Literature Review

Submitted in partial fulfilment
of the requirements of the degree of

Bachelor of Science (Honours)

of Rhodes University

Luke Ross

Grahamstown, South Africa
May 28, 2012

0.1 Introduction

Fiducial marker navigation on mobile robots has been achieved successfully by researchers, for example, in monitoring underwater domains using an amphibious robot [21]. But, many implementations of these kinds of systems do not optimize the paths that the mobile robots travel. In most robot navigation systems in which minimum paths are calculated whilst avoiding obstacles, target tracking is not done using fiducial markers; instead methods based on GPS coordinates are used.

This chapter reviews literature on the WiFiBoT Lab mobile robot, fiducial markers, the toolkits available for fiducial marker detection and various search algorithms that can be used for robot path planning. Current systems that implement fiducial markers for navigational purposes as well as systems that implement efficient path planning are also discussed.

The chosen toolkit for marker detection needs to be reasonably accurate and, owing to the processing and memory constraints on many mobile robots, not too computationally demanding. In addition, the chosen search algorithm needs to have the following properties:

- It must be able to calculate the optimal path from the robot to the marker, avoiding obstacles.
- It must be able to function effectively in known, partially known and unknown environments.
- It must be able to function effectively with a moving target.
- It must be sufficiently computationally efficient for use on mobile robots.

0.2 WiFiBoT Lab

Wifibot Lab is a mobile robot intended for educational purposes as well as the development of affordable robotic systems[1]. Processing is done on the robot using an Intel Atom D510 processor[2]. Since this high-end mobile robot is being used for experimentation in

[1] http://www.wifibot.com/wifibot-wifibotlab.html
[2] http://ark.intel.com/products/43098/Intel-Atom-Processor-D510-(1M-Cache-1_66-GHz)

this project, more common, lower specification robots must be considered when choosing between the different libraries and algorithms available. WiFiBoT has been used successfully in various systems in the past, two of which are mentioned. The first is the PotPet robot [10]. This robot is a mobile flowerpot that allows users to grow plants in a more effective and enjoyable way. It is built using a WiFiBoT platform owing to its good mobility. PotPet is autonomous and automatically moves into areas with more sunlight as well as towards people when it requires more water. Demetriou et al. [5] describe the second system, which allows for mobile robot localization using WiFi signal strength measurements from a number of access points. This system provides a low-cost localization method for robotic applications in indoor environments, where GPS is unavailable. WiFiBoT was the robot chosen for testing the system.

0.3 Fiducial Marker Detection

Fiducial Markers

Fiducial, or fiduciary markers are image like objects, which are designed to be detectable and which usually contain an interpretable meaning [34]. Fiducials are used in many fields, including augmented reality, robotics and medicine. In robot navigation systems, an autonomous robot can follow a set path with the aid of fiducial markers [16]. Such markers offer performance, identification and localization improvements, and are more cost-effective when compared to other techniques used for path-planning in unknown environments [16]. In the medical field, fiducial markers are used when treating prostate cancer. During the treatment, doctors need to have an accurate view of the prostate gland and since this view is often not sufficient, gold fiducial markers are placed in the prostate to enhance vision[3].

Fiducial markers can vary from small dots to complicated bar-code images and can be of various shapes [18]. According to Owen et al. [18] and verified by Li et al. [15], two-dimensional, bar-coded, square fiducial markers are the best and most popular form of marker to use. Since these are the types of markers used for augmented reality or robot navigation systems [8], they seem appropriate. Examples of these types of markers are ARToolKit or ArUco markers. ARToolKit and ArUco markers consist of a black border and a black and white, uniquely patterned interior. The border of such a marker aids in the initial detection of the marker, whilst the interior pattern is used for identification of

[3]http://clinicaltrials.gov/ct2/show/NCT00061347

the marker [18]. The pattern must be unique [18] and as stated above, may have meaning encoded into its graphical representation. Technology such as a robot with an attached camera and running the required software can easily interpret such a pattern. Owing to the square nature of the marker, its position and orientation can be calculated accurately with respect to a calibrated camera [18].

Important Design Criteria

Owen et al. [18] state that, in order for fiducial markers to be most accurately recognized, various factors should be considered when designing the markers. Firstly, the shape of the fiducial should emit at least four points. The simplest of these shapes is a square and due to its simplicity, computational advantages are noted. In addition, the fiducial image can contain most colours, but it is best if a monochrome colour is used. This is because a monochrome image is easier to recognize against bright, contrasting backgrounds. Using a monochrome colour also gives computational advantages as the algorithms are simplified. Thirdly, the size of the marker is dependent on the resolution of the camera being used, with the border of the marker constituting at least 13% of the marker's width. Other important and more technical criteria concerning effective marker design, according to Owen et al. [18] are as follows:

- There should be no ambiguity when determining the marker's position and orientation relative to a camera.
- Markers should not favour certain orientations.
- If multiple markers are used in the system, they must all be unique.
- Simple algorithms that are not intensive should be used to locate and identify the marker quickly. For these algorithms to be used, the marker itself needs to be designed with this intent in mind.

Libraries used for Detecting Fiducial Markers

Libraries that can be used for marker detection include ARTag[4], ARToolKit[5] and ArUco[6]. According to ARTag's homepage, it is currently unavailable as a research aid and therefore is not discussed further for use in this project.

[4]http://www.artag.net/index.html
[5]http://www.hitl.washington.edu/artoolkit/
[6]http://www.uco.es/investiga/grupos/ava/node/26

0.3.1 ARToolKit

ARToolkit was first released in 1999 by Dr Hirokazo Kato, after which its further development was maintained by the Human Interface Technology Laboratory [1].

ARToolKit is a successful, robust marker based system, commonly used in the augmented reality field [7]. It is C and C++ language oriented, although Java and Matlab are also supported. ARToolKit can be used to detect pre-programmed ARToolKit markers in images and augment three-dimensional virtual objects onto these markers [1]. However, the marker detection functionality of the software can be used on its own without implementing any of the augmented reality functionality. Once a marker is identified, ARToolKit returns the marker ID as well as the location of the four corners of the marker relative to the camera device [6]. ARToolKit markers are two-dimensional and planar, and consist of a unique, black and white, patterned interior which is surrounded by a black border [7]. The large contrast between the black and white colours on the markers aids in ARToolKit's robustness [7].

Advantages

Two of ARToolKit's advantages with greatest relevance to this project are that it is open source for non-commercial use and is widely used [6]. Since it is widely used, there is extensive research material available. Furthermore, according to the feature list on the ARToolKit home page [1], ARToolKit supports multiple platforms including Windows, Linux, Mac OS X and SGI; it has real-time detection for two-dimensional markers; and there is sufficient documentation. The documentation available includes installation instructions, a simple calibration procedure as well as executable sample code. ARToolKit has precise detection, is easy to use and is well suited for the implementation of low-cost tracking systems, as the only added hardware requirement is a camera [2].

Disadvantages

Hornecker and Psik [9] point out some disadvantages of using ARToolKit, the first of which being that it does not recognize a marker if the full marker is not in the camera's view. Secondly, owing to its large black border, when the marker is printed on certain laser printers, light may be reflected causing inconsistencies in the video feed. The lighting issue can be resolved by printing the markers on an ink-jet printer [9]. Moreover, since ARToolKit is threshold based [8], a chosen threshold may not detect markers in different lighting conditions. Hirzer [8] further claims that ARToolKit has a high false positive rate.

History of Use

ARToolKit has been used extensively in marker tracking and augmented reality and, according to the "Projects" tab on the ARToolKit homepage, is used globally by over 300 researchers for a wide variety of purposes. Past applications that use ARToolKit vary from creating virtual environments in which ancient artifacts can be restored and viewed [22], to tracking human body movements [20].

0.3.2 ArUco - Based on OpenCv

The search for literature on ArUco yielded few results, perhaps owing to its fairly recent release. As a result, a large section of the literature presented below was found in online documents such as the ArUco homepage[7].

ArUco was developed by Rafael Munoz-Salinas from the University of Cordoba and released in November 2010 under a BSD license[8]. It is a basic C++ library used for the detection of fiducial markers and intended for augmented reality purposes [3]. It is based on OpenCv (Open Source Computer Vision), a vision based library, which is the most popular library in the computer vision field [3][4].

Features

According to ArUco's homepage, some of the features that ArUco offers are listed below:

- Markers can be detected using one line of C++ code.
- Use can be made of AR boards (a grid of markers) to increase detection accuracy.
- There are 1024 different ArUco markers available.
- Since it is based on OpenCv, ArUco detects markers quickly and reliably and is cross-platform.
- Examples and sample code are available.
- It is low-cost owing to possession of a BSD license.

[7] http://www.uco.es/investiga/grupos/ava/node/26
[8] http://softwaredd.net/softs/linux/games/aruco.html

ArUco also provides various applications with the library, two of which are relevant to this project [3]. The first application creates a marker, given an identification number, and saves this as a jpg file for printing. The second application, which is the main application, detects markers in a live video feed or pre-recorded video. Camera calibration is also possible using OpenCv.

History of Use

Speers et al. [21] reported using an amphibious, autonomous robot to monitor underwater sensors. In this system, fiducial markers are displayed on sessile sensors and are used to communicate with the robot using the ArUco library. ArUco has also been used in various unpublished projects, some of which are listed on the ArUco homepage. These projects include: the Soldamatic Project[9], the purpose of which is to aid in the training of welders by using augmented reality to create welding simulations, as well as OpenSpace3D[10], which is used for developing interactive, real-time 3D projects.

0.4 Search Algorithms

Requirements for Project

The robot in this project needs to be able to travel towards the fiducial marker in an environment that may contain obstacles. Since the robot should travel along the shortest path, a suitable search algorithm needs to be implemented such that the optimal path, avoiding obstacles, from the robot's current position to the marker is chosen. Furthermore, an efficient search algorithm needs to be implemented to keep computation to a minimum. Incremental search algorithms, some of which are presented below, are search algorithms that use previous search information when searching for minimal traversals and therefore can perform faster than searching from scratch [29]. These algorithms are popular for robot path planning[11].

Most research literature concerning path planning for mobile robots does not take navigation in unknown environments into account but rather, it is assumed that the entire map is initially known [24]. In this project, the robot will be placed in an environment which may be partially known or completely unknown as well and will therefore need to

[9] http://seabery.es/simulador-de-soldadura-soldamatic/?lang=en
[10] http://www.openspace3d.com/
[11] http://www-scf.usc.edu/ xiaoxuns/research.html

acquire map information using its lidar sensor. An algorithm that functions in unknown environments is therefore required. Since the marker will be moving whilst the robot attempts to travel towards it, the moving target search problem needs to be taken into account as well. The moving target search problem is a path planning problem, often encountered in computer games, where a hunter attempts to catch a moving target [29].

Representing the Environment

A specific robot configuration can be represented using a state, and the distance between two states can be represented using an arc. Therefore, a graph of these states, joined together by arcs, can represent a robot's environment. The search algorithms discussed use this representation [29][25]. The robot's environment, or map information, needs to be stored internally in a structure on the robot. This will allow the robot to update its map information whenever it is acquired from its sensors, plan an optimal path and know where to move. Two common structures that can be used to store map information are metric and topological maps.

Metric maps are usually implemented using occupancy grids. According to Thrun and Bucken [32], in the area of mobile robotics, occupancy grids are the most successful environmental representations. This is verified by Kraetzschmar et al. [13]. Although two-dimensional occupancy grids are most common, three-dimensional grids have also been implemented [30]. These grids represent the robot's environment using a matrix of cells, each of which contains an occupancy value [30][32]. This occupancy value is the probability that the cell is occupied [32]. Kuffner [14] states that many systems use grid-based maps to internally represent a robot's environment and then search for optimal paths from start to goal states using the grid's embedded graph. Occupancy maps are easily implemented and simple to use [13][31]. There is also no ambiguity in determining the robots position in its grid from its actual position in the environment [31]. Occupancy maps have the disadvatage of generating maps that are often inaccurate and do not represent sections of the environment correctly [30]. These inconsistencies generally occur in busy environments, unlike the environment used in this project. Another disadvantage of using grid-based maps such as occupancy grids is that, when a high resolution grid is required, memory requirements can become an issue [19][13].

Topological maps represent the environment using a graph [12]. This graph consists of a collection of nodes which are connected by arcs. The nodes correspond to distinctive landmarks and the arcs correspond to the distances between these landmarks. Unlike oc-

cupancy grids, the resolution of a topological map is purely dependent on the complexity of the environment. This allows for faster planning and less memory requirements than when using an occupancy grid [31]. Topological maps have disadvantages in that they are difficult to implement and maintain in large evironments and that similar landmarks are often ambigouosly recognized, resulting in inacurate map information [31][32].

Both grid-based and topological maps have their unique advantages and disadvantages. Therefore, researchers have integrated the two types of maps by generating topological maps on top of grid-based maps [32][19]. Firstly, the grid-based map is partitioned into smaller regions. These regions are seperated by criticial lines. An algorithm is then used to map this partitioned map into a topological graph with each region corresponding to a node and each critical line corresponding to an arc. This hybrid allows for accurate map representations as well as efficient planning. [32]

History of Use

Search algorithms have been used extensively in the fields of Computer Science and Robotics in a wide variety of applications. Examples are: computer games [17] such as computer chess[12], Google's famous search engine[13], and robot path planning. Further applications are discussed below.

Principal Search Algorithms

Although some other search algorithms are mentioned, the main algorithms that I have investigated for use in the robot's navigation are A*, the D* family of algorithms, Generalized Fringe-Retrieving A*, and Moving Target D* Lite.

0.4.1 A*

The A* search algorithm, which is an extension of Dijkstra's algorithm [17], is one of the most widely used search algorithms in the field of Artificial Intelligence [29], and forms the basis for the other search algorithms discussed in this section. It was first introduced by Hart, Nilsson and Raphael in 1968 [17].

A* performs faster than Dijkstra's algorithm by using heuristics [17]. It is a best-first

[12]http://verhelst.home.xs4all.nl/chess/search.html
[13]http://www.techi.com/2012/03/googles-search-algorithm-changes-1998-2012/

search algorithm and is used to calculate a minimal path from a start to a goal state [17]. A* calculates paths most accurately when all the map information is known. When not all the map information is known, D* is recommended[14].

According to Sun et al. [29], the moving target search problem can be solved using A* in a dynamic environment to calculate the path with the lowest cost between the current start and goal states whenever a change in the environment or a deviation of the goal state from the current path occurs. This is computationally expensive as the new path has to be planned from scratch each time a change occurs [29].

0.4.2 D* (Dynamic A*)

D* refers to any of the three incremental search algorithms: Original D*, Focused D* and D* Lite [17]. Stentz first described the original D* algorithm in 1994 [24] and further developed it in 1995, giving rise to the Focused D* algorithm [23]. D* (Dynamic A*) is a generalization of A* for partially unknown, completely unknown or dynamic environments [24][23]. D* Lite differs algorithmically from the other D* algorithms, but is used to solve the same path-planning problems [17][11]. D* Lite was introduced by Koenig and Likhachev in 2002 and is based on the authors' Lifelong Planning A* (LPA*) [11].

Algorithm Description

All three of the D* algorithms can be used to solve the path-planning problem in which a robot traverses from a start to a goal state in an unknown environment [17]. Since this is possible in an unknown environment, the three algorithms are also successful in this task in partially unknown and known environments. Nosrati et al. [17] gives a brief overview of how these algorithms function.
Initially, assumptions regarding the unknown map information are made and the minimal path from start state to goal state is calculated. The robot then traverses the planned path until a discrepancy in the map information is found. This new map information (a previously unknown obstacle, for example) is then added to the robot's map information. If this new information conflicts with the current path, a new minimal path from the robot's current position to the goal state is planned. This is repeated until either the goal state is reached, resulting in success, or no path can be calculated to the goal state, resulting in failure.

[14]http://theory.stanford.edu/ amitp/GameProgramming/Variations.html

Map Strategies for Unknown Terrain

When calculating the lowest cost path from the start state to the goal state using partial map information, three map strategies can be used to estimate the cost values of these unknown cells [25]. These strategies are known as the optimistic, pessimistic and average values strategies. For the optimistic strategy, unknown sections of the map are assumed to be easily traversable areas. For the pessimistic strategy, these sections are assumed to be the hardest areas to traverse, yet are still traversable. For the average value strategy, unknown cells are assumed to be similar to the known cells in their surrounding area and so an average of these cost values is calculated. [25]

Comparison of Original D*, Focused D* and D* Lite

If D* is implemented in an unknown environment, replanning occurs often. As discussed in the A* review, this replanning calculation can be computationally expensive. D* solves the computational issue that algorithms such as A* suffer from by using incremental graph theory techniques [25]. Stentz [25] states that, in environments with many states, D* can perform hundreds of times faster than brute force replanning algorithms such as repeated A* searches and, as stated in [17], provided that the goal state is static, the D* algorithms all outperform repeated A* searches. Stentz [25] further states that D* can guide a robot in real-time through unknown and dynamic environments. Compared to the original D*, Focused D* reduces computation by only updating states that are still relevant to the robots traversal using a heuristic [17]. The Focused D* algorithm is algorithmically complicated and is therefore difficult to understand and implement [11][26].

Unlike, Focused D*, D* Lite is simple to understand and therefore easily analysable and extendable [17][11]. Koenig and Likhachev [11] state that its efficiency is greater than or equal to that of Focused D*'s. This is confirmed by Nosrati et al. [17]. Although, researchers have extended D* Lite in a straightforward manner in order to solve moving target search problems in changing environments, it is slow [29].

History of Use

The D* algorithm is viewed as a landmark in the area of mobile robot navigation [26] and has been implemented extensively on mobile robots [11]. Two of these implementations include indoor Nomad robots and outdoor High Mobility Multipurpose Wheeled Vehicles (HMMWVs) [11]. It seems that current systems generally implement D* Lite rather

than the older D* algorithms. Even researchers in Stentz's lab, from where the D* algorithm originated, now often use D* Lite instead of D* for their current projects [33]. On Koenig's research page[15], some applications of D* Lite are listed. These include the prototype system tested on the Mars rovers "Spirit" and "Opportunity", in which elements of D* Lite were used. Another was Carnegie Mellon University's winning DARPA Urban Challenge vehicle, where elements of D* Lite were used for maneuvers such as navigating through parking lots and complicated U-turns.

0.4.3 Generalized Fringe-Retrieving A*

Generalized Fringe-Retrieving A* (G-FRA*) was introduced in 2010 by Xiaoxun Sun, William Yeoh and Sven Koenig [28]. G-FRA* is an incremental search algorithm, which is a generalization of Fringe-Retrieving A* (FRA*). It allows for moving target searches to be solved on arbitrary graphs instead of using two-dimensional gridworlds, which are not realistic when working with robotic systems. This algorithm was designed for moving target search problems in static, known environments. The hunter traverses along the optimal path from its current state to the current state of the target and whenever the target deviates from the path, a new optimal path is calculated. This new path is calculated using A*, but instead of replanning from scratch, previous search information is used. This process is repeated until the hunter catches the target. [29][28]

According to Sun et al. [28], FRA* is the fastest algorithm for moving target search problems using two-dimension grids and, according to experimentation, G-FRA* proved to be the fastest algorithm for moving target search problems using arbitrary graphs. It performed better than Generalized Adaptive A* (GAA*) which was the former fastest algorithm for the task using arbitrary graphs, by up to one order of magnitude. These experiments were performed on known state lattices, used for UGV navigation [28]. Although GAA* was tested in a static known environment [28], it can also be used in dynamic environments [27].

0.4.4 Moving Target D* Lite

The search for literature on this algorithm yielded few results, perhaps owing to its recent publication. Moving Target D* Lite (MT-D* Lite) as the name suggests, is an algorithm

[15] http://idm-lab.org/research.html

used for solving moving target search problems [29]. Specifically, it is an extension of D* Lite which makes use of the principle behind G-FRA* for recalculation of optimal traversals from the hunter to the target in dynamic environments [29]. MT-D* Lite is a recent algorithm introduced by Sun, Yeoh and Koenig in 2010 [29].

Sun et al. [29] state that D* Lite is not suitable for moving target search problems as it performs slowly. This is because it shifts the map in order to maintain a stationary start state. Much of the information obtained from previous searches is not reusable owing to this shift and therefore D* Lite can perform slower than letting A* search from scratch [29]. Sun et al. [29] go on to state that many other incremental search algorithms including D*, FRA* and G-FRA*, are not suitable for solving moving target search problems in dynamic environments. This is because these algorithms were designed for use in static environments or systems in which the start state is kept stationary [29]. These performance issues led to the development of the two incremental search algorithms, Basic MT-D* Lite and MT-D* Lite, which are both extensions of D* Lite, but do not have to transform the map and solve moving target search problems in dynamic environments [29]. Under experimentation, MT-D* Lite performed four to five times faster than Generalized Adaptive A* (GAA*) [27], the former fastest incremental search algorithm for solving the moving target search problem in dynamic environments [29]. GAA* was also introduced by Sun, Yeoh and Koenig [27].

Basic MT-D* Lite and MT-D* Lite Performance

For moving target search problems in dynamic environments, D* Lite can be used by calculating the optimal path from the hunter to the target whenever a change in the environment is observed or the target deviates from the known path. Basic MT-D* Lite increases the performance of this algorithm by calling its BasicDeletion() method instead of problematically shifting the map [29]. MT-D* Lite is an optimization of Basic MT-D* Lite which calls an optimized method, OptimizedDeletion(), instead of BasicDeletion(), and therefore, also does not require shifting of the map [29]. Pseudocode can be found in [29] for both the Basic MT-D* Lite and the MT-D* Lite algorithms. MT-D* Lite achieves improved performance compared to Basic MT-D* Lite by using the principle behind G-FRA* [29]. Under test conditions, Basic MT-D* Lite and MT-D* Lite performed better than GAA* in both static and dynamic environments. This is due to the fact that Basic MT-D* Lite and MT-D* Lite reuse previous search information whereas GAA* searches from scratch [29]. MT-D* Lite also performed better than Basic MT-D* Lite in both static and dynamic environments [29]. These experiments were performed in both

static and dynamic, known environments.

The search for literature on the performance of MT-D* Lite in partially unknown and completely unknown environments yielded no results.

0.5 Conclusion

The discussed literature indicates that systems with similar purposes to this project's do exist, but are implemented differently. Although some literature could not be found, it seems plausible that, using the toolkits and algorithms discussed in this chapter, the proposed robot navigation system can be implemented on mobile robots successfully, meeting all the project requirements. By implementing this system using a slightly different approach, this project provides an opportunity to add to the areas of robot navigation using fiducial markers and robotic path planning.

Bibliography

[1] ARToolKit. ONLINE. Available from: `http://www.hitl.washington.edu/artoolkit/`.

[2] Artoolkit - based tracking: A new device in VRPN. Tech. rep., Universidad de los Andes, 2007.

[3] Brunner, J. ArUco: Augmented Reality library from the University of Cordoba. ONLINE. Available from: `http://www.ros.org/doc/api/aruco_pose/html/index.html`.

[4] Coombs, J., and Prabhu, R. OpenCV on TI's DSP+ARM platforms: Mitigating the challenges of porting OpenCV to embedded platforms. White Paper - Texas Instruments, 2011.

[5] Demetriou, G., Vystavkin, A., Anastasiou, S., Theofilou, A., Giannopoulos, C., and Yerolemou, D. Indoor mobile robot localization using a wireless network: Wifibot case study. In *IC-AI* (2008), H. R. Arabnia and Y. Mun, Eds., CSREA Press, pp. 668–674.

[6] Fiala, M. ARToolKit applied to panoramic vision for robotic navigation. In *Proceedings of the Vision Interface, Halifax, Nova Scotia, Canada, June 11-13, 2003* (2003).

[7] Fiala, M. ARTag, an improved marker system based on ARToolKit. Tech. rep., National Research Council of Canada, July 2004.

[8] Hirzer, M. Marker detection for augmented reality applications. *Image, Rochester, NY* (2008).

[9] Hornecker, E., and Psik, T. Using ARToolKit markers to build tangible prototypes and simulate other technologies. In *Proceedings of the 2005 IFIP TC13 in-*

ternational conference on Human-Computer Interaction (Berlin, Heidelberg, 2005), INTERACT'05, Springer-Verlag, pp. 30–42.

[10] Kawakami, A., Tsukada, K., Kambara, K., and Siio, I. Potpet: pet-like flowerpot robot. In *Proceedings of the fifth international conference on Tangible, embedded, and embodied interaction* (New York, NY, USA, 2011), TEI '11, ACM, pp. 263–264.

[11] Koenig, S., and Likhachev, M. Fast replanning for navigation in unknown terrain. *IEEE Transactions on Robotics 21*, 3 (2005), 354–363.

[12] Kortenkamp, D., and Weymouth, T. Topological mapping for mobile robots using a combination of sonar and vision sensing. In *Proceedings of the twelfth national conference on Artificial intelligence (vol. 2)* (Menlo Park, CA, USA, 1994), AAAI'94, American Association for Artificial Intelligence, pp. 979–984.

[13] Kraetzschmar, G. K., Pagès Gassull, G., and Uhl, K. Probabilistic quadtrees for variable-resolution mapping of large environments. In *Proceedings of the 5th IFAC/EURON Symposium on Intelligent Autonomous Vehicles* (Lisbon, Portugal, July 2004), M. I. Ribeiro and J. Santos Victor, Eds., Elsevier Science.

[14] Kuffner, J. Efficient optimal search of euclidean-cost grids and lattices. In *Proc. IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS'04)* (September 2004), IEEE.

[15] Li, Y., Wang, Y., and Liu, Y. Fiducial marker based on projective invariant for augmented reality. *J. Comput. Sci. Technol. 22*, 6 (2007), 890–897.

[16] Mutka, A., Miklic, D., Draganjac, I., and Bogdan, S. A low cost vision based localization system using fiducial markers. *World Congress 17* (2008), 9528–9533.

[17] Nosrati, M., Karimi, R., and Hasanvand, H. A. Investigation of the * (star) search algorithms: Characteristics, methods and approaches. *World Applied Programming 2*, 4 (April 2012), 251–256.

[18] Owen, C. B., Xiao, F., and Middlin, P. What is the best fiducial? In *The First IEEE International Augmented Reality Toolkit Workshop* (Darmstadt, Germany, Sept. 2002), pp. 98–105.

[19] Portugal, D., and Rocha, R. P. Extracting topological information from grid maps for robot navigation. In *Proc. of 4th Int. Conf. on Agents and Artificial Intelligence (ICAART'2012)* (Vilamoura, Algarve, Feb. 2012), pp. 137–143.

[20] Sementille, A. C., Lourenço, L. E., Brega, J. R. F., and Rodello, I. A motion capture system using passive markers. In *Proceedings of the 2004 ACM SIGGRAPH international conference on Virtual Reality continuum and its applications in industry* (New York, NY, USA, 2004), VRCAI '04, ACM, pp. 440–447.

[21] Speers, A., Topol, A., Zacher, J., Codd-Downey, R., Verzijlenberg, B., and Jenkin, M. Monitoring underwater sensors with an amphibious robot. In *Proceedings of the 2011 Canadian Conference on Computer and Robot Vision* (Washington, DC, USA, 2011), CRV '11, IEEE Computer Society, pp. 153–159.

[22] Stanco, F., Tanasi, D., Gallo, G., Buffa, M., and Basile, B. Augmented perception of the past. the case of hellenistic syracuse. *Journal of Multimedia 7*, 2 (2012), 211–216.

[23] Stentz, A. The focussed D* algorithm for real-time replanning. In *Proceedings of the 14th international joint conference on Artificial intelligence - Volume 2* (San Francisco, CA, USA, 1995), IJCAI'95, Morgan Kaufmann Publishers Inc., pp. 1652–1659.

[24] Stentz, A. T. Optimal and efficient path planning for partially-known environments. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA '94)* (May 1994), vol. 4, pp. 3310 – 3317.

[25] Stentz, A. T. Map-based strategies for robot navigation in unknown environments. In *Proceedings of the AAAI Spring Symposium on Planning with Incomplete Information for Robot Problems* (March 1996).

[26] Sumbal, M. S. U. K. Environment detection and path planning using the e-puck robot. Master's thesis, University of Girona, 2010.

[27] Sun, X., Koenig, S., and Yeoh, W. Generalized adaptive A*. In *Proceedings of the 7th international joint conference on Autonomous agents and multiagent systems - Volume 1* (Richland, SC, 2008), AAMAS '08, International Foundation for Autonomous Agents and Multiagent Systems, pp. 469–476.

[28] Sun, X., Yeoh, W., and Koenig, S. Generalized fringe-retrieving A*: faster moving target search on state lattices. In *Proceedings of the 9th International Con-*

ference on Autonomous Agents and Multiagent Systems: volume 1 - Volume 1 (Richland, SC, 2010), AAMAS '10, International Foundation for Autonomous Agents and Multiagent Systems, pp. 1081–1088.

[29] Sun, X., Yeoh, W., and Koenig, S. Moving target D* Lite. In *Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems: volume 1 - Volume 1* (Richland, SC, 2010), AAMAS '10, International Foundation for Autonomous Agents and Multiagent Systems, pp. 67–74.

[30] Thrun, S. Learning occupancy grid maps with forward sensor models. *Auton. Robots 15*, 2 (Sept. 2003), 111–127.

[31] Thrun, S., and Bcken, A. Learning maps for indoor mobile robot navigation. *Artificial Intelligence 99* (1998), 21–71.

[32] Thrun, S., and Bucken, A. Integrating grid-based and topological maps for mobile robot navigation. In *Proceedings of the thirteenth national conference on Artificial intelligence - Volume 2* (1996), AAAI'96, AAAI Press, pp. 944–950.

[33] Wooden, D. T. *Graph-based path planning for mobile robots.* PhD thesis, Georgia Institute of Technology, Atlanta, GA, USA, 2006. AAI3248795.

[34] Xu, A., and Dudek, G. Fourier tag: A smoothly degradable fiducial marker system with configurable payload capacity. In *Proceedings of the 2011 Canadian Conference on Computer and Robot Vision* (Washington, DC, USA, 2011), CRV '11, IEEE Computer Society, pp. 40–47.