Evolving an environment model for robot localization

Marc Ebner

Eberhard-Karls-Universität Tübingen
Wilhelm-Schickard-Institut für Informatik
Arbeitsbereich Rechnerarchitektur
Köstlinstraße 6, 72074 Tübingen, Germany
ebner@informatik.uni-tuebingen.de

Abstract. The use of an evolutionary method for robot localization is explored. We use genetic programming to evolve an inverse function mapping sensor readings to robot locations. This inverse function is an internal model of the environment. The robot senses its environment using dense distance information which may be obtained from a laser range finder. Moments are calculated from the distance distribution. These moments are used as terminal symbols in the evolved function. Arithmetic, trigonometric functions and a conditional statement are used as primitive functions. Using this representation we evolved an inverse function to localize a robot in a simulated office environment. Finally, we analyze the accuracy of the resulting function.

1 Motivation

Robots are usually equipped with a variety of sensors that can be used to sense their environment. Information from these sensors can be used to create an internal model of the environment. Such models can be very helpful to the robot in moving from one place to another. Sometimes, however, it may be necessary that the robot relocalizes itself if it has lost track of its current position. This may happen if the robot performs a movement due to an external force that is not sensed by the internal sensors or if odometry errors accumulate. We are exploring the use of an evolutionary method to evolve a representation of the environment that can be used for robot localization. Before we describe our evolutionary approach to robot localization we give a brief review about existing approaches.

2 Background

A detailed overview about traditional methods for self localization of a mobile robot is given by Talluri and Aggarwal [16]. They classify the different methods into four categories: landmark-based methods, methods using trajectory integration and dead reckoning, methods using a standard reference pattern, and

methods using a priori knowledge of a world model which is matched to the sensor data. Often, the world model is constructed using sonar sensors, a laser range finder or from a set of images. Basic methods for map learning are reviewed by Thrun et al. [18]. The information from different types of sensors may be fused to create a three dimensional map of the environment [21].

Talluri and Aggarwal [14, 15, 17] used visual information to locate a mobile robot in an outdoor environment. Image features are used to search for the current position in a model of the environment. Burgard et al. [5] used position probability grids to estimate the absolute position of a mobile robot equipped with sonar sensors. Yamauchi and Beer [23] incrementally built a topological map of the environment using information from the robot's odometry. Kurz [11] constructed a topological map from ultrasonic range data using a self-organizing feature map. Von Wichert [19] and von Wichert and Tolle [20] developed a self-organizing visual environment representation for a mobile robot. Geometric moments calculated for image segments are used to relocate the robot inside a topological map created from an omnidirectional view of the environment. Crowley et al. [6] used principal components of range data for position estimation of a mobile robot. First, training scans are used to generate a lookup table mapping eigenvalues of the range data to possible robot positions. The acquired range data projected into eigenspace is used as an index into this table and a number of candidate poses are extracted. A Kalman filter is used to reject invalid hypotheses as the robot moves. Beetz et al. [3] also used an active method for relocalization for a mobile robot. Whenever the robot has lost track of its position it performs an action which will minimize the uncertainty taking a cost criteria to perform the action into account. For an experimental comparison of two localization methods, Markov localization and scan matching, see Gutmann et al. [7].

Balakrishnan and Honavar [1] followed a biologically motivated approach to spatial learning for robot localization. They developed a computational model of the hippocampus. Their simulated mobile robot incrementally constructs a neural model of the experienced environment. Sensor readings that have not been experienced previously are remembered together with position information obtained from dead reckoning. Later, if similar sensor readings are experienced, the position information is recalled and used to correct possible localization errors. Yamada [22] evolved behaviors for a mobile robot to recognize different types of environments.

Our approach to robot localization is completely different from traditional methods. We are trying to evolve a function that estimates the robots position given the current sensor readings. Thus the whole system is under evolutionary control. We try to evolve an internal representation of the environment. Nordin and Banzhaf [12, 13] previously evolved a world model using genetic programming with linear genotypes [2] for robot control. They evolved a function which estimates the expected fitness for possible motor commands given the current sensor readings. A planning process searches the space of possible motor commands using the evolved function to predict the expected fitness of the action.

The best action is executed on the real robot. The resulting actual fitness of this action is stored along with sensor readings and motor commands in a memory buffer. This memory buffer is used by a learning process to evolve the function to predict fitness values before the action is executed.

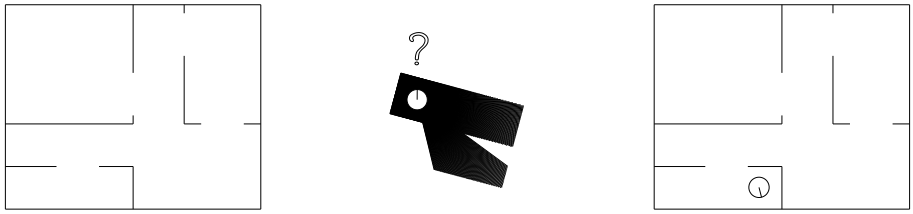

Fig. 1. Where in the map shown on the left is the robot located given the shown sensor readings? The correct location is shown on the right.

3 Evolving an environment model for robot localization

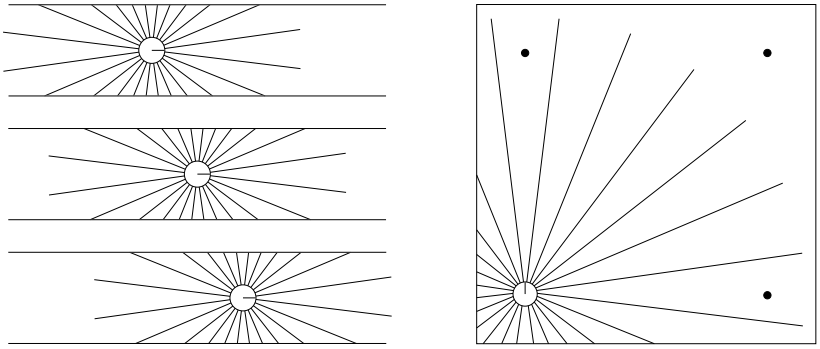

Fig. 2. Two examples for ambiguous sensor readings. If the robot travels through a long corridor it will get the same sensor readings for all points along a line parallel to the corridor [5]. If the robot is in a large room it will not be possible to tell in which corner the sensor readings were taken. In addition to the actual position of the robot three other positions are also possible.

In our experiments it is assumed that the robot perceives its environment using a laser range finder or a ring of sonar sensors. The environment is perceived like a floor plan. Robot simulators usually use a floor plan to calculate distance

information for the sensors. There exists a mapping M from the robot's position P to sensor values S.

$$M : P \xrightarrow{\text{World}} S$$

To localize a mobile robot using the sensor information we are trying to invert this mapping. That is, we are trying to find a mapping M^{-1} which estimates the robot position P for the current sensor values S (Figure 1).

$$M^{-1} : S \xrightarrow{\text{Model}} P$$

Of course, it is often not possible to find an inverse mapping as the examples in Figure 2 show. Without any point of reference this task cannot be solved. Points of reference that are usually available include doors, windows or objects which are distributed inside a room such as a desk and some chairs. The location of these objects can be used to disambiguate the sensor readings.

Traditional methods usually rely on some type of matching method to locate the robot inside the world model using the current sensor readings. Often these methods only locate the robot inside a model up to some prespecified grid resolution. Differently from these traditional methods, we are trying to evolve an inverse function which converts sensor readings into a map location. If it is indeed possible to find such a function it would be possible to continously localize the robot for all possible sensor readings. On a real mobile robot the method could be used to associate sensor readings with position information obtained using the robot's odometry. The evolved inverse function represents the robot's model of the world. It could later be used to relocate the robot if odometry information becomes inaccurate or is completely absent due to a system malfunction.

4 Symbolic regression using genetic programming

To evolve the inverse mapping from sensor readings to robot localizations we are using genetic programming [8, 10, 2]. Koza [9] has shown that genetic programming can effectively be used to search for a function described by a finite number of mappings. To apply genetic programming to the task of robot localization we first have to define the set of input variables.

The task of finding a function which maps raw range values to position in the environment is a very difficult task due to the number of variables. Therefore we preprocess the raw range values. This preprocessing has to have a number of requirements. Its major purpose is to perform a data reduction to reduce the size of the search space. Relevant features of the environment should be amplified and no valuable information should be lost. In addition, the reduced set of variables should vary smoothly with a change in position of the robot. Because we are trying to estimate the position (x and y coordinates) of the robot the resulting set of variables should be independent of the robot's orientation.

In our experiment we calculated the moments of the distribution of range values [4]. The following set of terminals was used.

- Moments $M_x = \frac{1}{n}\sum_{i=1}^{n} (\text{Range}(i))^x$ (x={1,2,3,4}),
- Central moments $CM_x = \frac{1}{n}\sum_{i=1}^{n} (\text{Range}(i) - M_1)^x$ (x={2,3,4,5}),
- Moments of first derivative $DM_x = \frac{1}{n}\sum_{i=1}^{n} (\Delta\text{Range}(i))^x$ (x = {2,3,4,5}),
- and the ephemeral random constants RAND (range [0,1)), RAND10 (range [0,10)), and RAND100 (range [0,100)).

The following elementary functions were used:

- Arithmetic functions: +,-,*,/ where the division operation evaluates to 1 if the absolute value of the divisor is less than 10^{-10}.
- Trigonometric functions: `SIN`, `COS`, `TAN`, `ASIN`, `ACOS`, `ATAN`, `ATAN2`,
- and the conditional statement `IFLTE`.

where

$$\texttt{ASIN}(x) = \begin{cases} \frac{\pi}{2} & x > 1 \\ -\frac{\pi}{2} & x < -1 \\ \text{asin}(x) & \text{otherwise} \end{cases}, \texttt{ACOS}(x) = \begin{cases} 0 & x > 1 \\ \pi & x < -1 \\ \text{acos}(x) & \text{otherwise} \end{cases},$$

$$\texttt{TAN}(x) = \begin{cases} 0 & |\cos(x)| < 10^{-10} \\ \frac{\sin(x)}{\cos(x)} & \text{otherwise} \end{cases}.$$

Provided that the sensors are placed dense enough the resulting moments are nearly independent of the orientation of the robot. For our experiments we have used 720 sensors. This is a typical number of sensor readings available from a standard laser range finder.

5 Experiments

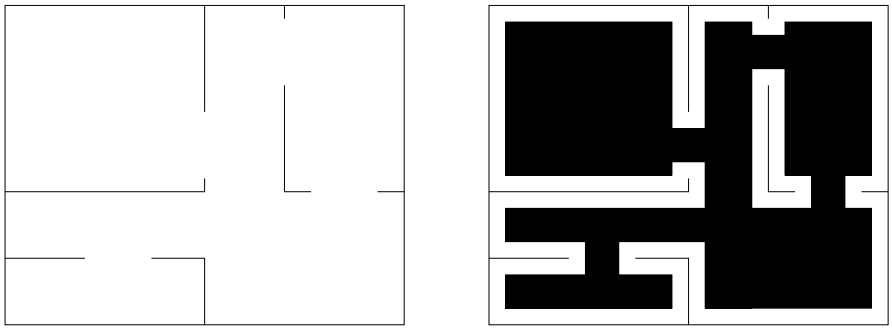

Fig. 3. Environment used during the experiments (left). Black areas show possible positions for fitness cases (right).

For our experiments we have used the simulated environment shown in Figure 3. The environment shows a floor plan of a section from a hypothetical office

building with size 7.5m × 6m. The task is to evolve a representation of the environment using 1000 fitness cases (associations between sensor readings and robot location) which are distributed over the environment. The fitness cases are randomly distributed over the black areas shown in Figure 3. We evolved one individual for the x coordinate and another individual for the y coordinate. The error of the individual which tries to recall the x coordinate is calculated according to

$$\text{error} = \frac{1}{n}\sum_{i=1}^{n}\left(x_{\text{actual}} - x_{\text{estimated}}\right)^2 \tag{1}$$

where x_{actual} is the actual x coordinate of the robot, $x_{\text{estimated}}$ is the value estimated by the evolved individual and n is the number of fitness cases. The estimated position is truncated if it falls outside an area of a size twice as large as the original map. The error of the individual which calculates the y coordinate is calculated in the same way. The adjusted fitness of each individual is calculated as $\text{fitness}_{\text{adj}} = \frac{1}{1+\text{error}}$.

We performed five runs for each coordinate using different random seeds to initialize the first generation. A population size of 1000 individuals was used. Tournament selection with size 7 was used and crossover, reproduction and mu-

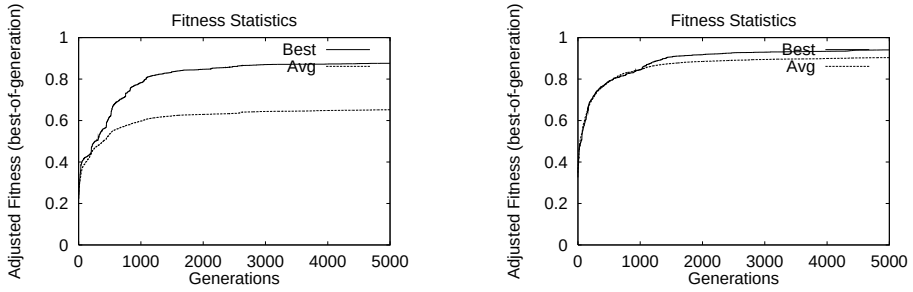

Fig. 4. Adjusted fitness for all generations averaged over 5 runs. The graph on the left shows the statistics for the x coordinate, the one on the right shows the statistics for the y coordinate.

tation probabilities were set to 85%, 10% and 5% respectively. The run was aborted after 5000 generations. Figure 4 shows the resulting fitness statistics averaged over five runs for the two coordinates. The task of evolving an inverse function for the y coordinate seems to be easier than the task for the x coordinate. The best individuals of the five runs were combined to form a single individual consisting of two trees. This individual was tested on 10 additional random fitness cases. The result of this test is shown in Figure 5. The robot is shown in the current location and the estimated position is marked with a small circle. The estimated location is very accurate for most locations but also

differs by a large amount for some locations. To locate points where the error of localization was correct up to a certain error we used an additional 1000 fitness cases. Figure 6 shows locations where position estimation was accurate up to 1m (left) and locations where position estimation differed by more than 1m (right).

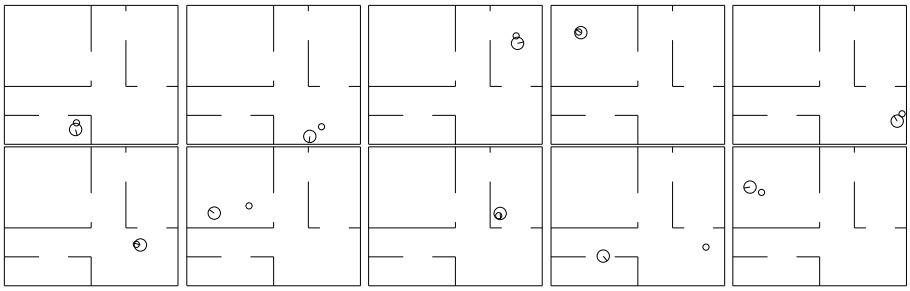

Fig. 5. Generalization results for 10 additional fitness cases. The robot is shown in the correct position. The estimated position is marked with a small circle.

6 Conclusion and directions for further research

Our experiments have shown that it is possible to evolve an internal world model for robot localization using genetic programming. To reduce the size of the search space moments of the distribution of sensor readings are calculated. The moments are used to amplify relevant features of the environment while maintaining as much information as possible. Using the genetic programming paradigm with the moments as terminals and arithmetic functions, trigonometric functions and a conditional statement as elementary functions we evolved a single function which approximates the inverse mapping from sensor readings to robot positions.

The estimated position could be filtered as the robot moves to improve the accuracy of the method. Also, other input representations might be more suitable for the task of robot localization. These could also be placed under evolutionary control.

7 Acknowledgements

This work was supported in part by a scholarship according to the Landesgraduiertenförderungsgesetz. For our experiments we used the lil-gp Programming System [24].

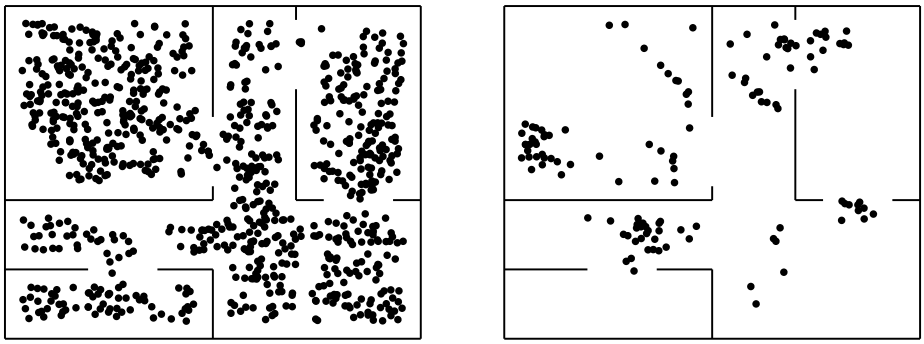

Fig. 6. 1000 additional fitness cases were used to locate points where position estimation was correct up to an error of 1m (left) and where position estimation differed by more than 1m (right).

References

1. K. Balakrishnan and V. Honavar. Spatial learning for robot localization. In J. R. Koza, K. Deb, M. Dorigo, D. B. Fogel, M. Garzon, H. Iba, and R. L. Riolo, editors, *Genetic Programming 1997: Proceedings of the Second International Conference on Genetic Programming, 1997*, pages 389–397. Morgan Kaufmann, 1997.
2. W. Banzhaf, P. Nordin, R. E. Keller, and F. D. Francone. *Genetic Programming - An Introduction: On The Automatic Evolution of Computer Programs and Its Applications*. Morgan Kaufmann Publishers, San Francisco, California, 1998.
3. M. Beetz, W. Burgard, D. Fox, and A. B. Cremers. Integrating active localization into high-level robot control systems. *Robotics and Autonomous Systems*, 23:205–220, 1998.
4. I. N. Bronštein und K. A. Semendjajew. *Taschenbuch der Mathematik*. Verlag Harri Deutsch, Thun und Frankfurt/Main, 24th edition, 1989.
5. W. Burgard, D. Fox, D. Hennig, and T. Schmidt. Estimating the absolute position of a mobile robot using position probability grids. In *Proceedings of the 14th National Conference on Artificial Intelligence*, pages 896–901. AAAI Press/MIT Press, 1996.
6. J. L. Crowley, F. Wallner, and B. Schiele. Position estimation using principal components of range data. *Robotics and Autonomous Systems*, 23:267–276, 1998.
7. J.-S. Gutmann, W. Burgard, D. Fox, and K. Konolige. An experimental comparison of localization methods. In *International Conference on Intelligent Robots and Systems, Victoria, B.C.*, October 1998.
8. J. R. Koza. *Genetic Programming, On the Programming of Computers by Means of Natural Selection*. The MIT Press, Cambridge, Massachusetts, 1992.
9. J. R. Koza. *Symbolic Regression - Error-Driven Evolution*. In J. R. Koza. *Genetic Programming I: On the Programming of Computers by Means of Natural Selection*, pages 237–288. The MIT Press, Cambridge, Massachusetts, 1992.
10. J. R. Koza. *Genetic Programming II, Automatic Discovery of Reusable Programs*. The MIT Press, Cambridge, Massachusetts, 1994.
11. A. Kurz. Constructing maps for mobile robot navigation based on ultrasonic range data. *IEEE Transactions on Systems, Man, and Cybernetics – Part B: Cybernetics*, 26(2):233–242, April 1996.

12. P. Nordin and W. Banzhaf. Real time evolution of behavior and a world model for a miniature robot using genetic programming. Technical Report SysReport 5/95, Department of Computer Science, University of Dortmund, 44221 Dortmund, Germany, 1995.
13. P. Nordin and W. Banzhaf. Real time control of a Khepera robot using genetic programming. *Cybernetics and Control*, 1997.
14. R. Talluri and J. K. Aggarwal. Position estimation for an autonomous mobile robot in an outdoor environment. *IEEE Transactions on Robotics and Automation*, 8(5):573–584, October 1992.
15. R. Talluri and J. K. Aggarwal. Image/map correspondence for mobile robot self-location using computer graphics. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15(6):597–601, June 1993.
16. R. Talluri and J. K. Aggarwal. Position estimation techniques for an autonomous mobile robot – a review. In C. H. Chen, L. F. Pau, and P. S. P. Wang, editors, *Handbook of Pattern Recognition and Computer Vision*, chapter 4.4, pages 769–801. World Scientific Publishing Company, 1993.
17. R. Talluri and J. K. Aggarwal. Mobile robot self-location using model-image feature correspondence. *IEEE Transactions on Robotics and Automation*, 12(1):63–77, February 1996.
18. S. Thrun, A. Brücken, W. Burgard, D. Fox, T. Fröhlinghaus, D. Hennig, T. Hofmann, M. Krell, and T. Schmidt. Map learning and high-speed navigation in RHINO. In D. Kortenkamp, R. P. Bonasso, and R. Murphy, editors, *Artificial Intelligence and Mobile Robots: Case Studies of Successful Robot Systems*, pages 21–52, Menlo Park, California, 1998. AAAI Press/The MIT Press.
19. G. von Wichert. Vismob: Aufbau und Nutzung selbstorganisierender, bildbasierter Umweltrepräsentationen für mobile Roboter. In P. Levi, T. Bräunl, and N. Oswald, editors, *Autonome Mobile Systeme 1997*, pages 84–94, Berlin, 1997. Springer-Verlag.
20. G. von Wichert and H. Tolle. Towards constructing and using selforganizing visual environment representations for mobile robots. In M. Á. Salichs and A. Halme, editors, *3rd IFAC Symposium on Intelligent Autonomous Vehicles, March 25-27, 1998, Madrid, Spain*, pages 712–717, 1998.
21. P. Weckesser and R. Dillmann. Modeling unknown environments with a mobile robot. *Robotics and Autonomous Systems*, 23:293–300, 1998.
22. S. Yamada. Learning behaviors for environment modeling by genetic algorithm. In P. Husbands and J.-A. Meyer, editors, *Proceedings of the First European Workshop on Evolutionary Robotics, Paris, April 1998*, pages 179–191, 1998.
23. B. Yamauchi and R. Beer. Spatial learning for navigation in dynamic environments. *IEEE Transactions on Systems, Man, and Cybernetics – Part B: Cybernetics*, 26(3):496–505, June 1996.
24. D. Zongker and B. Punch. *lil-gp 1.01 User's Manual (support and enhancements Bill Rand)*. Michigan State University, March 1996.