Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 35 (2014) 653 – 661

Procedia Computer Science

18th International Conference on Knowledge-Based and Intelligent Information & Engineering Systems - KES2014

Viability of decisional DNA in robotics

Carl Sheffer, Cesar Sanin*, Edward Szczerbicki

The University of Newcastle, Callaghan, NSW 2308, Australia

Abstract

The Decisional DNA is an artificial intelligence system that uses prior experiences to shape future decisions. Decisional DNA is written in the Set Of Experience Knowledge Structure (SOEKS) and is capable of capturing and reusing a broad range of data. Decisional DNA has been implemented in several fields including Alzheimer's diagnosis, geothermal energy and smart TV. Decisional DNA is well suited to use in robotics due to the large amount of data available and the generally repetitive nature of the tasks robots perform. However, there is very little evidence about the system's performance in this application. This project aims to assess the viability of SOEKS in robotics. Several knowledge representation approaches were explored then coded in the Java programming language. A hardware platform was constructed from readily available electronics and set up to be compatible with the Java language. Codes were installed on the hardware platform and tested by conducting a series of feature mapping tasks. Success of this project could lead to the future use of SOEKS in robot control.

Peer-review under responsibility of KES International.

Keywords: SOEKS; Decisional DNA; Mapping; Knowledge Representation; Robotics

1. Introduction

The proliferation of computer systems into greater parts of industry and domestic life, combined with the expansion of cloud systems, means that every day, vast quantities of data are not only gathered, stored and

* Cesar Sanin. Tel.: +612-4921-7465; fax: +612-4921-7465.
E-mail address: cesar.sanin@newcastle.edu.au.

Peer-review under responsibility of KES International.
doi:10.1016/j.procs.2014.08.147

processed but can also potentially be shared and re-used. This data revolution has the potential to automate human decision making, similar to the way machines automated human labour during the industrial revolution.

The sharing of knowledge between computers is critical to computers supplanting human decision makers. For example, Google's self-driven car needs knowledge from Google Maps and Google Street View to drive.

The Set Of Experience Knowledge Structure (SOEKS) could facilitate this sharing of knowledge. SOEKS is a knowledge representation structure that uses prior experiences to shape future decisions. It has been implemented in many fields, from data mining and augmented reality to Alzheimer's diagnosis and Smart TV.

While the SOEKS has been shown to be shareable and capable of use on a wide variety of platforms, there is very little evidence about its performance in robotics, which - as far as replacing humans is concerned – is arguably the most critical field because it is the one concerned with interacting with the physical world.

This project aims to assess the viability of SOEKS in robotics. SOEKS is implemented on a basic robotic car and the car is tested on a coloured map. Since the best approach to knowledge representation within SOEKS could not be known ahead of time, several knowledge representation approaches were explored. Testing was conducted by performing a series of feature mapping tasks.

Success of this project could lead to the future use of SOEKS in robot control, which would allow the broad range of data sharable in SOEKS to be used to enhance the performance of robots.

2. Background

A wide variety of AI techniques have been applied to robotics in attempts to facilitate learning and sharing of experience.

For example, the Agilo robot team at Munich University of Technology used neural network techniques to train their robots for the 2001 Soccer World Championship[1] and Stulp, Fedrizzi, Zacharias, Tenorth, Bandouch and Beetz used experience based learning to train a humanoid robot to manoeuvre and lift a cup off a bench[2]. Experience based learning was also used in navigational path planning by Goel, Donnellan, Vazquez and Callantine[3].

All these AI systems had difficulty transferring the experience they gained to other platforms or to other robots due to the way the experience was stored. Finding a method to store knowledge in a way that can be more widely used and understood is a problem that has been tackled by researchers in the knowledge representation field. Borgida, Brachman, McGuinness, Resnick developed the 'Classic' data model[4] to allow the storage of incomplete information. Their program used classes as the building blocks of the system. The classes had any number of properties associated with them. The database allowed querying of the attributes belonging to each class and common attributes across classes. Storage and querying were able to be conducted efficiently as the logical functions available to the database had been reduced to a minimum. The database only had basic commands such as 'and', 'at-least' and 'one-of' but not 'or' which limited the database but allowed the database to work more efficiently.

The JTP architecture by contrast represented knowledge by forming logical rules[5]. Assumptions, premises and rules were developed by the system to handle information. The system's main advantage was its ability to develop 'proofs' that could shortcut large sequences of logic, saving time during execution. The JTP system had greater computational efficiency than Classic but could not deal with incomplete information as gracefully[5]. However both these solutions have characteristics that limit their use in the wide-ranging knowledge sharing that may be needed in the future.

The Set of Experience Knowledge Structure (SOEKS) is a domain-independent standardised knowledge structure[6] created by Sanin and Szczerbicki between 2004 and 2007 and is being expanded by the Knowledge Engineering Research Team (KERT) at the University of Newcastle.

Domain independence allows experience stored in SOEKS to be transported across platforms, unlike most knowledge representations languages, which are specialised to their individual use. SOEKS was developed to capture formal decision events and store them as explicit knowledge, so they can later be used, analysed and categorised[6].

Knowledge is stored through a combination of four basic components; variables, rules, functions and constraints. Variables represent the environment using an attribute-value language and form the basis of the SOEKS. Functions

describe the relationship between a group of independent variables and a dependent variable. Constraints place limits on variables. Finally, rules are a way of expressing relationships between actions and the conditions under which they should be performed[7].

The combination of four basic elements to create complex, adaptable forms closely resembles the structure of DNA. This gives rise to the Decisional DNA analogy. Just as the four building blocks of DNA (G, A, T and C nucleotides) combine to form chromosomes which then generate proteins, the four building blocks of Decisional DNA (variables, functions, constraints and rules) combine to form decisional chromosomes which then generate decisions. This is illustrated in figure 1 below.

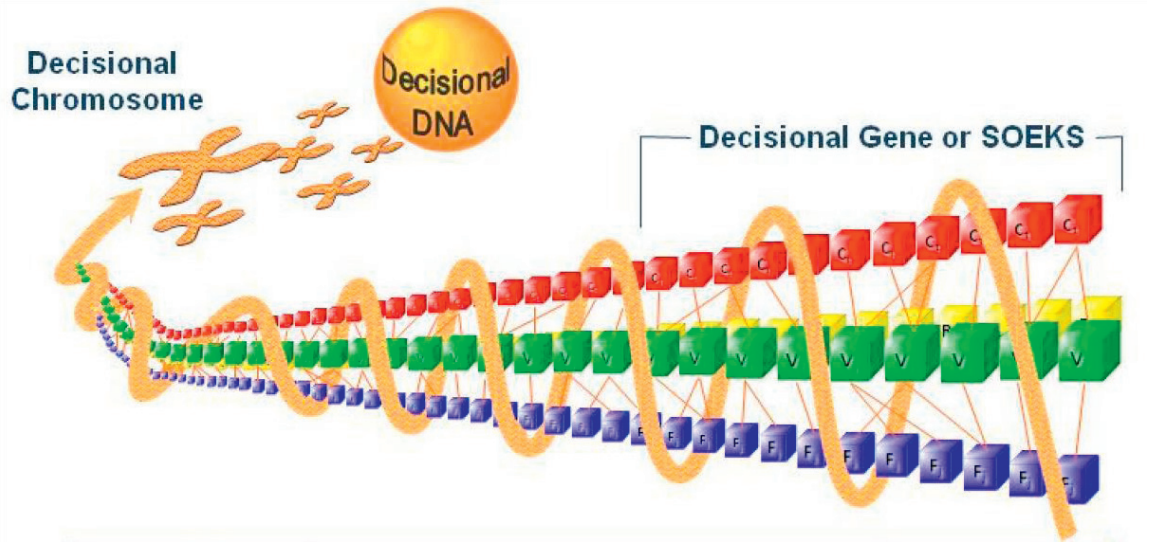


Fig. 1. SOEKS DNA analogy[6].

SOEKS is represented in Extensible Mark-up Language (XML) which makes it transportable and shareable across many platforms and applications. The Java programming language is used as an interface between the XML repository and the human programmer[6].

3. Methodology

3.1. Aim

This paper demonstrates the viability of SOEKS in robotics and assesses the suitability of three knowledge representation approaches.

This paper builds on the work of Zhang et al[8] by tackling the knowledge representation problem directly, by capturing a greater range of environmental features and storing features more explicitly. Since experience is central to the SOEKS system, it was important the system captured the greatest possible amount of experience in order to demonstrate the viability of SOEKS in robotics.

SOEKS already operates in many areas and is able to transfer experience between those areas. Adding another field to the SOEKS domain will increase the functionality of SOEKS and pave the way for SOEKS to be used as a robot control system, allowing robotic systems to one day benefit from the experience captured by SOEKS in other fields.

3.2. Experimental design

To demonstrate the viability of SOEKS in robotics, the core functions of SOEKS had to be tested. SOEKS had to perform its core functions of storing and re-using experience gathered during operation on a robotics platform. A test was devised to determine whether experience has been successfully reused.

3.3. Experimental setup

A white map with a black line and several coloured spots was printed. The robotic CarBot was given the operation of driving along the black line and gaining 'experience' of the location of the coloured dots. Successful re-use of the experience was tested by instructing the CarBot to drive directly to a coloured dot.

3.4. Knowledge representation approaches

Within this experimental method, there were many ways to represent the variables associated with landmarks. Three different approaches were implemented and tested. These were: the vector, Cartesian and combination approaches.

- Vector: The robot's direction, expressed as an angle and a length was combined to create a vector that represented its movement. The combination of these vectors would then be used to represent the position of objects. This would allow easy navigation between objects, through simple vector addition. This approach involved very simple calculations so put little strain on the robot's computational facilities and also allowed easy addition of paths.
- Cartesian: The robot's location was given in x and y coordinates (with the starting position as the origin). Each feature was stored with x and y coordinates. The CarBot needed to constantly calculate its x and y position, which could put strain on the very limited processor in the CarBot. The Cartesian system was an explicit system, so could be interpreted by other robots or by human users. Once objects' locations were stored, calculating paths to and between them was easy. This approach allowed the simplest representation of object location.
- Combination: A hybrid method that used the best of the Cartesian and vector approaches. The robot updated its position in Cartesian coordinates but still stored the start and end of straight line segments, like the vector method. This approach stored locations explicitly while also storing the path taken.

3.5. Testing apparatus

Four pieces of equipment were required:

- Map: The map contained a series of coloured landmarks, a black line for the CarBot to follow and a marked starting square to allow the CarBot to be placed in the same position for each test. The map was printed on A0 (841 × 1189 mm) poster card.
- Hardware Platform (CarBot): The CarBot was based on the work of Zhang[8] with several improvements made. It was constructed using the Lego Mindstorms NXT 2.0 system, with two motors connected directly to two wheels and the colour sensor attached to the front of the CarBot, 80mm in front of the wheel axles. All motors and sensors were attached to the NXT brick, which had LeJOS –an open source firmware- burned onto it. LeJOS allowed Java script to control the Mindstorms hardware point.
- PC Platform: During the experiment, the CarBot was in constant Bluetooth contact with a PC. Java programming was conducted in the Eclipse program. Eclipse also managed connection to the CarBot and data retrieval.
- Java Code: The Java code intermediated between the CarBot hardware (running LeJOS firmware) and the SOEKS system. Three variants of the code were required to test the Cartesian, vector and combination approaches. First, for the Cartesian, the CarBot followed the black line using a basic line follower script, updating (but not storing) its position and direction, based on readings from the inbuilt tachometers. When a landmark was detected, the x and y coordinates were saved. For the vector approach, the CarBot followed the black line and stored the length and angle of its movement at every point the CarBot turned. When a landmark was detected, all vectors leading up to that detection were stored in a group. And thirdly, for the combination approach, the

location was updated as for Cartesian but when a landmark was detected, the coordinates of every turning point were stored. The code also controlled the re-use of data and the control group sequence.

3.6. Experiment

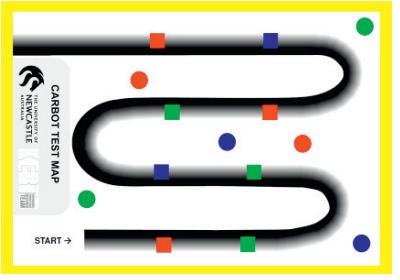
The CarBot was placed in the starting square and the script (vector, Cartesian or combination) was called. After the CarBot had reached the end of the line the script was ended. For the DriveTo stage, the CarBot was placed in the starting square then instructed to drive to a specified landmark (determined by averaging coordinates from the first stage). In total, there were 120 Cartesian runs, 120 combination runs, 120 vector runs and 30 control runs. The control group consisted of a script that searched the map for a desired colour in the most efficient way possible without 'experience'. The script instructed the CarBot to drive back and forth across the map, detecting one strip of the map at a time. The CarBot was able to detect the edge of the map because the border was coloured yellow. Once a coloured spot was detected, the time taken to drive to that spot was stored.

4. Results

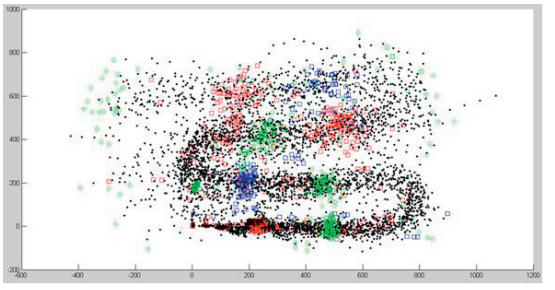
4.1. Summary

A total of 11,181 'sets of experience' were captured during the experiment. A table summarising the data is shown below.

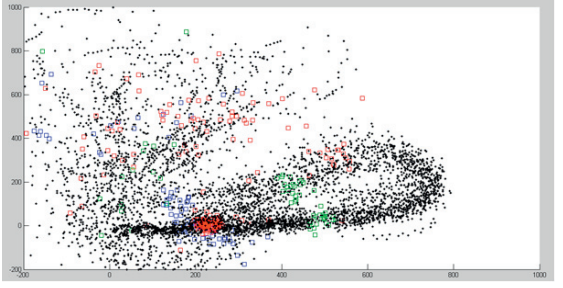
Table 1. Comparison of knowledge representation methods



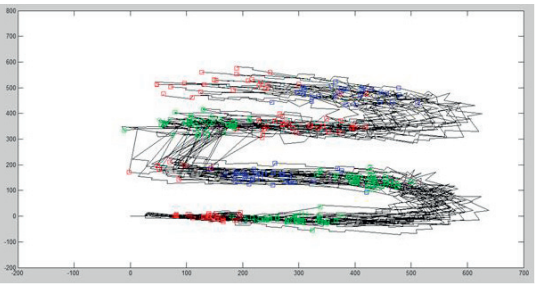
Actual Map



Cartesian Map



Combination Map



Vector Map

4.2. Accuracy

Accuracy was defined as the distance from the actual landmark to the point predicted by the SOEKS system after gaining experience. The "predicted point" was calculated by averaging the data points for each landmark. "positional error" is the distance between the predicted point and the actual point. The average error was 40.6mm for cartesian and 137.4mm for vector.

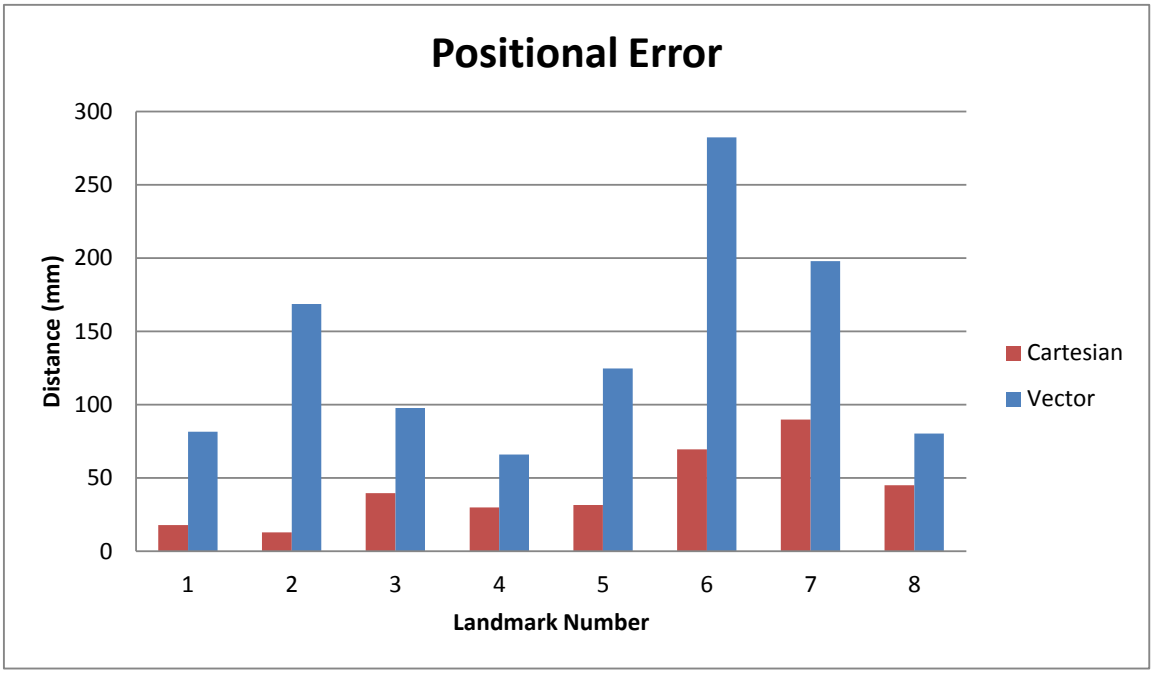


Fig. 2. Comparison of Cartesian and Vector accuracy.

4.3. Completeness

A metric was created to assess the percentage of the map correctly stored by SOEKS. Completeness was calculated by comparing SOEKS's "best guess" of the colour in each square to the colour actually present in each square. The total completeness for cartesian was 79.25%. Completeness for vector was 77.52%.

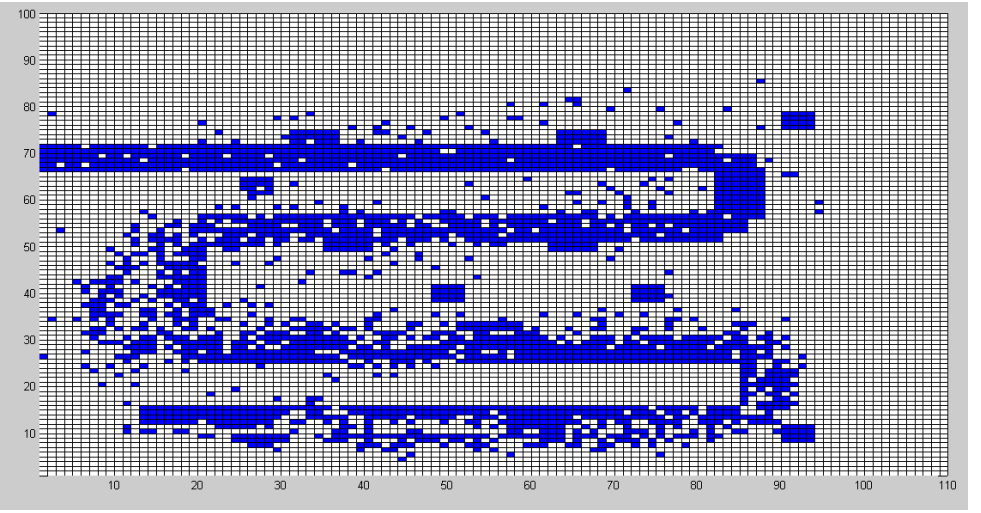


Fig. 3. Completeness of Cartesian. White squares are 'complete' blue squares are 'incomplete'

4.4. Comparison to control group

The average time taken for the CarBot to find a particular landmark for the control was 94.4 seconds, with a standard deviation of 68.1s. The average time in the SOEKS was 3.43s, with a standard deviation of 1.17. There were 1.34 standard deviations between SOEKS and the control. The time taken for the CarBot to find a particular landmark was significantly lower with SOEKS than in the control.

4.5. Causes of inaccuracy

There were multiple sources of inaccuracy, including sensor noise, sensor bias, battery level variation, backlash in motor gears, ambient light variations, slipping of tyres, coding method, the limited cycle time and data sharing between PC and CarBot.

4.6. Variation between methods

Since the same calibration procedures were followed for each method and almost identical procedures were followed for finding landmark location, it is perhaps surprising that there was any difference between the methods at all. However, there were significant differences in both the magnitude and causes of error between the different approaches.

The combination method gave the worst set of results, which was surprising because it used exactly the same localisation method as the Cartesian method, which gave the best set of results. The only difference between the methods was that the Cartesian method saved only the current coordinates when it found a landmark, whereas the combination method saved the current coordinates as well as every coordinate that led to it. This increased the time required to save. Since the CarBot processor was occupied during the saving process, an increased saving time caused less fine approximations to curved paths.

The vector method, unlike the other methods, did not save explicit coordinates. It saved a series of vector angles and lengths that lead to a landmark. This constant saving was far less onerous than the combination but was still significant. This is reflected in the vector results being more accurate than the Cartesian results but less accurate than the combination results.

The Cartesian was the most accurate method. This was due to the much faster saving of landmark variables than either of the other methods. When the Cartesian method saved a landmark, only four numbers were stored. For vector and combination, anywhere between 10 and 200 numbers were stored. It was the fact that all computational resources could be dedicated to updating the location that gave Cartesian the greatest accuracy

4.7. Comparison to other methods

4.7.1. Zhang et al (2010) CarBot

Zhang's CarBot reduced the average time taken to find a dot by 50%[8]. The CarBot in this project reduced drive times from 94.4s to 3.43s, a reduction of 96.4%, which was a much greater improvement.

4.7.2. LeJOS SLAM

Oliveira et al used hardware identical to that used in this project to perform a mapping task[9]. The fact that identical hardware but different coding was used provides a great opportunity for comparison. Unfortunately, the authors did not provide a value for completeness or any numerical accuracy values. However, the output of the two can be compared visually in figures 4 and 5 below.

Fig. 4. Map (a) against reality (b) for LeJOS SLAM[9].

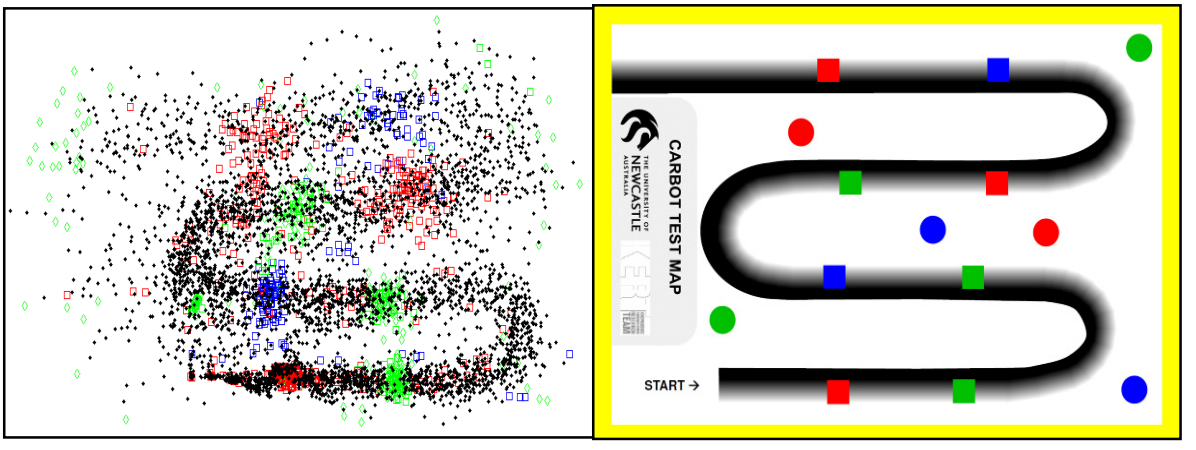


Fig. 5. Map against reality for CarBot

While nothing concrete can be said, the generally 'scruffy' nature of the outputs can be seen in the figures above. In both cases, this was caused by noisy sensors and limited computational power.

5. Conclusions

The implementation of SOEKS onto the CarBot significantly improved the time taken to find a landmark. This demonstrated that SOEKS is a viable architecture for robot control.

Due to its higher accuracy and simplicity, the Cartesian method is the recommended method for robot control going forward. The vector method was deemed to be workable and could be used, dependent on the future application.

The effect of saving times on accuracy was significant, as seen in the variation between vector, Cartesian and combination. In future applications of SOEKS in robotics, only the absolute minimum of data should be saved.

6. Recommendations for future work

The following areas of work are suggested to increase the accuracy or functionality of SOEKS in robotics.

- One of the many data mining and analysis tools that that operate within SOEKS could be applied to the current system. This would allow SLAM-like functionality such as changing old points when new data is gathered. Functions could be found that combine similar readings into a single point and eliminate outliers.
- The experiment could be re-run with the addition of an external compass to the Lego Mindstorms system. This would improve accuracy, especially given current issues with the facing angle drifting away from the correct value over time.
- The entire experiment could be repeated on more accurate hardware. The Lego system has many merits but was not highly precise. New, accurate equipment would show the potential of SOEKS on commercial equipment.

Acknowledgements

Thanks go to all members of the Knowledge Engineering Research Team (KERT); Edward Szczerbicki, Cesar Sanin, Peng Wang and Imran Khan for their advice and guidance. Particular thanks to the former KERT member Haoxi Zhang whose work formed the basis for this project. The help he provided during the early stages of this project was invaluable.

References

1. Beetz, M., Buck, S., Hanek, R., Schmitt, T., & Radig, B. (2002, July). The AGILO autonomous robot soccer team: Computational principles, experiences, and perspectives. In Proceedings of the first international joint conference on Autonomous agents and multiagent systems: part 2 (pp. 805-812). ACM.
2. Stulp, F., Fedrizzi, A., Zacharias, F., Tenorth, M., Bandouch, J., & Beetz, M. (2009, December). Combining analysis, imitation, and experience-based learning to acquire a concept of reachability in robot mobile manipulation. In Humanoid Robots, 2009. Humanoids 2009. 9th IEEE-RAS International Conference on (pp. 161-167). IEEE.
3. Goel, A., Donnellan, M., Vazquez, N., & Callantine, T. (1993, May). An integrated experience-based approach to navigational path planning for autonomous mobile robots. In Robotics and Automation, 1993. Proceedings., 1993 IEEE International Conference on (pp. 818-825). IEEE.
4. Borgida, A., Brachman, R. J., McGuinness, D. L., & Resnick, L. A. (1989, June). CLASSIC: A structural data model for objects. In ACM Sigmod record (Vol. 18, No. 2, pp. 58-67). ACM.
5. Fikes, R., Jenkins, J., & Frank, G. (2003, July). JTP: A system architecture and component library for hybrid reasoning. In Proceedings of the Seventh World Multiconference on Systemics, Cybernetics, and Informatics (pp. 27-30).
6. Sanin, C., & Szczerbicki, E. (2007). Towards the construction of decisional DNA: A set of experience knowledge structure java class within an ontology system. Cybernetics and Systems: An International Journal, 38(8), 859-878.
7. Sanin, C., & Szczerbicki, E. (2009). Experience-based knowledge representation: SOEKS. Cybernetics and Systems: An International Journal, 40(2), 99-122.
8. Zhang, H., Sanin, C., & Szczerbicki, E. (2010). Decisional DNA applied to robotics. In Knowledge-Based and Intelligent Information and Engineering Systems (pp. 563-570). Springer Berlin Heidelberg.
9. Oliveira, G., Silva, R., Lira, T., & Reis, L. P. (2009). Environment mapping using the lego mindstorms nxt and lejos nxj. http://paginas. fe. up. pt/~ ei04085/psite/LejosPaper. pdf, Acessado em, 20(01), 2013.