Details

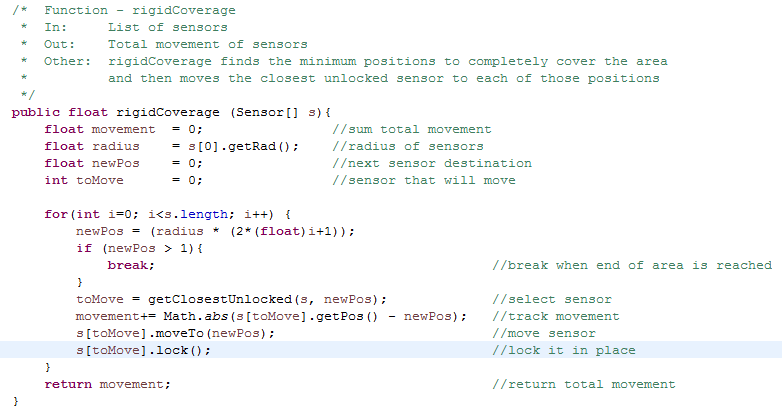
The simple coverage algorithm begins by selecting the closest unlocked sensor to the radius and moving the sensor to the position. This will happen regardless of if radius is already covered by another sensor. Following this, the algorithm iterates through the list of sensors while the position of the current sensor being considered is less than 1-radius. This condition allows the algorithm to scan through the list of sensors from left to right, as the sensors are in ascending order by position.

To find the next sensor the algorithm by looks for the sensor to the rightmost range of the radius of the current sensor. If a sensor is found in that range it is locked and moved to that position. Otherwise, the algorithm will look for the closest unlocked sensor to the uncovered area to the right of the current sensors radius. The difference here is that the latter will allow for any unlocked sensor to be moved where the former attempts to minimize movement. If the algorithm finds an unlocked sensor it locks it and moves it to the new position. If no unlocked sensor is found, then the algorithm knows that all sensors have been used and the algorithm is complete.

## Issues

The main problem with this implementation is that in some iterations, the area is not completely covered, even if there is a sufficient number of sensors for it to be covered completely. This is due to the implementation choices made. As this algorithm allows for overlap, the algorithm will lock sensors even if they overlap with another sensor. As a result of the algorithm going from left to right, sensors that have overlap but are locked can no longer be moved to the right to cover areas without sensors. This allows for situations that would otherwise be covered if overlap was not allowed, and results in the algorithm failing in some situations.

# Rigid Coverage



## Implementation details

Rigid Coverages implementation is quite simple. The algorithm will find a dominant set of positions for sensors that will cover the whole area between 0 and 1. While it finds these positions, it goes through the list of sorted sensors and finds the sensor closest to the current dominant position the algorithm is currently considering. The sensor chosen to be moved to that dominant position must be unlocked, which is to say that the algorithm has not previously moved it.

By virtue of having a dominant set of positions in which a sensor will be moved to, when the algorithm is re-run with the same radius, no matter how many sensors are specified the dominant set of positions will remain the same.

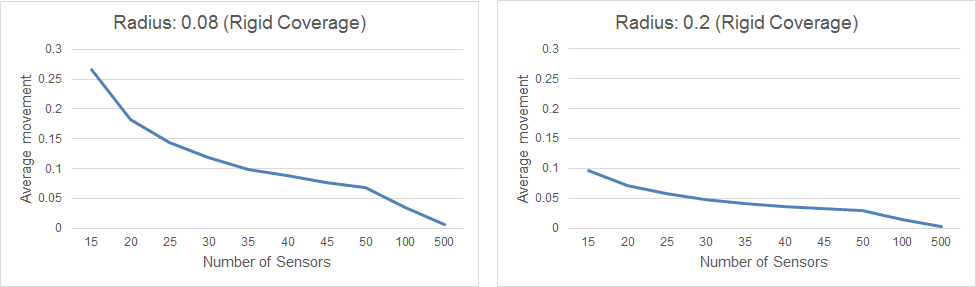
Furthermore, unless a sensor was already in the exact position required by the dominant set that, a sensor will be moved to that position.

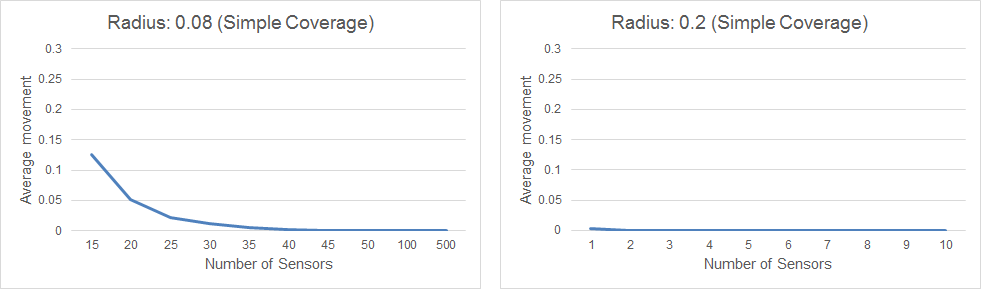
## Issues

The algorithm has unnecessary movement when there are more sensors than are needed. If there are 500 sensors with radius 0.1, the algorithm will still move a sensor to the exact position the algorithm needs for a sensor to be at every point in the dominant set. This creates unnecessary movement in a set of sensors that have already covered the whole area.

# Graphs

For each of the algorithms we ran a series of trials to compare their efficiency. We chose to graph the results of r=0.08 and r=0.2 across a number of sensors from 15-500. Each point is measured using the multi trial functionality over 1000 attempts to get a statistically valid average.





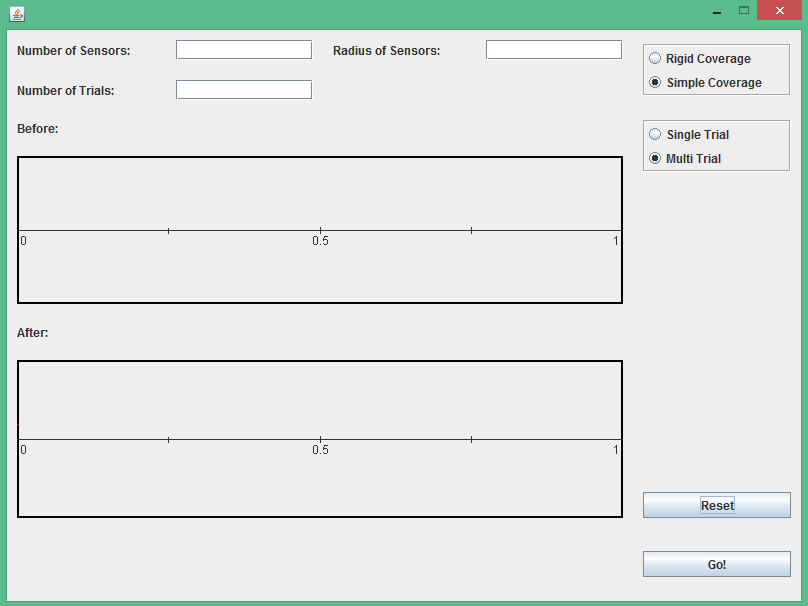
As expected, the average movement trends downwards as you add more sensors or increase the radius. More interestingly, the simple coverage method appears to be significantly more efficient than rigid coverage, especially with higher radii.

# Future Improvements

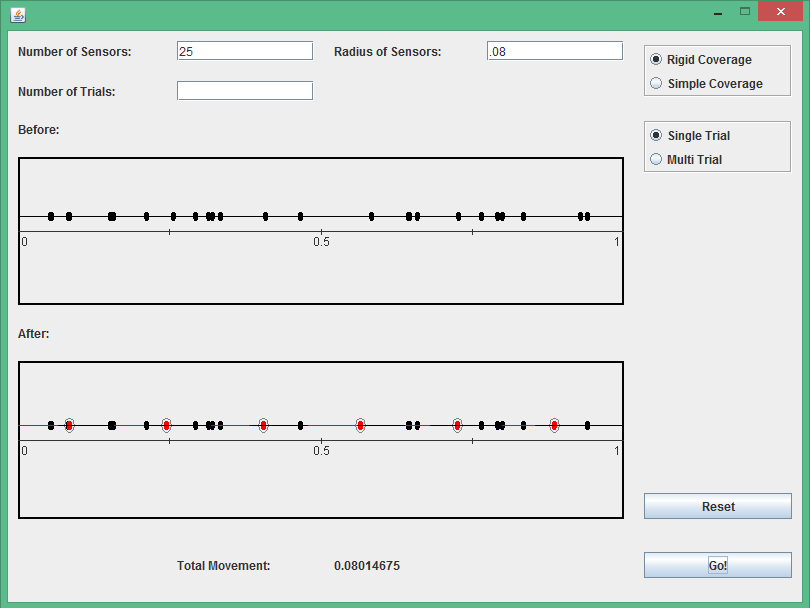
Future improvements that we did not have time to implement would include allowing the user to input multiple differing radii for the sensors. Also, we would like to allow the user to choose an option that has the application randomly generate radii, as this would make the sensors more realistic.

Appendix

### Figure 1. Blank application.



### Figure 2**.** Result of a single trial of the Rigid Coverage algorithm.



### Figure 3. Result of a single trial of the Simple Coverage algorithm.

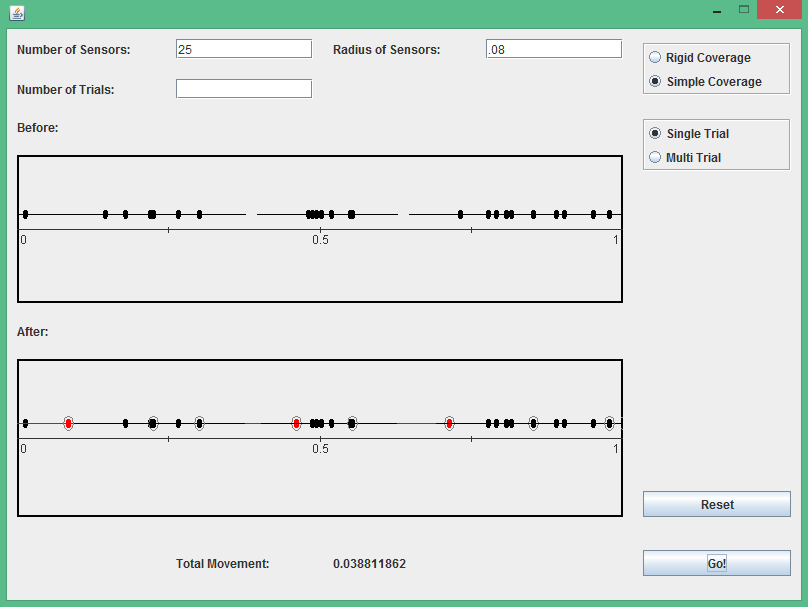


Figure 4. Result of multiple trials of the Simple Coverage algorithm.  
