Neural Network-based Multiple Robot Simultaneous Localization and Mapping

Sajad Saeedi[†], Liam Paull[†], Michael Trentini[⋆] and Howard Li[†]

[†]COBRA Group at the University of New Brunswick, Fredericton, Canada, http://www.ece.unb.ca/COBRA/, {sajad.saeedi.g, liam.paull, howard}@unb.ca

[⋆]Defence Research and Development Canada, Suffield, Alberta, Canada, Mike.Trentini@drdc-rddc.gc.ca

1 Introduction
- Problem statement
- Significance
- Contributions of research

2 Multiple Robot SLAM
- Overview of proposed method
- Self Organizing Map (SOM)
- Map Segmentation
- Clustering with SOM
- Relative Orientation
- Relative Translation

3 Experimental Results
- Experiment1
- Experiment2

4 Conclusion

Problem statement

In this research, a neural network-based map fusion for SLAM with multiple robots has been developed

- Given: Two occupancy grid maps, each developed by a robot.
- Find: The transformation which fuses two maps.
  - This is different than image registration, since there is no a priori knowledge about the shared areas in maps.
  - The algorithm should run fast.

Significance

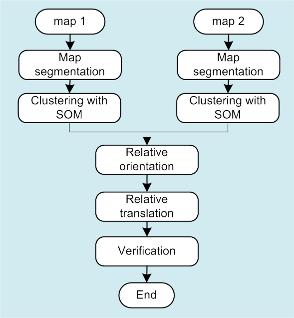
- Exploration and mapping can be done faster and more accurately by multiple robots.
- A distributed system is more robust.
- Applications in collaboration based operations: fire fighting in forest and urban areas, rescue operation in natural disasters, cleaning marine oil spills, underwater and space exploration, security, surveillance and maintenance investigations.
- Processing time in these operations is required to be as less as possible.

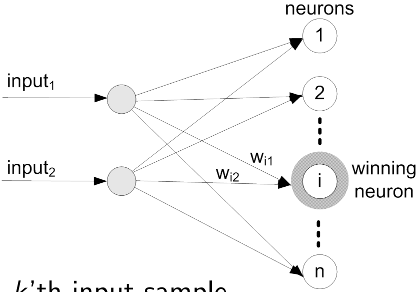
Contributions of research

- A high level map segmentation algorithm for occupancy grid map preprocessing.
- Application of SOM to cluster the preprocessed map.
- Estimation of the relative transformation matrix of two maps using the cluster points.
- The use of surface norms to associate cluster points from two different maps.

Overview of proposed method

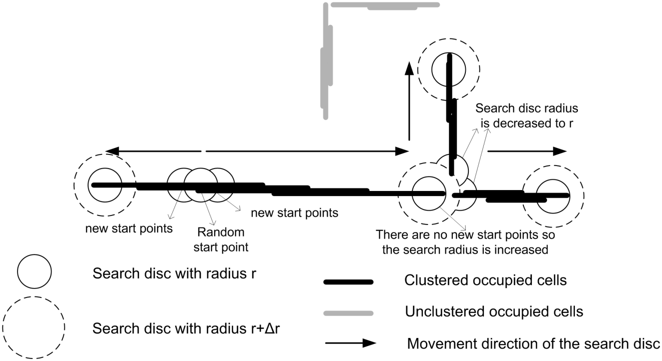
Main idea: Use neural networks to cluster each map into a few points. This will downscale the map, preserving the spatial information of the map and makes further processings faster.





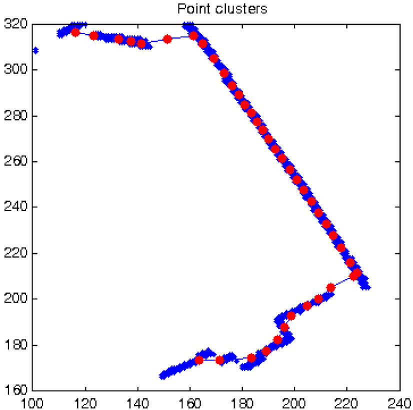
- $x(k) \in \mathbf{R}^2$: k'th input sample,
- $w_i(k) \in \mathbf{R}^2$: weights computed for the i^{th} neuron.
- weight update: $w_i(k+1) = w_i(k) + h_i(k)(x(k) - w_i(k))$
- h_i: the neighborhood function. The neuron with the minimum distance is called the winner
- advantage: unsupervised training (no need for output patterns)

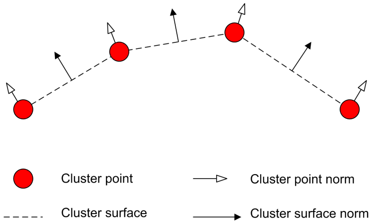
- Problem: the map is usually composed of a few major segments which are relatively far from each other. This does not let the SOM operate properly.
- Solution: The segmentation identifies discontinuous segments of obstacles located far enough from each other that they should be considered as separate.



Clustering by SOM has two main properties:

- clusters are features of the map.
- clusters downscale the map.
- training is unsupervised.

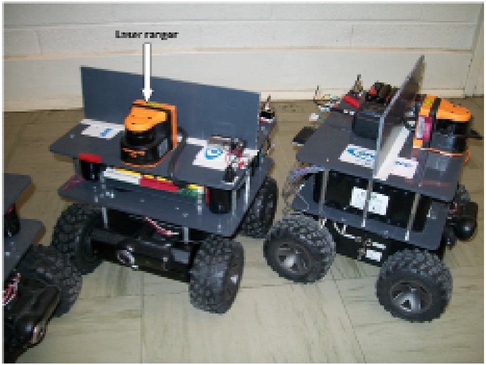




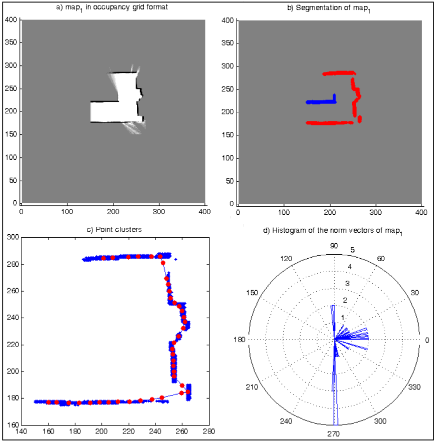
- The relative orientation between the two maps is determined by performing a 360° histogram on the directions of the cluster surface norms, and then matching the histograms of the two maps.
- Radon transform can be applied to tune the results.

- First the rotation is applied to align maps.
- Then the relative orientation between the two maps is determined by performing an iterative approach similar to Iterative Closest Point (ICP).
  - The point correspondence of the algorithm is established by comparing the norm.
  - Minimization of the Euclidian distance of the corespondent points is required.
  - There is no rotation involved, so this can be done by difference of the centroids along x and y:
    - $T = \begin{bmatrix} \delta_x \\ \delta_y \end{bmatrix} = \begin{bmatrix} \frac{1}{L-1}(\sum_{l=1}^{L} PT_{1x}[l] - \sum_{l=1}^{L} PT_{2x}[l]) \\ \frac{1}{L-1}(\sum_{l=1}^{L} PT_{1y}[l] - \sum_{l=1}^{L} PT_{2y}[l]) \end{bmatrix}$
    - PT_1 and PT_2: corespondent point sets from two maps.
    - L: the cardinality of the sets PT_1 and PT_2,
    - $PT_{1x}[l]$ and $PT_{1y}[l]$: correspond to the x and y components at location l of the array. (Similarly for PT_2.)
  - Once convergence happens, the algorithm stops.

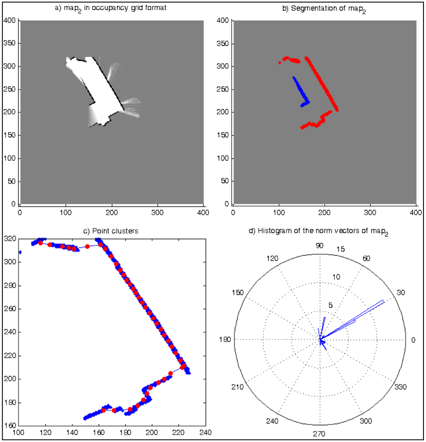
- CoroBots: Hokuyo UBG-05LN and Phidget Encoders



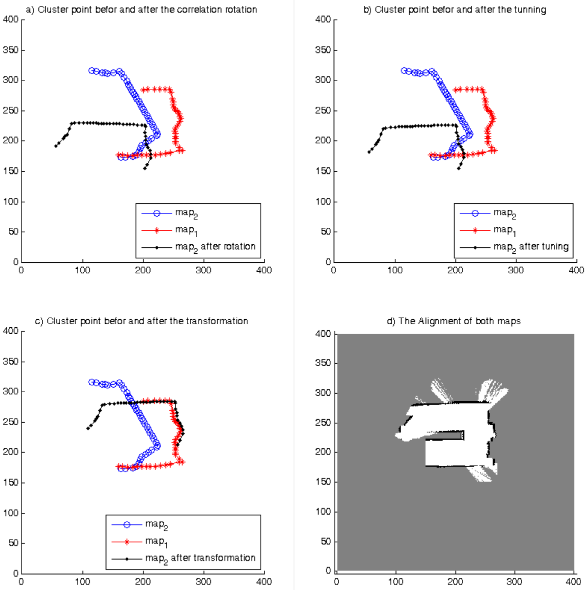
- Laser Odometry: Iterative Closest Point (ICP)
- Data Fusion: Extended Kalman Filter (EKF)

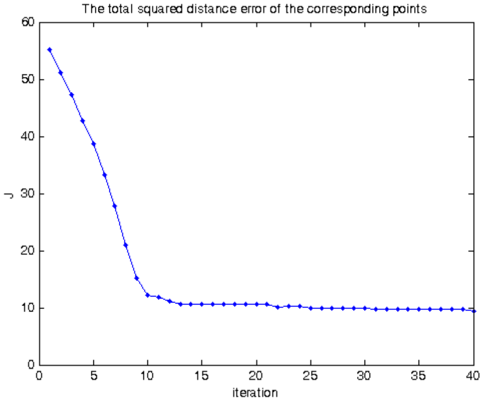


Robot 1

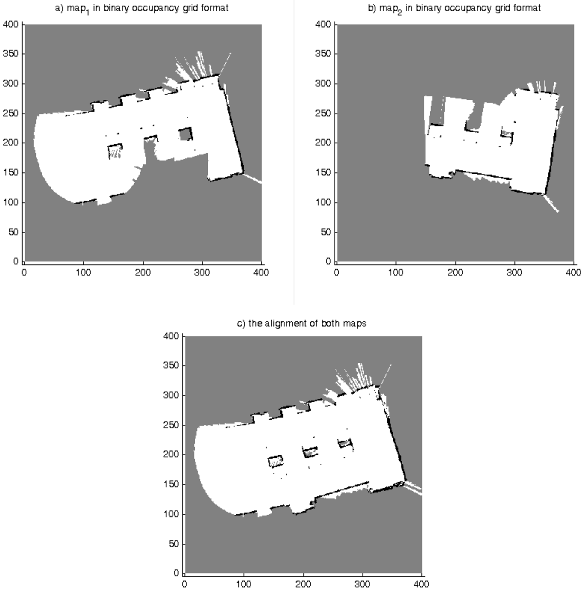


Robot 2


a) Cluster point befor and after the correlation rotation
map2
map1
map2 after rotation
b) Cluster point befor and after the tunning
map2
map1
map2 after tuning
c) Cluster point befor and after the transformation
map2
map1
map2 after transformation
d) The Alignment of both maps
0
50
100
150
200
250
300
350
400



- For 40 cluster points, the average time for training is about 20 seconds
- The result converges after 5 iterations.
- The processing time for random walk is more than 70 seconds.


a) map$_1$ in binary occupancy grid format
b) map$_2$ in binary occupancy grid format
c) the alignment of both maps
0
50
100
150
200
250
300
350
400

- Map fusion based on neural networks.
- Considerably fast.
- future work: developing adaptive methods to determine the number of the neurons (clusters).

Thank You.