Rubber Band Paradigm: A Method for Environment Reconstruction for Global Planning of a Mobile Robot

John Harvey and Shahram Payandeh
School of Engineering Science
Simon Fraser University
Burnaby B.C. Canada
V5A-1S6
harveyb@sfu.ca, shahram@cs.sfu.ca

Abstract

The Rubber Band Method (RBM) is a simple and effective method for storing sensor data collected by a mobile robot. A variety of useful information relevant to path planning and problem solving can be extracted. The RBM provides a frame work for evaluating the accuracy of newly acquired sensor data. This paper discusses the goals, the components, and ultimately the uses of the RBM.

1. Introduction

In this paper the control of a robot is broken into three distinct layers[13]. The lowest layer is designed to interface to the motion control (motors), and presents position information to the top layers[2,11]. The middle layer is the local obstacle avoidance, and data collection layer. This layer sends motion commands, such as move forward 1 meter, to the lowest layer. The top and final layer is the high level control. This layer contains the problem solving intelligence, and directs the middle layer to complete smaller sub tasks such as move to a goal point[4,7,10]. The Rubber Band Method (RBM) is designed to act as a middle layer in the control of a mobile robot.

The RBM is a middle layer that provides the movement directives needed for the lower control layer, and provides a foundation for higher level path planning and goal seeking. Sensor information is verified with previous knowledge, and an accurate model of the robots location is maintained. This layer does not control the motors directly, and does not do goal seeking.

2. Review of Middle Layer Architectures

There are several other successful middle layer designs. The first, and perhaps simplest, is an occupancy grid[13,17]. This algorithm has a certainty associated with each grid location in the "world". Occupancy grids are easy to add data to, and you can easily tell if the area directly in front of the robot is occupied. Occupancy grids generally take up a lot of memory, and have difficulty identifying error.

The field potential methods are less common, but present an interesting approach[14,16]. Instead of an occupancy grid, the field potential method stores how far away obstacles are, or stores how close a position is to the goal. This method is good for solving unstructured environments with free standing obstacles, but again requires a large amount of memory, and new data may be difficult to add.

3. The Rubber Band Method

The rubber band method represents the world as a single border between travel able space and obstacles. As the border is built it stretches out like a rubber band. Figure 1 shows a trivial example. As the robot explores new areas the rubber band expands to represent the new sensor data. Unfortunately, this intuitive idea is not easily implemented. The following sections describe the components of the rubber band method.

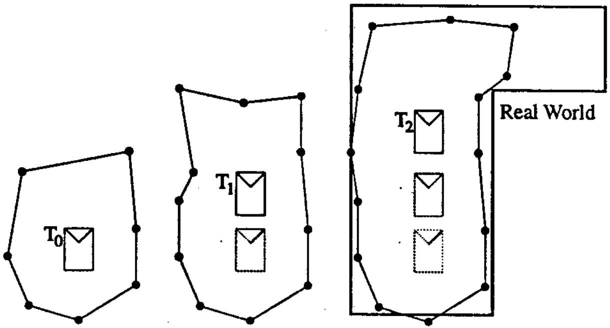


Figure 1, A robot with the rubber band representation superimposed at three consecutive times (T_0, T_1, T_2).

There are several components to the RBM. First, the new data points collected must be added to the existing data structure. Second, the location of the robot must be recalculated after each move, taking into account both the movement of the robot, and the positional error accumulated from previous moves. Third, any sensor errors that are introduced in the rubber band must be found, accounted for, and removed. Finally, the data must be processed into a more abstract form, such as topology, for use by the top layer.

The above tasks have several common characteristics. Each task has a minimum required execution time and frequency. Many tasks can get a more accurate solution if they are given more execution time beyond their minimum requirements. None of the tasks have a required execution order relative to each other. Using these criteria, we selected an appropriate control architecture.

The black board architecture adequately fills the needs of this system. Using the black board system we control how frequently tasks are executed.. Every task to the rubber band method must be aware of the computation constraints of an embedded system. These constraints may include a limited amount of memory, and limited processing power. We have broken the tasks into three broad categories. *Insertion* agents add new sensor data into the rubber band; *cleaning* agents remove redundant, and incorrect sensor data, and *reasoning* agents derive information from the rubber band.

3.1 Insertion of New Data

When new sensor data is collected, it must be inserted into the existing rubber band, or it must be discarded. There always exists error, regardless of the method of sensing the environment. Most error modes can be anticipated, and steps are taken to reduce their effect. Many experimental robots use some limited combinations of touch, time of flight ultrasound, intensity of return infrared, phase based laser and vision. In the rubber band method data insertion is abstracted into specific agents, each customized to the sensor.

There are several good papers and books on ultrasound data[1,6,8,9]. We do not intend to discuss the details of interpreting ultrasound data, but rather we will detail the steps involved with inserting new data.

When the rubber band starts it assumes it is located in a small box.[1] When new data is collected it either moves an existing point, or adds a new point. The actual implementation is broken into two broad heuristics: a) if the new sensor data is close to an existing point, the existing data point is moved; b) if the new sensor data is not close to any existing points, a new point is inserted into the rubber band. These ideas are illustrated in Figures 2 and 3.

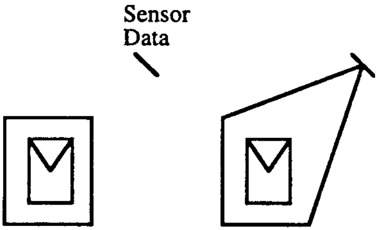


Figure 2, Adding a Data Point with Point Moving

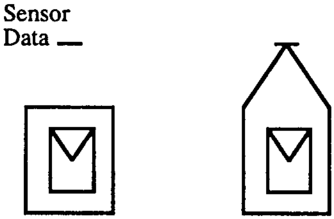


Figure 3, Adding a Data Point with Point Insertion

While scanning through the rubber band, several possible candidates for moving or adding data points may arise. Figure 4 shows a possible situation of point moving. The insertion agent must choose the best point among the candidates (circled) or be prepared to not add the new sensor data with the expectation further sensor data will be unambiguous. Several heuristics are available to determine the best point to move. The best strategy depends on the type of sensor being used.

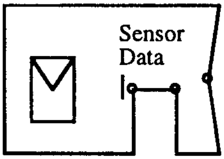


Figure 4, A set of points to consider

Thin walls, such as cubical partitions, represent some ambiguity in the rubber band representation. If the robot starts on one side of a wall and travels around to the other side of the wall, the side of the wall which a rubber band segment represents becomes ambiguous. To reduce possible erroneous expansion, the direction from which a point is observed should be noted. When a new point is added or moved, the "side" of the rubber band must be considered.

3.2 Map Cleaning

The insertion agent continuously adds data into the rubber band. Over time there will be a lot of data added to the band, some of which may be redundant, some of which may be incorrect. There are several different methods to clean the rubber band. Figure 5 summarizes the notation used in this section.

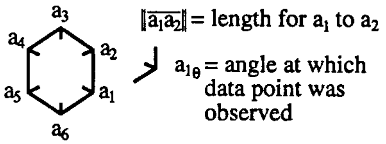


Figure 5, Notation Used For Rubber Bands

3.2.1 Wall Straightening: The first type of map cleaning agent is the straightener. The straightener takes several points (a_i to a_n) that fall on a best fitted line, and replaces these points with just the endpoints of the line. Figure 6 shows a simplified example of where a straighten would take place. Simple linear regression is a valid method for straightening, but it is quite computational demanding. Using small angle assumptions ($\sin(\theta) \approx \theta$, $\cos(\theta) \approx 1$) we can analyze a series of points against a fitting constant. If relationship 1 holds for a section of the rubber band, then a straighten should take place on this section. The fitting constant is proportional to the error of the fit (generally $K_{fit} < 1.1$)

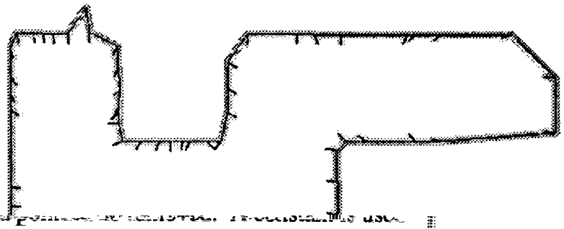
Figure 6, A Straightened Section

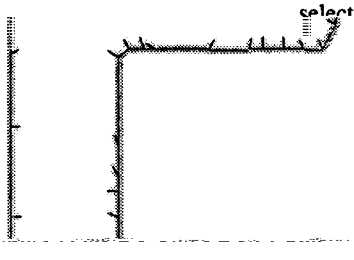
$$\left|\frac{\sum_{i=0}^{n}\left\|\overline{a_i a_{i+1}}\right\|}{\left\|\overline{a_i a_n}\right\|}\right| < K_{fit} \qquad [1]$$

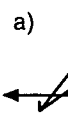
---

[1] "O God, I could be bounded in a nutshell and count myself a king of infinite space, were it not that I have had dreams"[12!]

3.2.2 Spike Removal: In the ideal world the insertion agents would never add incorrect data. When the insertion agent is limited finite amount of time, sometimes errors in the form of stray points are added. The spike removal agent removes these stray points. Not everything that appears to be a spike is. Figure 7 shows some instances for and against spike removal. As with the walls straightening agent, several different algorithms are available. Relationship 2 shows the selection criteria for a point to be removed. A constant is used in much greater than (>>) operation. This constant is generally greater than 2.







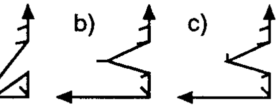
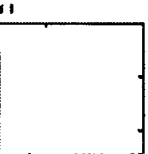
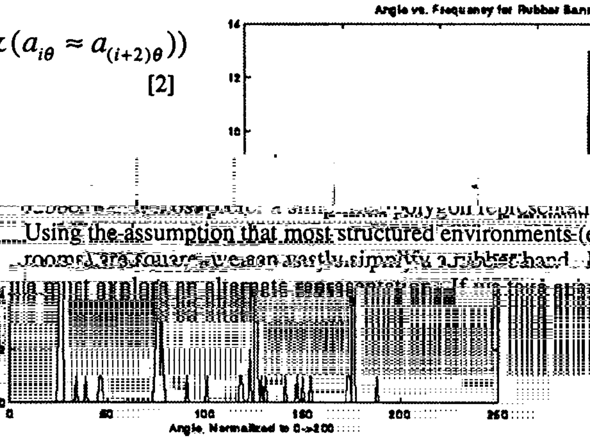


Figure 7. a,b Do Cut, c Don't Cut.

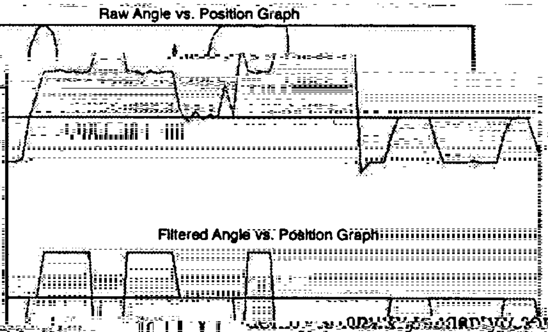
$$\forall(a_i, a_{i+1}, a_{i+2}) \Rightarrow$$
$$((\|\overline{a_i a_{i+1}}\| + \|\overline{a_{i+1} a_{i+2}}\| >> \|\overline{a_i a_{i+2}}\|) \& (a_{i\theta} \approx a_{(i+2)\theta})) \quad [2]$$





3.3 High Level Reasoning

Although some planning layers may be able to deal with a raw rubber band, most prefer a simplified polygon representation. Using the assumption that most structured environments (e.g. rooms) are square we can easily simplify a rubber band. First we must explore an alternate representation. If we take every interconnection and represent it as a unit vector we get a collection of vectors with direction. Plotting a histogram of direction the vectors point (0->360°) we can observe the relative frequency of each direction. This graph will have a peak for each of the basic directions, N, S, E, W. Figure 8b and 9b show two different rubber bands with their histograms.

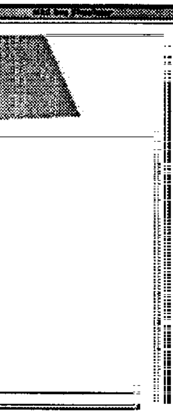


Once we have built the histogram of angles we try to best fit a four line into the histogram. Using these angles we build a graph of angle vs. position on the graph. Transitions on this graph represent corners in the rubber band. Connecting these corners builds a simplified version of the rubber band. Figure 8d and 9d show these graphs, and resulting simplified polygons in detail.

4. Correcting Error

Perhaps the most useful part of the rubber band method is it's ability to identify some types of motor error and compensate for it. Slippage in the motors introduces error in the presumed location of the robot. Over time this error grows to make previous data useless. Using the straight wall assumption, the rubber band can be "repaired" to compensate for some of this error.

Once a small section of the rubber band has been built the directions of the walls are set. After a new section of wall has been added it is un-skewed into its proper place. Because most of the damage done by motor error is due to slippage while turning, un-skewing reduces most of the error.



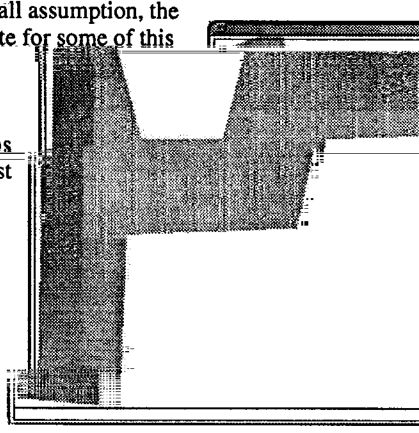


Figure 8, High Level Reasoning for an almost square rubber band.

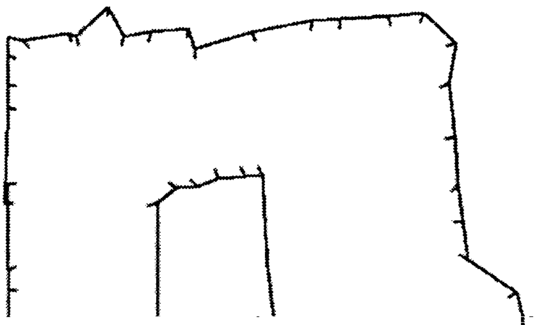
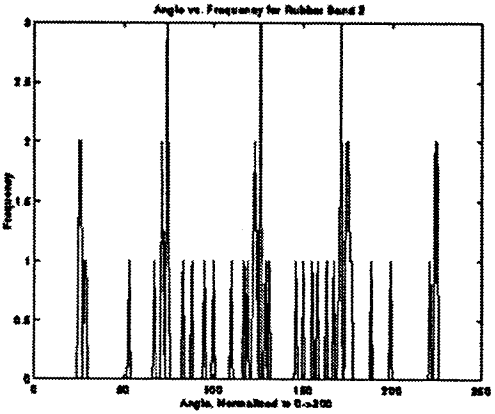
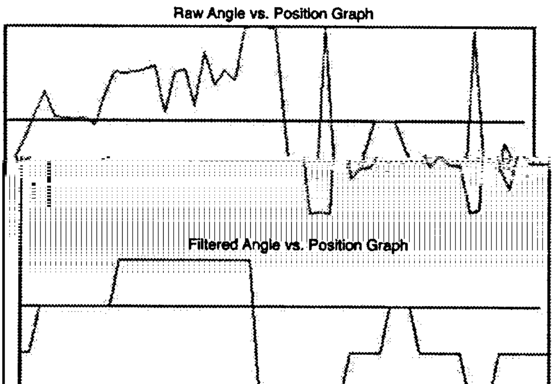


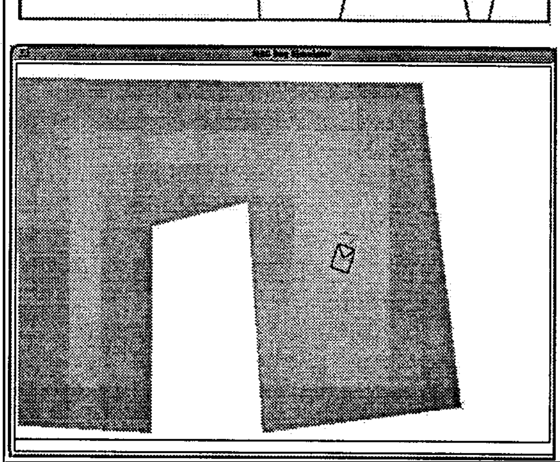
Figure 9, High Level Reasoning for an Less Than Square Rubber Band.

5. Conclusion and Future Work

The Rubber Band Method is still in its infancy. The RBM must be implemented on a real hardware platform to evaluate its true effectiveness. Free standing obstacles, or islands, have not been discussed. A rubber band to topology agent is required to use more traditional AI. The rubber band method has been shown to be a viable middle layer algorithm, and an intuitive means to store sensor data.







enstein J., Koren Y., "Histogramic In-Motion
for Mobil Robot Obstacle Avoidance", IEEE
ions on Robotics and Automation, 7, No 4, August
535.

oks, A. "A Robust Layered Control System For A
Robot", IEEE Journal of Robotics and Automation,
-2 March 1986, pp. 14

Censke J., Gini M., "Why Is It So Difficult For A
o Pass Through A Doorway Using UltraSonic
' Proceedings of 1994 IEEE International Conference
ics and Automation, 4, Pg 3124

x G., Heymann M., Bruckstein A., "Two-Dimensional
avigation Amoung Stationary Polygon Obstacles,"
ansactions on Robotics and Automation, 9, No 1, Feb
96.

zalez J., Ollero A., Reina A., "Map Building for a
Robot Equipped with a 2D Laser Range finder,"
ngs of 1994 IEEE International Conference on
and Automation, 3, Pg 1904

enstein A.A. , Badreddin E. , "Mobile-Robot Position
Using Ultrasonic Range Measurements," International
f Robotics and Automation, 9, No 2, pg 72.

T., Kahgn A., Robins G., "Optimal Robust Path
g in General Environments," IEEE Transactions on
s and Automation, Vol 9, No 4, December 1993, pg

onard J., Durrant-Whyte H., "Directed sonar sensing
bile robot navigation," Boston, Kluwer Academic
ers, 1992

onard J., Durrant-Whyte H., Cox I., "Dynamic Map
g for an Autonomous Mobile Robot", The International
of Robotics Research, Vol 11, No 4, August 1992, pg

Liu Y., Arimoto S., "Path Planning Using a Tangent
for Mobile Robots Among Polygonal and Curved
es," The International Journal of Robotics Research,
No 4, August 1992, pg 376.

Martinez A., Tunstel E., "Fuzzy Logic Based
on Avoidance for a Mobile Robot," Robotica, Vol 12,
g 521.

Shakespeare, W. "Hamlet, Prince of Denmark,"
of the Future (CDROM), 3rd Edition

[1] Bor
Mapping
Transact
1991, pg

[2] Bro
Mobile R
Vol. RA-

[3] Buc
Robot To
Sensors,'
on Robot

[4] Fou
Robot Na
IEEE Tra
1993, Pg

[5] Gon
Mobile R
Proceedi
Robotics

[6] Hol
Update U
Journal o

[7] Hu
Plannin
Robotic
775

[8] Le
for mob
Publish

[9] Le
Buildin
Journal
286.

[10]
Graph f
Obstacl
Vol 11,

[11]
Collisio
1994, p

[12]
Library

[13] Rødseth O., "A Software Model for Control of Autonomous Robots," International Journal of Robotics and Automation, Vol 5, No 1, 1990, pg 4

[14] Takeda H., Facchinetti C., Latombe J., "Planning the Motions of a Mobile Robot in a Sensory Uncertainty Field," IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol 16, No 10, October, 1994.

[15] Vacherand F., "Fast Local Path Planner in Certainty Grid," Proceedings of 1994 IEEE International Conference on Robotics and Automation, Vol 3, Pg 2132

[16] Wyard-Scott L., Meng Q., "A Potential Maze Solving Algorithm for a Micromouse Robot", Proceedings of IEEE Pacific Rim Conference on Communications, Computers, and Signal Processing 1995, 1, pg 612.

[17] Zelinsky A., "A Mobile Robot Exploration Algorithm", IEEE Transactions on Robotics and Automation, 8, No 6, December 1992, pg 707.