Towards Autonomous Robotic Systems (TAROS), UK, LNCS 10454, pp. 211-221, Springer 2017

Local real-time motion planning using evolutionary optimization*

Steffen Müller, Thanh Q. Trinh and Horst-Michael Gross

Technische Universität Ilmenau, 98693 Ilmenau, Germany,
steffen.mueller@tu-ilmenau.de,
WWW home page: http://www.tu-ilmenau.de/neurob

Abstract. In order to allow for flexible realization of diverse navigation tasks of mobile robots, objective-based motion planner proved to be very successful. The quality of a selected control command for a certain time step is inherently connected to the considered diversity of future trajectories. Therefore, we propose an evolutionary motion planning (EMP) method to solve this high-dimensional search problem without restricting the search space. The algorithm optimizes sequences of acceleration commands with respect to objective functions for evaluating the resulting movement trajectories. The method has been successfully deployed on two robots with differential drive, and experiments showed that it outperforms the Dynamic Window Approach [1] with its restricted discretized search space. Furthermore, car-like and holonomic robots could be controlled successfully in simulations.

Keywords: Motion Planning, Motion Control, Evolutionary Optimization

1 Introduction

For a general purpose service robot, navigation skills in a populated and sometimes constricted environment are essential. The projects of our lab involving navigation in home environments [2] and also public buildings [3] showed that a simple target-directed, obstacle avoiding motion behavior often is not sufficient. For example, in a public environment social aspects have to be considered in order to perform a navigation behavior that is accepted by the people indirectly involved in the robot's activities. Soft constraints, like respecting people's personal space or driving on the right side of an aisle, have to be considered in the motion planning algorithm in addition to the hard criteria of collision avoidance and target directed motion. In a home environment, socially assistive robot companions additionally may want to communicate their internal emotional state by means of different movement styles while performing navigation tasks. As part

* This work has received funding from the German Federal Ministry of Education and Research (BMBF) to the project SYMPARTNER (grant agreement no. 16SV7218).

Towards Autonomous Robotic Systems (TAROS), UK, LNCS 10454, pp. 211-221, Springer 2017

of the SYMPARTNER[1] project, we currently work on an objective function for rating movement trajectories of a robot according to emotional expressivity.

In order to realize diverse navigation behaviors, in our lab the Dynamic Window Approach (DWA), an objective-based motion planner introduced by Fox et al. [1], has been further developed and successfully applied for several real-world scenarios ranging from large public shopping centers and hospitals, to small and constricted senior apartments. Nevertheless, for generating very specific motion patterns, in our case the DWA reached its limits due to the very restricted types of hypothetical movement trajectories used for evaluation. The assumption of constant future velocity only generates arc-like trajectories. Especially, complex objective functions, like those for consideration of personal space and emotional expressivity, need more complex unrolled candidate trajectories than the arc-like ones of the DWA. Therefore, we propose a new method for generating trajectories with a high degree of freedom, which then are optimized in order to satisfy the various objectives mentioned before.

The remainder of this paper is organized as follows: First, a brief discussion of alternative motion planning methods is given followed by a description of the objective-based navigation framework used in our lab. After that, our evolutionary optimization approach will be introduced, followed by some experimental results.

2 Related Work

In the robotics field, decoupling the motion planning into global and local planner is a well established paradigm. The global planner operates on a coarser world representation and generates a route to the goal, whereas the local planner has to find optimal control commands to follow the global plan respecting the robot's dynamic constraints and the dynamics of the sensed environment.

One of the first local planner, which was able to operate a robot with non-holonomic dynamics, more specifically a synchro-drive robot, was the DWA [1]. The DWA directly searches in the velocity space adhering to the robot's dynamics. To keep the search computationally feasible, the velocity space is discretized, and a constant velocity is assumed for the whole planning horizon. However, this assumptions results in very restricted trajectories.

More flexible trajectories can be considered with a lattice-based discretization of the state space using motion primitives [4, 5]. As result, the state space is represented as a graph with nodes standing for states, and edges connecting nodes which are reachable using the motion primitives. Then search algorithms are used on the graph to generate feasible trajectories. The trajectories' flexibility is determined by the amount of motion primitives. This must be chosen carefully as a greater number of motion primitives can result in a computational burden, but too few primitives generate inflexible trajectories as well. The Rapidly Exploring Random Tree (RRT) approach [6, 7] is not subject to this restriction, since it

[1] SYMbiosis of PAul and RoboT companion for Emotion sensitive caRe (www.sympartner.de)

builds a representation by sampling the state space. Beginning with the robot's position as initial tree node, in each iteration a new state is sampled, and the tree is expanded towards that sample at the closest existing tree node. Although, RRTs can generate a feasible solution even for higher dimensional state spaces, the resulting trajectories for common mobile platforms are jerky, and additional smoothing is needed. Furthermore, the definition of a distance metric required to find the closest node is not always trivial (e.g. for car-like vehicles).

So, we propose to use an evolutionary algorithm [8] for solving the optimization problem. Similar to the DWA, our approach conducts the search in the space of possible control commands to satisfy the robot's dynamic constraints. Unlike other approaches, no constant velocity is assumed over the planning horizon, nor a discretization of the command space or predefined motion primitives are required. Overall, this is leading to more flexible candidate trajectories enabling a more accurate estimation of a control commands outcome in the future (see Fig. 1).

3 Objective-based Motion Planning

The main requirement for a motion planner is the ability to satisfy different combinations of constraints on the movement in various scenarios. The objective-based approach introduced with the DWA [1] can realize this by decomposing the resulting navigation behavior into a set of objective functions, each of them responsible for a certain aspect. By means of an interface where the higher level application only specifies a navigation task, the objectives are activated or deactivated individually depending on the current needs (see [3] for further details on the architecture of our navigation stack). Each of the objectives is able to evaluate costs for a given hypothetic movement trajectory of the robot. These costs are real-valued numbers or even a hard deny which prohibits a trajectory if it led to a collision, for example. The motion planner then has to combine the costs of all active objectives by means of a weighted sum, in order to find a global rating for a given trajectory.

For our experiments, a basic set of objective functions has been applied, which comprises

1. a **path and heading objective** responsible for approaching the minimum in a globally planned navigation function (using E* planner [9]) and turning towards a given goal orientation in proximity to the goal position,
2. a **distance objective** for avoiding collisions with static and dynamic obstacles,
3. a **direction objective** preferring forward motion of the robot to account for the limited sensor capabilities in the rear,
4. a **personal space objective** to keep distance to people in the close proximity of the robot by predicting their movements with a linear model

For all the objectives, the costs are computed for all the points along the trajectory. By averaging along the trajectories, future outcome as well as immediate

Towards Autonomous Robotic Systems (TAROS), UK, LNCS 10454, pp. 211-221, Springer 2017

costs are considered equally, which helps to find command sequences that contain the useful actions already in the beginning rather than in the far future parts, as it could be the case if only the trajectories' endpoints would be evaluated.

With these objectives, the aim of the motion planner is to generate potential movement trajectories which are associated with the next velocity command for the current planning cycle (250 ms in our case) and find the most suitable with respect to the rating yielded by the objectives. This means that a compromise of all active objectives has to be found. After that, the corresponding velocity command will be executed and the planning cycle starts again.

Since the selection of the next motor command has to be real-time capable, this optimization is only done for a local trajectory of a few seconds maximum. An appropriate time horizon has to be selected carefully. On the one hand, the minimum time frame considered is determined by the given physical properties of the robot. Maximum velocity and deceleration define a stopping time, which is the minimum planning horizon to include safe stopping trajectories in optimization. On the other hand, a small time horizon may lead to suboptimal solutions and potential oscillations. Therefore, the size of the predicted local trajectories should be as long as possible given the real-time restrictions. A robot then for example may be far-sighted enough to turn in front of a narrow passage in order to reach a target orientation at the end which may be unreachable if the robot could not turn inside the narrow gap. Also for avoiding moving obstacles (like walking people), a longer planning horizon with flexible trajectories allows to react on the changing situation more appropriate (see Sect. 5).

4 Evolutionary Optimization Algorithm

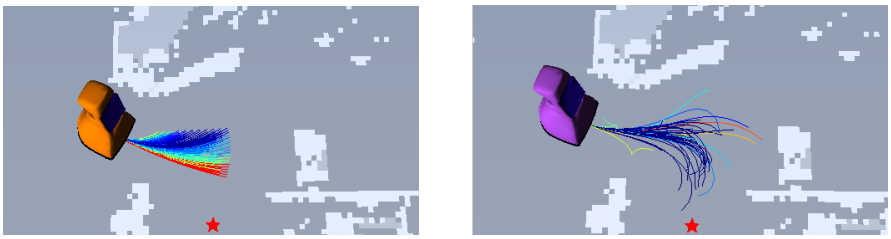

Fig. 1. Trajectories evaluated by the DWA (left) and our proposed approach (right), color codes for the costs (red high, blue low), the target is behind the lower border in direction of the red star.

For a given time horizon (e.g. 3.5 s in our case), the set of possible movement trajectories is enormous. For each time step, a real-valued tuple of translational v_x, v_y ($v_y = 0$ in case of differential drive) and rotational velocity v_ϕ is possible which define the actual trajectory over time if applied to the robot's motor controllers. So, the motion planner has to solve a difficult search problem.

Contrary to the DWA, which exhaustively searches the whole discretized velocity space and to be computationally tractable assumes a constant velocity

for the whole planning horizon, our approach's trajectories are far more flexible (see Fig. 1) by permitting individual velocities at each time step of the planning horizon. In order to search in this high dimensional trajectory space, an evolutionary algorithm is applied.

That means, we hold a population $P = \{A_i | i = 1, \ldots, M\}$ of possible acceleration sequences as individuals. These individuals $A_i = (\mathbf{a}^t | t = 0, \ldots, T)$ consist of an acceleration vector $\mathbf{a}^t = (a_x^t, a_y^t, a_\phi^t)$ for each time step t of the planning horizon. See Fig. 2 for a visualization of the encoding scheme. Most

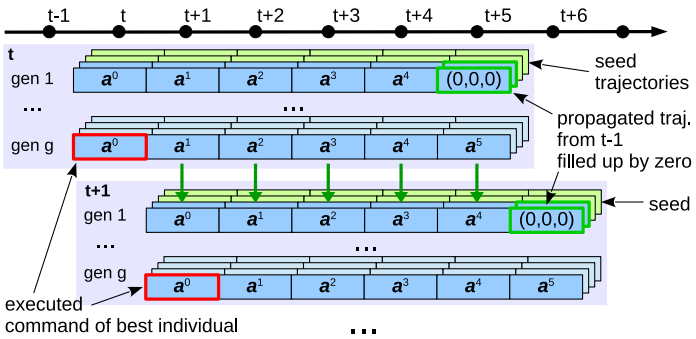

Fig. 2. Encoding of individuals as sequences of acceleration vectors which are optimized for g generations in each planner cycle, propagation of the final population of one planner cycle to the next helps reusing the optimization effort from past cycles and reduces number of generations needed in each planner cycle, which enables real time operation.

evolutionary algorithms consist of the following main steps, which can be found in our motion planner as well: (1) proper initialization of the population, (2) evaluation of each individual's fitness, (3) selection of the best individuals, and (4) producing a new generation by applying mutation and crossover operations on the best selected individuals. These steps are repated for several generations to bring the population closer to an optimum.

For application in the motion planning domain, due to real time restrictions, only a small number g of generations ($g = 5$ in our case) can be evaluated in the available time slots, but we can benefit from the fact, that trajectories of the subsequent time step are not independent of the last population. A trajectory found for one situation is also valid for immediately succeeding situations, except that the robot has simply traveled along that trajectory for a small step.

Propagating Individuals Therefore, we apply a propagation operation to the final population P of the last planning time step in order to initialize a new population P', which is going to be used as the start for the new planning cycle (see Fig. 2). The new population is built up from the best individuals of the last generation in the last planning cycle, which are shifted by one element in time. That means the acceleration vectors of $t = 0$ (which have been executed currently)

are cut off, and a new vector $(0, 0, 0)$ will be appended at the end to fill up the sequence. The remaining individuals of the population will be generated deterministically in order to ensure that the population at least contains a minimum number of individuals encoding a safe stopping trajectory. Since a randomized search algorithm is used for finding an optimal trajectory, otherwise it can not be guaranteed, that a collision free trajectory is included in the population. This means the green seed individuals of Fig. 2 consist of deceleration trajectories which try to reach zero velocity at different rotational and translational speeds.

Fitness Evaluation Having an initial population, we enter a loop of g generations for optimization. The first step in that loop is the evaluation of the fitness, for each individual in the population. Therefore, the acceleration sequence of each individual is transformed into a movement trajectory in space by means of a forward model of the robot. All these trajectories are then evaluated by the active objectives of our current motion task.

Selection of the Best After determining the fitness of the individuals, the next generation will be generated by combining two individuals from the last generation as parent for a new individual. In order to prefer better rated individuals for reproduction, the individuals are ordered by fitness and two random indices in that sorted list are drawn by means of a normal distribution with mean 0 representing the index of the best individual and a given variance specifying the selection pressure for the evolutionary algorithm. Over all generations, the best rated individual is stored for execution at the end of the planning cycle (this ensures that safe stop trajectories can be executed if non of the recombined individuals satisfy the constraints of the objectives).

Mutation and Crossover Once two parent individuals were drawn, the recombination of a new individual takes place by copying the acceleration vectors for each time step either from the first or from the second parent. With a probability of p ($p = 30\%$ in our case) the actual parent is switched after each time step. After, a new individual has been created, a mutation operation is applied, which operates in two modes. First, to each acceleration vector a normal distributed random acceleration is added and truncated to the physical limits. Second, a kind of symmetric mutation is executed. Therefore, two positions in the sequence are selected. Then a normal distributed random value is added to the acceleration at the first position and subtracted from the second position. This mutation is repeated for several iterations. Unlike the first mutation which results in curves and elongations of the resulting trajectories, small lateral translations are produced by the symmetric mutation.

Using only these operations on our robots, it showed that the resulting acceleration profiles are very noisy, and sometimes consecutive values compensate each other. Although, the evolution process would be able to find smooth trajectories if smoothness would be added as an objective, we decided to bias mutation towards smooth trajectories in order to reduce the search complexity. This has been achieved by randomly selecting segments of the acceleration sequence and applying a low pass filter on that part. As result, the trajectories are much smoother but still can take on arbitrary shapes.

Towards Autonomous Robotic Systems (TAROS), UK, LNCS 10454, pp. 211-221, Springer 2017

The computational effort in our tests was selected to be equal to the DWA approach. Since the main time is consumed by the costs evaluation in the objective functions, we used 5 generations with 60 individuals each resulting in 300 trajectory evaluations per time step (250 ms), which is the same amount as for a DWA with 15x20 bins in the velocity space. On our on board Core-i5 processor, the motion planner needs only 50% of one CPU core, which leaves enough power for perception skills and other parts of the service application.

The presented approach is not limited to tuples of acceleration. Individuals could be built up from arbitrary control commands, as long as a forward model for predicting the movement trajectories exists (e.g. steering angle and acceleration for car like robots). Therefore, also non-holonomic robots can be controlled using the proposed method. Errors in the prediction model are of minor relevance, since only the first command of an individual is really executed. For a high fitness, it is sufficient to know that there exists a possible control sequence producing a suitable trajectory starting with that first command. Deviations in the real behavior of the robot will be compensated in subsequent optimization cycles, since in each time step the true robot state is taken as the starting point for rendering the candidate trajectories.

5 Experimental Results

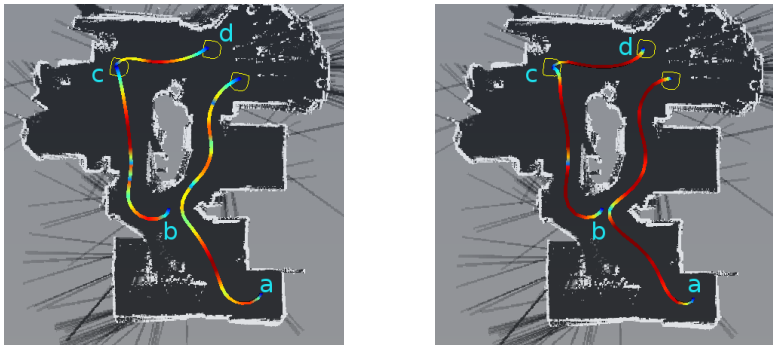

Fig. 3. Exemplary motion trajectories of the robot in one of the test apartments, (left) dynamic window approach, (right) evolutionary motion planner; the robot started at point (a) facing south for a first run and ended at the end pose depicted in yellow, the second run started at point (b), had an intermediate goal at (c) and ended at (d); color coded for velocity (red fast, blue slow)

The expected benefit of our evolutionary approach over the classical DWA is that it is more farsighted. The longer time horizon of locally planned trajectories can find solutions for the motion path, which can contain local cost peaks on a shorter scale. Therefore, the resulting movement behavior is much smoother and more natural than the one of the DWA, which has to slow down in narrow environments each time a fast bow trajectory would collide with an obstacle.

Especially motion paths requiring many changes in the curvature can be driven much faster.

Experiments with two of our SCITOS-G3 robots Max and Ringo confirmed that hypothesis. Max [2], constructed for applications in narrow home environments, has a maximum velocity of 0.6 m/s in forward direction and 0.3 m/s in backward direction. Rotational velocity was limited to 180°/s. Ringo [3], intended for public environments, is also a differential drive robot but has a maximum velocity of 0.9 m/s. Fig. 3 shows some exemplary trajectories in one of our test apartments. The color coding the velocity shows significantly smoother behavior of the evolutionary motion planner (right) compared to the DWA (left) using the same objectives for voting trajectories.

Even though the used objectives were identical, the behavior close to the goal is significantly different for the two planers. When in free space, the evolutionary planner approaches the goal in a wide bow tangentially to the target orientation, while the DWA first approaches the goal region and finally turns in place. This allows for a much faster approaching using the EMP and causes a significant improvement in the task execution time. For 500 random target approaches in the home-like environment of our living lab, the DWA took 3:52 hours compared to 2:51 hours required by the evolutionary planning algorithm. Fig. 4 shows some comparative histograms of the velocities reached by the DWA

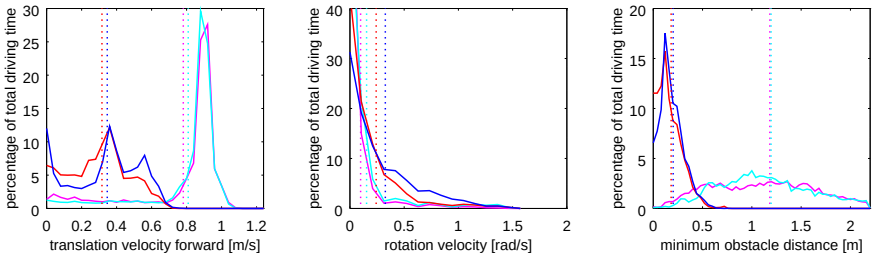

Fig. 4. Histograms of velocity and minimum obstacle distance during real-world random target navigation with our robots, Max in living lab (blue, red) and Ringo in office hallways (magenta, cyan); red and magenta: DWA; blue and cyan: EMP; dotted lines give the mean value of the histograms

and the evolutionary motion planner during the autonomous navigation in our experiments. From the obstacle distance histogram and the velocity histogram one can see, that in narrow environments, like our living lab, the evolutionary planner more often reached higher velocities (max. 0.6 m/s for Robot Max), which explains the enormous speedup compared to the DWA. In the free space environment of our office building the difference is not as large as in the narrow setting. Since most of the navigation path is straight ahead, the DWA is also able to find the best command in these situations. Only in the goal region or when passing doors, the DWA shows little more slow movements. Besides the basic

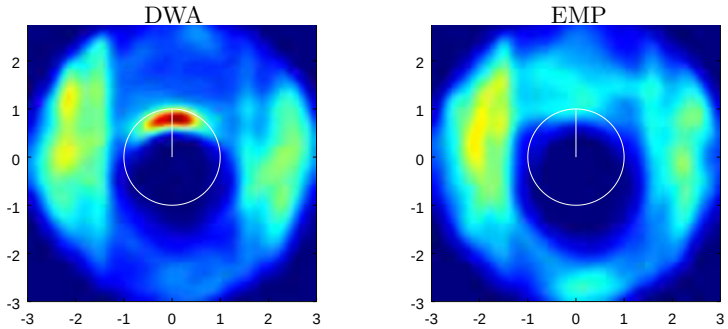

Fig. 5. Heat map of the robot's position relatively to all simulated persons. The white circle indicates the personal space of 1 m and the heading direction.

target oriented navigation, also the more complex objective functions for respecting the personal space (PS) were tested in comparison of DWA and evolutionary motion planning. Simulations were conducted to evaluate the planner's ability to avoid the PS of people in the robot's vicinity. To this end, a 20 m x 6 m sized room was constructed with eight moving pedestrians. The robot had to navigate across the room and had full knowledge of the surrounding persons' current positions and velocities within a radius of 5 m. To evaluate the performance, the relative positions of the robot to all the pedestrians were recorded. From this data, a heat map of the robot's positions and distances to the pedestrians (in the center) was extracted. Fig. 5 proves that the evolutionary motion planning did not significantly violate the personal space of the pedestrians, whereas the DWA driven robot entered the personal spaces more often by driving in front of the people. Often it simply stops in front of an approaching person unable to plan a suitable evasion maneuver (visible as a cumulation at the border of the PS in front of the person), while the evolutionary planner quickly finds an avoiding movement.

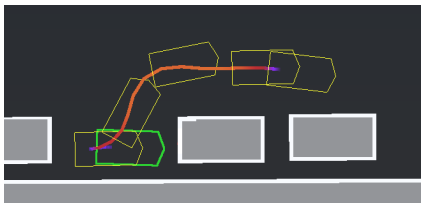

Fig. 6. Parallel parking of a vehicle with Ackermann steering geometry.The end position is marked in green. EMP was able to find the typical parking maneuver instead of using the shortest path to the parking spot.

In addition to the experiments with differential drive robots, we did simulations with a holonomic robot model (v_y not fixed to zero) as well. The planner could successfully control this kind of robot too. The basic idea of the EMP approach also seems to be applicable to control a car-like robot. Fig. 6 shows an experimental simulation run of a car parking setup. Remarkable is the ability to

plan a two phase movement with change of direction in order to reach the goal. For that experiment, the planning horizon and mutation rate was set very high in order to find satisfying trajectories within real-time constraints. Therefore, the resulting behavior is still relatively unstable. For such complex applications, further improvements to the algorithm are necessary.

6 Conclusion

We could show that the search problem for local motion control of a wide range of robots could be successfully solved within real-time restrictions without giving up flexible trajectories with a high degree of freedom. A key benefit of the proposed evolutionary motion planning algorithm is the reusability of the population from past time steps in order to focus the search on future promising control sequences. Experiments showed that the resulting navigation behavior of an objective-based motion planner is not only defined by the cost functions, but mainly depends on the diversity of the set of sampled future trajectories.

Preliminary results in simulations give indication that the presented approach is also able to control holonomic and car-like vehicles, which we intent to study in our future work.

References

1. Fox, D., Burgard, W., Thrun, S.: The dynamic window approach to collision avoidance. IEEE Robotics Automation Magazine 4(1), 23–33 (Mar 1997)
2. Gross, H.M., Müller, S., Schröter, C., Volkhardt, M., Scheidig, A., Debes, K., Richter, K., Döring, N.: Robot companion for domestic health assistance: Implementation, test and case study under everyday conditions in private apartments. In: IROS, 2015 IEEE/RSJ International Conference on. pp. 5992–5999 (Sept 2015)
3. Gross, H.M., Debes, K., Einhorn, E., Müller, S., Scheidig, A., Weinrich, C., Bley, A., Martin, C.: Mobile robotic rehabilitation assistant for walking and orientation training of stroke patients: A report on work in progress. In: 2014 IEEE SMC. pp. 1880–1887 (Oct 2014)
4. Likhachev, M., Ferguson, D.: Planning long dynamically feasible maneuvers for autonomous vehicles. The International Journal of Robotics Research 28(8), 933–945 (2009)
5. McNaughton, M., Urmson, C., Dolan, J.M., Lee, J.W.: Motion planning for autonomous driving with a conformal spatiotemporal lattice. In: IEEE ICRA 2011. pp. 4889–4895 (May 2011)
6. LaValle, S.M., Kuffner Jr, J.J.: Rapidly-exploring random trees: Progress and prospects. Algorithmic and Computational Robotics: New Directions (5), 293–308 (2007)
7. Karaman, S., Frazzoli, E.: Sampling-based algorithms for optimal motion planning. The International Journal of Robotics Research 30(7), 846–894 (2011)
8. Back, T.: Evolutionary algorithms in theory and practice: evolution strategies, evolutionary programming, genetic algorithms. Oxford university press (1996)
9. Philippsen, R., Siegwart, R.: An interpolated dynamic navigation function. In: Proceedings of the 2005 IEEE International Conference on Robotics and Automation. pp. 3782–3789 (April 2005)