Optimization of Waypoint-Guided Potential Field Navigation Using Evolutionary Algorithms

J. Savage and E. Marquez
Laboratorio de Interfaces Inteligentes
University of Mexico
Mexico City, Mexico

J. Pettersson, N. Trygg, A. Petersson, and M. Wahde
Department of Machine and Vehicle Systems
Chalmers University of Technology
Göteborg, Sweden

***Abstract*— This paper describes a method for optimization of waypoint selection for potential field navigation in autonomous robots. In the method presented here, a genetic algorithm (GA) is used for optimizing the potential field. The chromosome of each individual encodes parametrizations for the potential field generated by waypoints, obstacles, and goals. The waypoints themselves are obtained through a Voronoi tessellation of the environment in which the robot is operating. It is demonstrated that the algorithm allows a robot to navigate safely and efficiently through spaces with many obstacles, even in cases where these are placed in a strongly unfavorable way.**

Furthermore, the results from simulations were implemented successfully in an actual Khepera robot. Using a slightly simplified navigation procedure, in which the robot comes to a standstill between successive steps in the navigation, the Khepera robot managed to navigate through one of the most difficult environments used in the simulations.

Finally, the paper briefly describes a different implementation of potential field navigation, in the path planning adaptation submodule of a more advanced simulated mobile robot (VirBot).

I. Introduction

In structured environments, where the locations of obstacles remain constant or, at least, can be predicted, potential field navigation is a useful method for robotic navigation, provided that the robot can obtain information about its location, either directly via e.g. GPS or indirectly via e.g. visual sensors combined with dead reckoning. Potential field navigation was introduced in 1986 by Khatib [1], and has since then been used in many different applications (see e.g. [2], [3], [4], [5], and [6]). However, a robot using the simplest form of potential field navigation will often get stuck, due to the presence of local minima in the potential field: since the method is gradient-based, the robot will be unable to escape from a local minimum. The problem can be solved by the introduction of waypoints, i.e. local goals (attractive potentials) along the path of the robot. It is not trivial, however, to select the location of waypoints and the exact shape of waypoint potentials.

In [7] the authors used series of waypoints in order to reach the designated goal in a path planning application. However, between each waypoint the path was constrained to be a straight line and the genetic algorithm was used solely for the optimization of the number of waypoints and their locations; no potential fields were used in conjunction with these waypoints.

The aim of this paper is twofold: (1) to describe a method for waypoint placement in potential field navigation and (2) to introduce a method for automatic generation of the potential field generated by the navigation goal, obstacles, and waypoints.

II. Potential field navigation

In potential field navigation the robot is considered as a particle under the influence of an artificial potential field U whose local variations reflect e.g. the positions of obstacles and of the goal that the robot is supposed to reach [8]. The potential field function is defined as the sum of an attraction field that pulls the robot towards the goal and a repulsive field that repels it from the obstacles. The movement is executed in an iterative way, in which an artificial force is induced by

$$\vec{F}(q) = -\vec{\nabla}U(q) \tag{1}$$

that forces the robot to move to the direction that the potential field decrees, where $\vec{\nabla}$ is the gradient with respect to q and $q = (x, y)$ represents the coordinates of the robot position. The complete potential field is a superposition of contributions from obstacles, waypoints (if applicable), and the goal:

$$U(q) = \sum_{j=1}^{n_{\rm o}} U_j^{\rm o}(q) + \sum_{j=1}^{n_{\rm w}} U_j^{\rm w}(q) + U^{\rm g}(q), \tag{2}$$

where $n_{\rm o}$ and $n_{\rm w}$ denote the number of obstacles and waypoints, respectively, and $U_j^{\rm o}$ and $U_j^{\rm w}$ are their potential. $U^{\rm g}$ is the potential generated by the goal (navigation target).

A. Path generation using potential fields

In general, the force defined in Eq. (1) is not applied directly to dictate the motion of the robot, since the magnitude of the force (and hence the resulting acceleration) may vary strongly depending on the location of the robot. Instead, the force equation is normalized as

$$\vec{f}(q) = \frac{\vec{F}(q)}{\|\vec{F}(q)\|}, \tag{3}$$

and thus only used for generating the desired *direction* (heading) of the robot. In the simulations reported below,

the equations of motion of the simulated robot are taken as

$$M\dot{v} + \alpha v = A\left(\tau_{\mathrm{L}} + \tau_{\mathrm{R}}\right), \tag{4}$$

and

$$I\ddot{\varphi} + \beta\dot{\varphi} = B\left(-\tau_{\mathrm{L}} + \tau_{\mathrm{R}}\right), \tag{5}$$

where τ_{L} and τ_{R} denote the torques on the left and right wheel, respectively, v is the speed of the robot and φ its direction of motion. M is the mass of the robot and I its moment of inertia. α, A, β, and B are constants. The parameters were set so that the simulated robots were similar to a Khepera robot, which has a diameter of 55 mm and weighs around 80 grams.

The potential field provides a desired direction φ_{ref}. Furthermore, a reference speed v_{ref} should be specified. The motion control is performed by means of a simple proportional control law. Using the notation $\tau_v = \tau_{\mathrm{L}} + \tau_{\mathrm{R}}$ and $\tau_\varphi = -\tau_{\mathrm{L}} + \tau_{\mathrm{R}}$, the control law is defined by the two positive parameters C and D in the equations

$$\tau_v = \frac{\alpha v_{\mathrm{ref}}}{A} - C\left(v - v_{\mathrm{ref}}\right), \tag{6}$$

and

$$\tau_\varphi = -D\left(\varphi - \varphi_{\mathrm{ref}}\right). \tag{7}$$

In the simulations described in Sect. IV, v_{ref} is kept at a single, low value v^1_{ref} throughout the motion, except near obstacles where an even smaller value, v^0_{ref}, is used.

In implementations in actual robots rather than simulated ones, exact positions may be more difficult to obtain. In prepared environments, the robot may be given its locations using e.g. triangulation. Direct position information via GPS can also be used (in principle), but the requirements on accuracy may be too severe in most environments: in many cases, the robot must pass within centimeters from obstacles.

However, in other environments, which have not been specifically prepared for robotic navigation, the robot may have to rely on dead reckoning (occasionally re-calibrated at certain pre-specified locations). In such cases, it is imperative that the robot move in such a way that it can perform dead reckoning as accurately as possible. This topic will be discussed further in Sect. V-A below.

B. Specific potentials

Potentials of goals and waypoints should be attractive, whereas potentials of obstacles should be repulsive. In general, the equation

$$U(q) = a\ \mathrm{e}^{-\frac{(q-q_p)^2}{b^2}}, \tag{8}$$

is used, where q_p is the position of the object (goal, waypoint, or obstacle), b is a positive constant, and a is a negative constant in the case of attractive potentials, and a positive constant in the case of repulsive potentials.

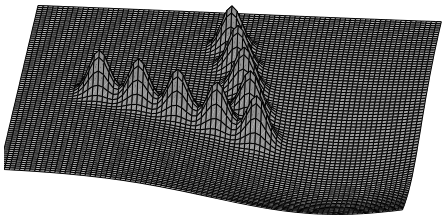

Fig. 1. An illustration of a potential generating a locking phenomenon. A simulated robot released in the upper left corner would be attracted towards the goal position in the lower right corner of the figure, only to find itself stuck inside the wedge-like obstacle configuration.

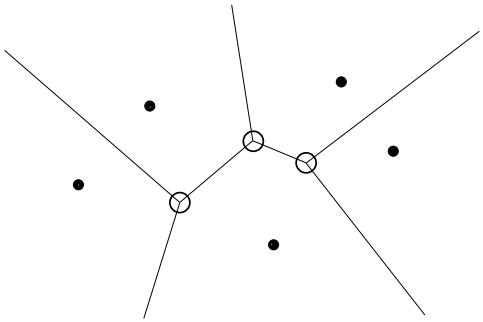

Fig. 2. A Voronoi diagram. The black dots represent the centers of obstacles, and the rings at the vertices of the resulting Voronoi diagram represent the locations of waypoints.

C. Navigation waypoints and their placement

In potentials field navigation, it is a common occurrence that a robot gets stuck in a local minimum of the potential field, a situation that will be referred to as the *locking phenomenon*. Local minima may appear, for example, due to an unfortunate placement of obstacles. One such situation is described in Fig. 1, in which a robot is attracted towards a goal position in the lower right corner of the figure, but ends up stuck in a wedge-shaped obstacle from which it cannot escape. In order to avoid such situations, the potential field can be augmented with waypoints represented by shallow attractive potentials, that help steer the robot towards the goal position. However, if waypoints are to be used, the problem of *where* to place them must be solved. Voronoi diagrams are a possible solution to this problem, and the method of choice in this paper. The method for generating such diagrams is described in [9], and can be summarized briefly as follows: the obstacles are considered as point-like objects, and are taken as central points (Voronoi generators) for the spatial tessellation. Next, polygons are shaped by drawing lines perpendicular to the lines connecting Voronoi generators, and the corners of the resulting polygons are taken as the waypoint locations. The procedure is illustrated in Fig. 2. When the waypoints have been placed (and their potentials determined, see Sect. IV), navigation proceeds as in standard potential field navigation, with the direction of

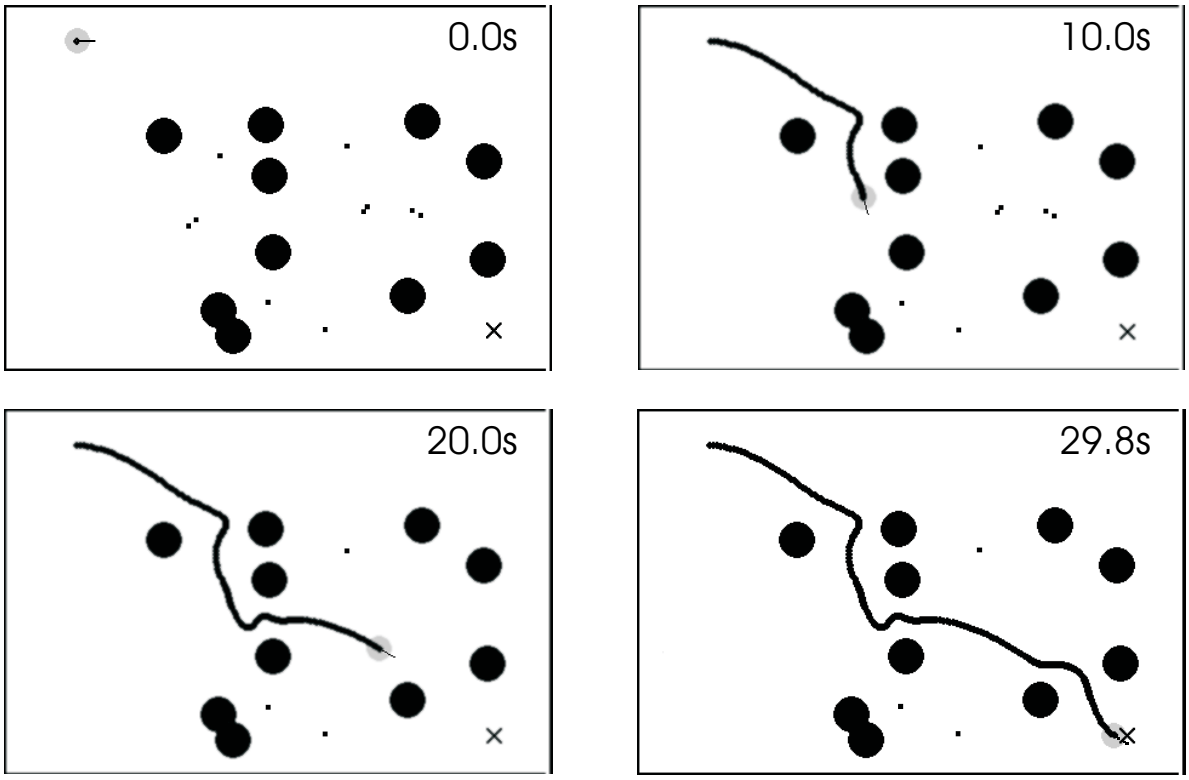

Fig. 3. Four snapshots showing a successful simulated robot moving from its starting position in the upper left corner of the environment to the goal position, marked with an X, in the lower right corner. The large black circles represent obstacles, and the small black dots represent waypoints. Note that the waypoints are successively removed as the simulated robot passes in their vicinity.

motion provided by the potential field, the only difference being that waypoints are successively removed as the robot passes in their vincinity to prevent it from being attracted back towards waypoints that have already been passed.

In this paper, the obstacles are all circular and thus well represented by the potential shown in Eq. (8). However, the waypoint placement procedure can be extended to larger, non-circular obstacles as well. Such obstacles would be divided into several smaller pieces, each of which could be represented by the potential given in Eq. (8).

D. Potential field optimization

Once the location of obstacles in a given environment is known, the location of waypoints is generated deterministically through the procedure just described. What remains to be determined is the depth (or height, in the case of obstacles) of the potentials, denoted $a_{\mathrm{g}}, a_{\mathrm{o}}$, and a_{w}, for the goal, obstacles, and waypoints, respectively, as well as the width of the potentials, denoted $b_{\mathrm{g}}, b_{\mathrm{o}}$, and b_{w}. Furthermore, the values of the set speeds v^1_{ref} (general navigation) and v^0_{ref} (near obstacles) must be determined as well as the distance d_{o} from the closest obstacle at which a simulated robot lowers its set speed from v^1_{ref} to v^0_{ref}. Finally, the distance d_{w} (between a waypoint and the robot) at which the waypoint is removed must also be determined. Thus, in all, a total of 10 parameters must be set and, in the implementation used here, their values are optimized using a standard genetic algorithm (GA) with tournament selection, single-point crossover, and mutation.

The chromosome consists of 10 values in the range $[0, 1[$, which, during the decoding procedure, are rescaled to appropriate ranges (see Sect. IV).

III. Simulator

A simulator was written, using the Delphi 5 (Object-oriented Pascal) language, for the purpose of investigating optimization of potential field navigation. The simulator generates environments containing N_{o} circular obstacles of width w_{o}[1] and one goal position. The simulated robots are evaluated in N_{e} different environments. Each simulated robot is associated with a chromosome, and the evaluation begins with a decoding procedure, during which the parameters in the chromosome are rescaled to appropriate values. Next, for each simulated environment, the robot is allowed to move under the influence of the potential field (whose exact shape is determined by the parameters obtained from the chromosome), until one of three things occurs, namely (1) the goal is reached, (2) the simulated robot hits an obstacle, or (3) a maximum time $T_{\max}$ is reached. Next, the fitness of the simulated robot for the environment in question is calculated as

$$f_i = \frac{T_{\max}}{T} \mathrm{e}^{-\frac{d}{D}}, \tag{9}$$

where T is the time at which the evaluation was terminated, d is the distance between the robot and the goal at the

[1]Note that the *actual* width of an obstacle, and the width of its potential, determined by the parameters a_{o} and b_{o}, may be very different.

termination point, and D is the initial distance between the robot and the goal. If the robot physically hits an obstacle, the evaluation is terminated immediately and T is set to $T_{\max}$. With this fitness measure, the robot is rewarded for moving quickly, and without collisions, towards the goal.

When the simulated robot has been evaluated in all $N_{\rm e}$ environments, the fitness values f_i, $i = 1, \ldots, N_{\rm e}$ are weighed together in one of two ways, either

$$f^{(1)} = \frac{1}{N_{\rm e}} \sum_{i=1}^{N_{\rm e}} f_i, \tag{10}$$

or

$$f^{(2)} = \min_i f_i. \tag{11}$$

IV. Results from simulations

Several simulation were performed, using both fitness measures defined above (see Eqs. (10) and (11)). All runs lasted for $1\,500$ generations, and the population consisted of 200 individuals. Each simulated robot was evaluated against $N_{\rm e} = 20$ randomly generated (but fixed, throughout all runs) environments, with $N_{\rm o}$ (the number of obstacles) in the range $[7, 12]$. The size of each environment was 1 meter by 1.5 meters, i.e of order 15-20 times the diameter of the Khepera robot. $T_{\max}$ was set to 200 s, even though the evolved robots usually managed to traverse the arena much faster (see Fig. 3).

In the decoding procedure, during which the 10 parameters of the chromosome are obtained, the set speed values $v^1_{\rm ref}$ and $v^0_{\rm ref}$ were rescaled to the interval $[0.02, 0.10]$ m/s, and the parameter $d_{\rm w}$ was rescaled to the interval $[0, 0.20]$ m. No rescaling was applied to the parameter $d_{\rm o}$, which therefore took values in the full range $[0, 1[$. The six first parameters in the chromosome, determining the exact shape of the potential field, were not rescaled (except that $a_{\rm g}$ and $a_{\rm w}$ were negated), since all that is relevant is the direction (not the magnitude) of the force obtained from Eq. (1). Experiments were performed using different (physical) obstacle diameters. In order to simplify the search performed by the GA, in some runs four of the 10 parameters (namely $b_{\rm w}$, $d_{\rm o}$, $v^1_{\rm ref}$, and $v^0_{\rm ref}$) in the chromosome were given pre-specificed values, thus reducing the number of GA-optimized parameters to six in those runs. The setup for eight representative runs is given in Table I, and the results of the runs are shown in Tables II and III. As is evident from the tables, the procedure of generating potentials through Voronoi diagrams and parameter optimization by means of GAs was successful in all cases where fitness measure $f^{(2)}$ was used, whereas runs using fitness measure $f^{(1)}$ reached less satisfactory results. This is understandable, as the fitness measure $f^{(1)}$ only measures average performance. Thus, with that fitness measure, it is possible for the simulated robot to fail completely in environments and still achieve a rather high average. By contrast, fitness measure $f^{(2)}$ focuses completely on the worst performance of the robot, and tends to generate much more robust results. In fact, in *all* runs with this fitness measure, the simulated robot corresponding to the best individual managed to reach its goal position in all environments. The parameter configurations obtained in those runs allow the robot to navigate safely in a large variety of environments containing obstacles of a given size. Thus, it would not be necessary to rerun the the optimization procedures if the positions of the obstacles were altered.

The waypoints turned out to play a pivotal role. Even though their influence was strongly localized to their immediate vicinity, they helped guide the simulated robot safely towards its goal. If the waypoint potentials were turned off, the robot usually failed to reach the goal position.

TABLE I

Simulation setup for eight representative runs. $w_{\rm o}$ is the physical obstacle diameter, $P_{\rm cross}$ is the crossover probability and $P_{\rm mut}$ the mutation rate, used by the GA. The population size was equal to 200 in all runs.

Run #	$w_{\rm o}$ [m]	$P_{\rm cross}$	$P_{\rm mut}$	fitness measure
1	0.05	0.80	0.100	$f^{(1)}$
2	0.07	0.70	0.080	$f^{(1)}$
3	0.05	0.80	0.100	$f^{(1)}$
4	0.07	0.80	0.100	$f^{(1)}$
5	0.05	0.50	0.100	$f^{(2)}$
6	0.05	0.75	0.100	$f^{(2)}$
7	0.07	0.50	0.075	$f^{(2)}$
8	0.10	0.80	0.050	$f^{(2)}$

TABLE II

Results for Runs 1-8. The second column shows the number of environments (out of a maximum of 20) in which the best agent in the corresponding run managed to reach the goal position in time $T_{\max}$ or less, and the third column shows the fitness values. Note that the fitness values of Runs 1-4 are not directly comparable to those of Runs 5-8.

Run #	# goals reached	maximum fitness	fitness measure
1	20	0.0869	$f^{(1)}$
2	18	0.0478	$f^{(1)}$
3	14	0.1492	$f^{(1)}$
4	19	0.1173	$f^{(1)}$
5	20	0.0722	$f^{(2)}$
6	20	0.1274	$f^{(2)}$
7	20	0.0962	$f^{(2)}$
8	20	0.0466	$f^{(2)}$

V. Implementations in actual robots

A. Khepera robots

It is of essential importance to verify, in actual robots, the results obtained in simulations [10]. Thus, a preliminary test of the simulation results has been performed, using a Khepera robot[2] in standard configuration (i.e. without added sensors or means of communication). These robots are equipped with incremental encoders, on the motor axis of each wheel, that can be used for determining the distance traveled by the robot. It was soon realized, however,

[2]The Khepera is a small, differentially steered two-wheel robot, manufactured by K-team (www.k-team.com).

TABLE III

PARAMETERS OBTAINED FOR THE BEST INDIVIDUALS IN RUNS 1-8. THE VALUES FOR THE FIRST SIX PARAMETERS ARE IN ARBITRARY UNITS, AND THE LAST FOUR PARAMETERS ARE GIVEN IN SI UNITS. PARAMETERS IN *italics* WERE KEPT CONSTANT, I.E. THE VALUES OBTAINED FROM THE CHROMOSOME WERE NOT USED. IN CASES WITH $d_{\rm o} = 0$, THE PARAMETER $v^0_{\rm ref}$ IS IRRELEVANT (AND THUS NOT GIVEN), SINCE IT WILL NEVER BE ACTIVATED.

Run #	$a_{\rm g}$	$b_{\rm g}$	$a_{\rm o}$	$b_{\rm o}$	$a_{\rm w}$	$b_{\rm w}$	$d_{\rm w}$	$v^1_{\rm ref}$	$v^0_{\rm ref}$	$d_{\rm o}$
1	0.746	0.352	0.644	*0.045*	0.018	0.190	0.198	*0.040*	—	*0.000*
2	0.975	0.637	0.130	*0.084*	0.009	0.341	0.188	*0.040*	—	*0.000*
3	0.586	0.344	0.169	0.060	0.002	1.295	0.123	0.043	0.099	0.594
4	0.885	0.631	0.492	0.057	0.002	1.064	0.079	0.100	0.041	0.082
5	0.836	0.390	0.611	*0.045*	0.010	0.494	0.155	*0.040*	—	*0.000*
6	0.886	0.434	0.425	0.045	0.016	0.200	0.086	0.099	0.039	0.115
7	0.777	0.397	0.565	0.057	0.016	0.539	0.006	0.099	0.044	0.085
8	0.569	0.463	0.263	0.057	0.346	0.063	0.167	0.098	0.029	0.128

that the navigation accuracy obtained by performing dead reckoning using the incremental encoders on the Khepera was not sufficient for dynamic motion according to Eqs. (6) and (7). Instead, a simplified scheme was implemented, in which the robot navigates in discrete steps. In each step the robot starts from a standstill, determines (via the potential field $U(q)$) its desired direction of heading, rotates to face this direction, moves a distance δ in this direction, so that the position changes according to

$$q_{i+1} = q_i + \delta \vec{f}(q), \tag{12}$$

and then stops again to compute a new desired direction etc. Provided that δ is chosen sufficiently small, i.e. smaller than the typical length scale of variations in the potential field, this navigation procedure represents a slow-motion version of that used in the simulations. Using this procedure, and after some calibration of the incremental encoders as well as the acceleration and deceleration phases of each movement, the Khepera rather successfully (but slowly) navigated through one of the most difficult environments used in the simulations, namely that shown in Fig. 3. A snapshot of the navigating robot is shown in Fig. 4. On some occasions, however, the Khepera was unable to reach the goal. The failure could be attributed partly to the limited accuracy provided by the incremental encoders, which caused a continuously increasing deviation between the actual and estimated position of the robot. Furthermore, there was a small problem with the motors in the Khepera robot, when running on batteries: for a few steps of the motion, one of the motors would temporarily cease to function, probably because of a malfunctioning battery, since the problem could be eliminated when using an external power source. Finally, in narrow passages between obstacles, the robot sometimes displayed an oscillatory behavior. This is a well-known phenomenon [11] in potential fields navigation, and was indeed noticed also in the simulations (see the two bottom panels in Fig. 3). While the oscillations are harmless in the simulations, they tend to further deteriorate the accuracy of the dead reckoning in the Khepera robot experiments.

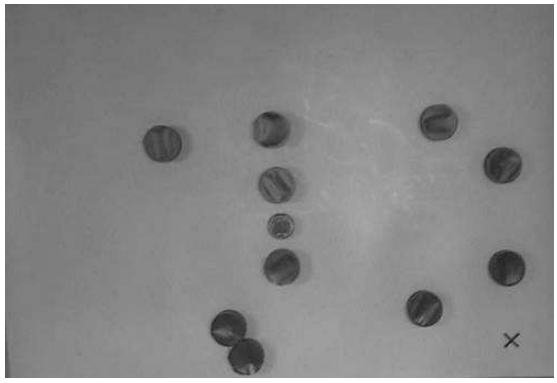

Fig. 4. A snapshot of the Khepera robot navigating through one of the environments used in the simulations (cf. Fig. 3). The navigation goal is marked with a cross, and the Khepera robot is located close to the middle of the figure.

B. VirBot

The VirBot system [12], illustrated in Fig. 5, is a considerably more advanced robotic system than Khepera. While VirBot also is a simulated robot, the VirBot structure illustrated in Fig. 5 has been implemented in an actual Nomad type robot. Potential field navigation using waypoint optimization has been implemented in the path planning submodule of the VirBot system. While the implementation shares certain basic features with the simulated systems and the Khepera implementation described above, the VirBot implementation is different in the sense that it attempts to optimize the *positions* of waypoints rather than having them generated automatically using the Voronoi tessellation procedure. A simulator for the VirBot system has also been written, and it has been tested successfully in a variety of environments. However, because of the need to evolve the locations of waypoints (which may be very time-consuming) for each new environment, this method is perhaps less suitable in realistic applications.

VI. DISCUSSION AND CONCLUSION

The exact potentials that evolved for the waypoints varied quite strongly, as evidenced by the results shown

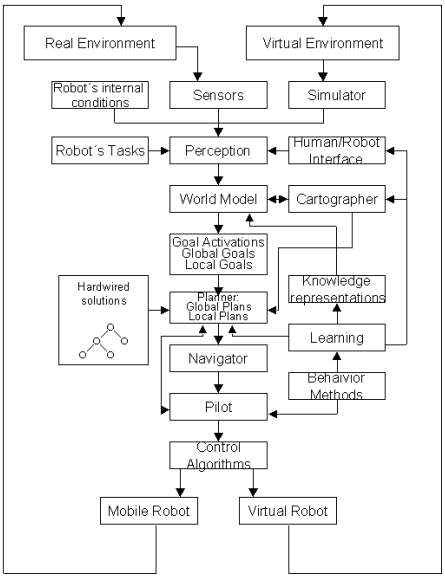

Fig. 5. The VirBot system.

in Table III. In some runs (e.g. Run 3), shallow but wide-ranging waypoint potentials were evolved, whereas in others (e.g. Run 8), the waypoint potentials were deep but narrow. It is interesting to note that the GA was able to find both these types of waypoints, and the results indicate that, while the use of waypoints is crucial to avoid locking phenomena, their exact shape can be chosen in many different ways.

Another indication that the GA may choose to use the available parameters in a way that differs from the intentions of the experimenter is given by Run 3, in which the speed near obstacles (v_{ref}^{0}) was actually set to a *higher* value than the nominal navigation speed. However, in this run, the parameter d_{o} was also set to a very high value, so that the simulated robot considered itself to be near obstacles almost all the time. In this case, the simulated robot moved quite fast, and obtained a high fitness value in those environments for which it managed to reach the goal, but at the cost of many complete failures. Indeed, the goal was only reached in 14 out of 20 environments, as shown in Table II.

The behavior of simulated robots evolved with fitness measure $f^{(2)}$ was more robust, and their navigation more careful than that of simulated robots evolved with fitness measure $f^{(1)}$.

The main conclusion of this investigation is that it is indeed feasible to evolve potential fields, containing waypoints, for efficient and robust robotic navigation, provided that the fitness measure used by the GA is chosen carefully. Even though the runs lasted $1\,500$ generations, satisfactory results were often obtained after a few hundred generations.

As is often the case when GAs are used, the performance of the algorithm depended quite strongly on the choice of fitness measure, and the general conclusion is that a more challenging fitness measure, if properly chosen, will generate more robust results.

As for the preliminary tests that were performed using a Khepera robot, the conclusion is that it is indeed possible, in this case, to transfer the simulation results more or less directly to an actual robot. However, not surprisingly, the results obtained with the Khepera are not very robust, due to the limited accuracy of the incremental encoders, and the occasional problems with the batteries reported above.

More tests are underway, in which attempts are made to find the optimal distance δ between successive readings of the potential field. In addition, adding a camera to the Khepera robot, attempts will be made to supplement the dead reckoning with landmark recognition, allowing the robot to navigate in a similar fashion to e.g. desert ants [13], by taking and storing snapshots of useful features in the environment. Interesting issues in this respect are the problems of determining *when* to take snapshots of landmarks, as well as determining *when* and *how* to use them.

ACKNOWLEDGMENT

The authors wish to express their gratitude to *Stiftelsen för internationalisering av högre utbildning och forskning*, STINT, which provided part of the funding for this project.

REFERENCES

[1] O. Khatib, "Real-time obstacle avoidance for manipulators and mobile robots," *International Journal of Robotics Research*, vol. 5, no. 1, pp. 90–98, 1986.

[2] G. Dozier, A. Homaifar, S. Bryson, and L. Moore, "Artificial potential field based robot navigation, dynamic constrained optimization and simple genetic hill-climbing," in *The 1998 IEEE International Conference on Evolutionary Computation*, pp. 189–194, May 1998.

[3] P. Vadakkepat, K. Tan, and M. Wang, "Evolutionary artificial potential fields and their application in real time robot path planning," in *Proceedings of the 2000 Congress on Evolutionary Computation*, vol. 1, pp. 256–263, July 2000.

[4] J. Guldner and V. Utkin, "Sliding mode control for gradient tracking and robot navigation using artificial potential fields," *IEEE Transactions on Robotics and Automation*, vol. 11, pp. 247–254, April 1995.

[5] M. G. Park, J. H. Jeon, and M. C. Lee, "Obstacle avoidance for mobile robots using artificial potential field approach with simulated annealing," in *IEEE International Symposium on Industrial Electronics*, vol. 3, pp. 1530–1535, June 2001.

[6] C. Warren, "Global path planning using artificial potential fields," in *IEEE International Conference on Robotics and Automation*, vol. 1, pp. 316–321, May 1989.

[7] B. Capozzi and J. Vagners, "Evolution as a guide for autonomous vehicle path planning and coordination," in *IEEE Aerospace Conference Proceedings*, vol. 5, pp. 2527–2536, March 2002.

[8] J.-C. Latombe, *Robot motion planning*. Kluwer Academic Publishers, 1991.

[9] F. Aurenhammer, "Voronoi diagrams - a survey of a fundamental geometric data structure," *ACM Computing Surveys*, vol. 23, no. 3, pp. 345–405, 1991.

[10] N. Jakobi, P. Husbands, and I. Harvey, "Noise and the reality gap: The use of simulation in evolutionary robotics," *Lecture Notes in Computer Science*, vol. 929, p. 704, 1995.

[11] Y. Koren and J. Borenstein, "Potential field methods and their inherent limitations for mobile robot navigation," in *Proceedings of the IEEE Conference on Robotics and Automation*, pp. 1398–1404, April 1991.

[12] J. Savage, M. Billinhurst, and A. Holden, "The virbot: a virtual reality robot driven with multimodal commands," *Expert Systems with Applications*, pp. 413–419, 1998.

[13] S. Judd and T. Collett, "Multiple stored views and landmark guidance in ants," *Nature*, vol. 392, pp. 710–714, 1998.