EPiC Series in Computing

Volume 41, 2016, Pages 200–213

GCAI 2016. 2nd Global Conference on Artificial Intelligence

Heuristic Decision-Making for Human-aware Navigation in Domestic Environments

Alexandra Kirsch[1]

Eberhard Karls Universität Tübingen Tübingen, Germany
alexandra.kirsch@uni-tuebingen.de

Abstract

Robot navigation in domestic environments is still a challenge. This paper introduces a cognitively inspired decision-making method and an instantiation of it for (local) robot navigation in spatially constrained environments. We compare the method to two existing local planners with respect to efficiency, safety and legibility.

1 Motivation

Domestic robots are one of the major application scenarios for future robots. A prerequisite is the basic skill of efficiently and safely navigating in apartments in a manner that is understandable and predictable for the human inhabitants. Robot navigation has often been researched in museums, shopping malls or office environments with an emphasis on finding a path between two locations. But space in an apartment is much more limited. Path planning is required to move between rooms, but inside a room local navigation (also called navigation control) is the crucial factor. The situation can change quickly, obstacles may not be on the map, either because they move themselves (like people or pets or other household robots) or because they are movable (like chairs). Relying too much on path planning in dynamic environments leads to illegible[1] behavior [11, 14]. On the positive side, rooms are usually furnished in a way to make navigation easy for people and thus hardly contain dead ends.

In this paper we present a navigation control method that moves a robot efficiently, safely and legibly to a given coordinate and orientation in a (typical) room in a household. We first introduce the *Heuristic Problem Solver (HPS)*, a general, cognitively inspired decision-making framework, and show how to use it specifically for robot navigation.

We assume in this paper that at least one way of achieving legibility is to copy human movement patterns. Doing this can best be achieved by copying or at least approximating the way that humans make decisions and act. HPS uses concepts from cognitive science to implement a decision process similar to that of humans (but we do not claim that it models human decision-making in detail).

HPS generalizes the AI search paradigm by Newell and Simon [17], consequently navigation with HPS generalizes the Dynamic Window Approach for navigation [7, 4]. In the Dynamic

[1] Lichtenthäler et al. [14] define a robot's behavior as being legible ", if a human can predict the next actions of the robot and the robot behavior fulfills the expectations of a human interaction partner."

Window Approach the state space is first filtered to contain only those commands that do not lead to a collision in the near future, and then every state (i.e. navigation command and its expected effect) is evaluated with a weighted sum of three optimization aspects (we call those aspects *heuristics*). With HPS we extend this approach in two respects: 1) a generalization of the optimization function to contain an arbitrary set of heuristics, the composition of which can be adapted during the navigation task; 2) generation rather than filtering of the state space to make the decision procedure more efficient and the resulting behavior more legible.

We evaluate our approach in a simulated kitchen with a PR2 robot[2], comparing the navigation with HPS to a P-controller [19] and the Dynamic Window Approach [7]. We use a 3D simulation rather than a real robot to give us more freedom in the experiments.

This paper makes the following contributions: 1) it introduces the Heuristic Problem Solver as a general decision-making algorithm; 2) it proposes an instantiation of HPS with heuristics for navigation for use as a local planner, 3) it compares the navigation method in a simulated household environment with two state-of-the-art methods.

2 Approach

The Heuristic Problem Solver (HPS) generalizes AI search for asynchronous control tasks and incorporates knowledge from cognitive science to robustly handle dynamics and uncertainty. We explain the use of HPS for robot navigation, but we are convinced that the same basic approach can benefit any kind of decision-making process in autonomous systems, not just navigation.

2.1 Heuristics

The term "heuristic" has been used in many different contexts. In psychology and other cognitive sciences it is often used as a catch-term for "any behavior in animals or humans we cannot explain". In artificial intelligence a heuristic is usually defined as a function estimating the cost of getting from some state to a goal state.

Gigerenzer and Gaissmaier [8] propose the following definition:

> "A heuristic is a strategy that ignores part of the information, with the goal of making decisions more quickly, frugally, and/or accurately than more complex methods." [8]

This definitions encompasses the AI definition of heuristics, as they make decisions more quickly and frugally; being an estimation of the true costs they also ignore part of the information. But the definition goes further in claiming that heuristics can help to make better decisions than complex, "rational" methods.

The cause lies not in the method, but in the environment: "rational" decision methods are designed for "small" worlds — "[situations] in which all relevant alternatives, their consequences, and probabilities are known, and where the future is certain, so that the optimal solution to a problem can be determined" [8]. But in "large" worlds — and reality is a large world — less information or a simpler form of decision-making can lead to better information than when all available knowledge is considered.

In this paper, we use the following definitions:

> *A* heuristic *is an isolated piece of knowledge that is efficient to compute and relevant to a given problem. Heuristics may be problem-specific or general.*

[2] www.willowgarage.com/pages/pr2/overview

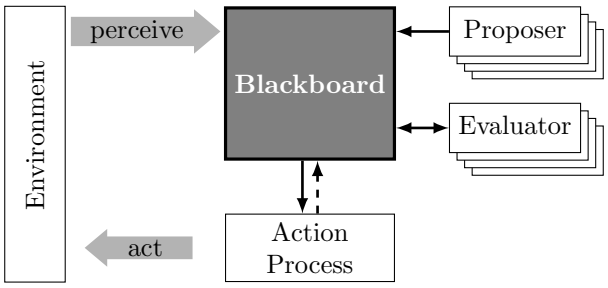


Figure 1: Architecture of the Heuristic Problem Solver.

An example of problem-specific a heuristics for navigation is "to reach a goal, you must decrease the distance between the current and goal position". A general heuristic that may be of use is "if consecutive commands differ a lot, any movement becomes jerky and illegible".

> *A* heuristic method *is an algorithm or architecture that combines heuristics to make adequate decisions in realistic environments.*

The Heuristic Problem Solver is a heuristic method.

> *An* expert *is a function or process that uses a heuristic to advance a problem solving task. HPS differentiates two types of experts:* Proposers *generate options for the problem solution or the next step (command) in the solution process;* evaluators *vote on the proposed options, they can also remove options and propose new ones.*

The example of the general heuristic can be used in a proposer by proposing the last issued command to be repeated as the next command. It can be used in an evaluator by preferring those proposed commands that are most similar to the last issued command.

2.2 Heuristic Problem Solver

The Heuristic Problem Solver (Figure 1) is a blackboard architecture [5]. The blackboard is a common memory used by expert processes to exchange information. *Action processes* monitor the "discussion" on the blackboard and at appropriate times pick the highest-rated command to execute it. The experts may contradict each other, for example one evaluator may prefer to turn the robot towards the goal, while another prefers it to turn along its movement trajectory.

We call the data objects on the blackboard *chunks*. The evaluators can have a weight between 0 and 1 and their votes are numbers between 0 and 1. The joint evaluation of a chunk is a weighted sum of the evaluators' weights and votes.

This architecture in principle enables a highly flexible, asynchronous decision procedure with intertwined decision-making on the blackboard and execution in several action processes. However, a full parallel execution of all the processes poses a challenge on debugging and is often not necessary.

For navigation, we use HPS in a synchronous, cyclic way. In each iteration, the action process chooses one command and executes it. An iteration can consist of several decision cycles, each of which goes through three steps: 1) consulting all proposers, 2) collecting votes from all evaluators, and 3) checking whether the evaluators agree enough to execute the highest-rated chunk. If after a decision cycle one chunk surpasses a fixed threshold (we use 0.4) and its joint evaluation surpasses the next-best chunk by more than a given percentage (we use 10%), the action process executes it right away. Otherwise another decision cycle starts: At

this point there may be chunks with incomplete evaluations on the blackboard (those proposed by evaluators in exchange for some chunk that the evaluator removed). First the evaluation of these chunks is completed. After that, those proposers able to produce more than one command (typically those involving randomness) are called again and another decision cycle starts. After a maximum number of decision cycles, the action process executes the highest-rated chunk regardless of its specific value or the distance to the second-best chunk.

Since evaluators may remove chunks (e.g. for safety reasons), it may happen that after the maximum number of decision cycles, there is no command on the blackboard. For this case, one can specify an emergency command, for navigation it stops the robot completely. If no applicable command is found after several iterations, HPS stops with a failure.

After each iteration, the composition and parametrization of proposers and evaluators may be changed.

2.3 HPS for Navigation

The PR2 robot has an omnidirectional drive. Our chunks consist of a navigation command (c_x, c_y, c_θ) with c_x: translation forward and backward, c_y: translation left and right, c_θ: rotation around robot z axis. We now introduce a set of experts for navigation.

Proposers Table 1[3] shows the proposers, ordered by their motivations: "Control" contains proposers that compute one specific command: the command of a P-controller, or the last issued command.

"Sampling" contains standard approaches to discretize from a state space (DWA DISCRETIZATION) or to randomly sample from the state space (RANDOM SAMPLING). A notable difference is that DWA DISCRETIZATION follows the scheme of the original Dynamic Window Approach that eliminates all commands that would lead to a collision in the near future, whereas RANDOM SAMPLING does not care about safety, it relies on evaluation experts to remove unsafe commands.

Another set of proposers is motivated by "Motion Primitives". These proposers generate the simple commands stopping, moving forward, backward and sideward, or turning on the spot. The magnitude is chosen randomly. Such simple movements are often used as recovery behavior when the robot gets stuck, but here we use them as suggestions for the next movement in normal operation.

We also experimented with "Command Modifications" (not shown in Table 1), generating new commands from existing ones, similar to mutation and crossover operations in genetic algorithms. We did not use them for the evaluation, because they result in similar behavior as random sampling (both combined with motion primitives).

Safety checks and evaluators To ensure safe navigation, the Dynamic Window Approach removes all unsafe commands from consideration. But as the evaluators in HPS can remove chunks, the other proposers ignore safety and we define two evaluation experts to check for safety, the same checks are used in the DWA DISCRETIZATION proposer[4]: using the robot's laser sensor covering 190 degrees of its front and sides, and a geometric collision check of the robot and tables or other objects that cannot be easily detected by the laser.

Both safety experts remove a command when it would cause a collision. The SAFETY-LASER replaces it by one with half the total translational velocity (the c_x and c_y components being

[3]The definitions of the symbols are explained in the table captions and in Figure 2.

[4]In the DWA version commands are only removed, not replaced as in the *HPSNAV* version.

Explanation of variables:
robot position: $\mathbf{r} = (x, y, \theta)$; predicted robot position: $\hat{\mathbf{r}} = (\hat{x}, \hat{y}, \hat{\theta})$; goal position: $\mathbf{g} = (x_g, y_g, \theta_g)$; robot velocity: $\mathbf{v} = (v_x, v_y, v_\theta)$; maximum translational velocity: $v_{\max} = \sqrt{v_{\text{xmax}}^2 + v_{\text{ymax}}^2}$; velocity control command: $\mathbf{c} = (c_x, c_y, c_\theta)$; maximum velocity command: $\mathbf{c}_{\max} = (c_{\text{xmax}}, c_{\text{ymax}}, c_{\theta\text{max}})$;

Name	Calculation	Parameters
	Control	
P CONTROLLER	$\Delta x = x_g - x$, $\Delta y = y_g - y$ $c_x = (\Delta x \cos(\theta) + \Delta y \sin(\theta)) \cdot \rho$ $c_y = (\Delta x \sin(\theta) + \Delta y \cos(\theta)) \cdot \rho$ $c_\theta = \lvert\theta_g - \theta\rvert \cdot \rho$ $\mathbf{c} = (c_x, c_y, c_\theta)$, scaled between minimum and maximum control value	$\rho = 1.2$
REPEAT LAST	$\mathbf{c} = (c_x^{i-1}, c_y^{i-1}, c_\theta^{i-1})$	
	Sampling	
DWA DISCRETIZATION	$S = \{(c_x, c_y, c_\theta) \lvert c_x$ from range$(0.0, 1.0,$ step $\nu_x)$, c_y from range$(-1.0, 1.0$step $\nu_y)$, c_θ from range$(-1.0, 1.0$step $\nu_\theta)\}$ Chunks = remove-unsafe-commands(S)	$\nu = (0.5, 0.2, 0.2)$
RANDOM SAMPLING	draw $S = \{(c_x, c_y, c_\theta) \lvert c_x = \text{rand}(-c_{\text{xmax}}, c_{\text{xmax}})$, $c_y = \text{rand}(-c_{\text{ymax}}, c_{\text{ymax}})$, $c_\theta = \text{rand}(-c_{\theta\text{max}}, c_{\theta\text{max}})\}$ $\lvert S \rvert = \eta$	$\eta = 20$
	Motion Primitives	
STOP	$\mathbf{c} = (0, 0, 0)$	
MOVE FORWARD	$\mathbf{c} = (\text{rand}(0, c_{\text{xmax}}), 0, 0)$	
MOVE BACKWARD	$\mathbf{c} = (\text{rand}(-c_{\text{xmax}}, 0), 0, 0)$	
MOVE LEFT	$\mathbf{c} = (0, \text{rand}(0, c_{\text{ymax}}), 0)$	
MOVE RIGHT	$\mathbf{c} = (0, \text{rand}(-c_{\text{ymax}}, 0), 0)$	
TURN LEFT	$\mathbf{c} = (0, 0, \text{rand}(0, c_{\theta\text{max}}))$	
TURN RIGHT	$\mathbf{c} = (0, 0, \text{rand}(-c_{\theta\text{max}}, 0))$	

Table 1: Proposers for navigation. The right column shows the parameters of each expert and the values we used in the evaluation. $\mathbf{c}$ is the command to be returned as a chunk.

reduced by the relative direction in which the collision would take place). This new command will be evaluated and checked for safety in the next decision cycle.

As domestic environments have little space to move, we used very small safety margins. Thus it can happen that even with the safety checks the Dynamic Window Approach or *HPSNAV* can slightly touch obstacles.

If a command is safe, the SAFETY-TABLES expert rates it with 1. The SAFETY-LASER expert selects the laser beam that is closest to the projected robot movement and returns the quotient of the length of the laser beam and the range of the laser, thus preferring directions with more free space.

All other evaluators are listed in Table 2, some of the angle definitions are explained in Figure 2. Many evaluators compare a chunk to some ideal value. For

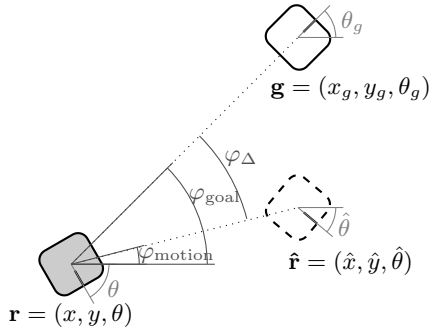


Figure 2: Angle definitions used by experts.

Name	Calculation	Parameters
P CONTROL	$\Delta x = x_g - x$, $\Delta y = y_g - y$ $c_{px} = (\Delta x \cos(\theta) + \Delta y \sin(\theta)) \cdot \rho$ $c_{py} = (\Delta x \sin(\theta) + \Delta y \cos(\theta)) \cdot \rho$ $c_{p\theta} = \lvert\theta_g - \theta\rvert \cdot \rho$ $\mathbf{c_p} = (c_{px}, c_{py}, c_{p\theta})$, scaled between minimum and maximum control value $e = \text{scale}(\text{dist}(\mathbf{c}, \mathbf{c_p}), 0, \lvert\mathbf{v_{max}}\rvert)$	$\rho = 1.2$
DWA ALIGN	$e = \text{scale}(\lvert\varphi_\Delta\rvert, 0, \alpha \cdot \pi)$	$\alpha = 0.5$
DWA VELOCITY	$e = \begin{cases} \frac{\lvert\mathbf{v}\rvert}{v_{\max}} & \text{if } \text{dist}(r,g) > \psi \\ 1 - \frac{\lvert\mathbf{v}\rvert}{v_{\max}} & \text{if } \text{dist}(r,g) \leq \psi \end{cases}$	$\psi = 0.5$
DWA GOAL REGION	$e = \begin{cases} 0 & \text{if } \text{dist}(r,g) > \psi \\ 1 & \text{if } \text{dist}(r,g) \leq \psi \end{cases}$	$\psi = 0.5$
STOP	$e = \text{scale}(\lvert\mathbf{c}\rvert, 0, \lvert\mathbf{v_{max}}\rvert)$	
GOAL DISTANCE	$e = \max(0, \text{scale}(\text{dist}(\hat{r}, g), 0, \alpha \cdot \text{dist}(r, g)))$	$\alpha = 1.5$
GOAL DIRECTION	$e = \max(0, \text{scale}(\lvert\theta_g - \theta_{\hat{r}}\rvert, 0, \alpha \cdot \lvert\theta_g - \theta_r\rvert))$	$\alpha = 1.5$
VELOCITY	$e = (\nu_t \frac{\sqrt{c_x^2 + c_y^2}}{v_{\max}} + \nu_r \left\lvert \frac{c_\theta}{v_{\theta\max}} \right\rvert) / (\nu_t + \nu_r)$	$\nu_t = 1, \nu_r = 0.5$
MOVEMENT DIRECTION	$e = \max(0, \text{scale}(\lvert\hat{r}_\theta - \varphi_{\text{motion}}\rvert, 0, \alpha \cdot \pi))$	$\alpha = 0.5$
LOOK AT GOAL	$e = \max(0, \text{scale}(\lvert\hat{r}_\theta - \varphi_{\text{goal}}\rvert, 0, \alpha \cdot \pi))$	$\alpha = 0.5$

Table 2: Evaluators for navigation. The right column shows the parameters of each expert and the values we used in the evaluation. $\mathbf{c}$ is the command in the chunk to be evaluated, e is the returned evaluation value. Explanation of other variables: see Table 1.

example, the P CONTROL evaluator calculates the same
control command as the P CONTROLLER proposer and would rate this as the best possible command with 1. The function *scale* maps the distance between a chunk and an ideal command to an output value between 0 and 1 using a linear interpolation; it takes three parameters: distance between given value and ideal, minimum distance (resulting in an output of 1), maximum distance (if the distance is greater than or equal to the maximum distance, the rating is 0).

Reconfiguration of experts The DWA VELOCITY and DWA GOAL REGION experts differentiate between situations close to and far from the goal. HPS supports a more general mechanism of changing the proposers and evaluators after each iteration. For navigation we only use the restricted distinction of cases as in the Dynamic Window Approach, but we implement it as a change of experts. The condition is the same as in the DWA experts: a radius of 0.5m from the goal differentiates between being close to or far from the goal.

3 Evaluation

3.1 Tested Algorithms

HPS can be instantiated in different ways to construct different navigation algorithms. We evaluate the following three (see Table 3):

PCONTROL implements a standard P-controller, modeled in HPS by one proposer and one evaluator that both compute the same command. It has no parameters.

DWA is the Dynamic Window Approach modeled in HPS by one proposer that suppresses

Algorithm	Proposers	Evaluators
PCONTROL	P CONTROLLER	P CONTROL
DWA	DWA DISCRETIZATION	DWA ALIGN (0.8)
		DWA VELOCITY (0.1)
		DWA GOAL REGION (0.1)
HPSNAV	STOP	SAFETY TABLES (0.3)
	MOVE-FORWARD	SAFETY LASER (0.3)
	MOVE-BACKWARD	far from goal:
	MOVE-LEFT	GOAL DISTANCE (1.0)
	MOVE-RIGHT	VELOCITY (0.5)
	TURN-LEFT	MOVEMENT DIRECTION (1.0)
	TURN-RIGHT	LOOK AT GOAL (1.0)
	REPEAT-COMMAND	DWA ALIGN (0.5)
	RANDOM SAMPLING	close to goal:
		STOP (1.0)
		GOAL DIRECTION (1.0)
		LOOK AT GOAL (1.0)
		P CONTROL (1.0)

Table 3: The algorithms used in the evaluation as a configuration of HPS. The numbers behind the evaluators give the used weights. The distinction between "close to goal" and "far from goal" is also implicitly made in two of the evaluation experts of the *DWA*.

unsafe commands as in the original Dynamic Window Approach algorithm and three evaluators, taken from [4]. The proposer discretizes the space of all safe commands. As discretization we follow the default configuration of the Dynamic Window Approach implementation in ROS[5], but with a cruder discretization for the rotations, as this worked just as well and we also evaluate the efficiency of the decision process. This generates a maximum of 300 commands (chunks), possibly less if some lead to collisions. The weights of the evaluators is taken from [7].

HPSNAV uses several proposers: all motion primitives (each generating one command per decision cycle), repeating the last command (one per iteration) and randomly sampling commands (20 per decision cycle). The proposers do not check for collisions, this is done by the safety evaluators. Pre-tests showed that pure random sampling is not good or needs a high number of samples. Motion primitives alone are also not enough, they should be combined with some additional randomization.

HPSNAV uses more evaluators than *DWA*, not only for safety, but also to account for other aspects of good navigation, especially the robot's orientation. The evaluators are reconfigured depending on whether the robot is closer or further than 0.5 m away from the goal. This distinction may look like a "hack", but it could also be regarded as a special case of reconfiguring the experts according to any situation. Doing this manually is cumbersome, but if adjustments were learned from observations, it could make the procedure more powerful. It turned out in pre-tests that the distinction of being close to or far from the goal improves the navigation behavior significantly and it is also used implicitly in two of the *DWA* evaluators.

We did not specifically optimize the weights of the evaluators. The parametrizations in Table 3 worked well in our tests, small variations in them lead to similar results.

[5]ROS: Robot Operating System, coming with packages for standard tasks. The package `dwa_local_planner` (`http://wiki.ros.org/dwa_local_planner`) is part of the ROS navigation stack (`http://wiki.ros.org/navigation`).

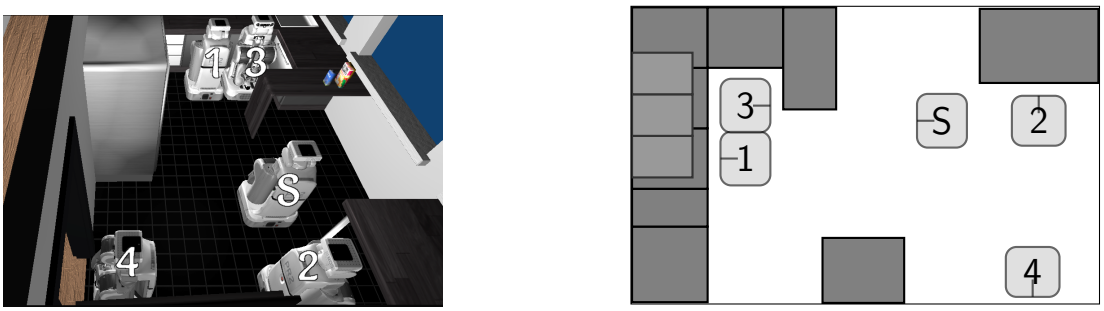


Figure 3: Course through kitchen, shown in simulator (left) and map (right). The tasks were to go from the starting pose (S) to a cupboard (goal 1), from the cupboard to the table (goal 2), from the table to the dishwasher (goal 3) and from the dishwasher to the door (goal 4) with the orientations as shown.

3.2 Experimental Setup

We defined a course of four navigation goals — defined by position and orientation — through a simulated kitchen (Figure 3). The single navigation tasks consisted of going from the start point to goal 1, going from goal 1 to goal 2, and so on. A goal was registered as being achieved if the robot was less than 20 cm away from the goal point and achieved its goal rotation within 0.3 rad (17°). A timeout of 30 seconds ended a run if the goal was not achieved, but this did not happen in the evaluated runs. None of the evaluators in the *DWA* configuration considers the robot's orientation and we did not include any additional control to turn the robot at the end. The goal orientation was nevertheless reached in all our trials.

We used the 3D simulator Morse [13], which provides realistic physics and an adjustable degree of realism regarding perception. As we were only interested in the action selection, we used ground truth perception and a noiseless laser sensor.

Every algorithm was run three times for each goal to account for variations that come from randomizations in the experts and uncertainties inherent even in a simulation.

Our goal was to evaluate efficiency, safety and legibility of the movements. To this end we recorded the robot's position at every iteration as well as the state of the blackboard and afterwards calculated the following measurements:

- **duration:** the time to reach the goal, for measuring navigation efficiency, which probably also benefits legibility to some extent [15]. This duration includes some startup time of about 5 seconds to register the goal; this applies to all three algorithms.
- **chunks:** the median number of chunks on the blackboard at the time of the decision. For *PCONTROL* this value is always one, for *DWA* it is 300 minus unsafe commands, for *HPSNAV* it depends on the number of decision cycles and the reduction of chunks by the safety evaluators; the maximum is 132. This measure indicates the efficiency with which decisions are made. If the robot's only task is to navigate, this measure is not too relevant with today's hardware. But if many different control decisions have to be taken, it can be relevant.
- **collisions:** the percentage of iterations in which the robot base overlapped with furniture or a wall in the environment, calculated in 2D. In our simulation we switched off the physics for the furniture to allow the robot to move through objects, so we could observe the whole trajectory without the robot going to pieces. The collisions are calculated afterwards based

on a 2D map. This leads to small inaccuracies, for instance it would be possible for the robot to move below the kitchen bar without an actual collision. But those miscalculations are rare and apply to all tested algorithms alike.

- **side-/backward:** the percentage of iterations in which the robot moved sidewards ($|v_y| > |v_x|$) or backwards ($v_x < 0$). This is at least one important criterion for the legibility of the movement. Robots with omnidirectional drives have the advantage of a high variety of movements, but they tend to overuse this freedom. People hardly ever move side- or backwards, so it looks strange. More importantly, the rotation of the robot can indicate its intended goal or path and thus help people to anticipate its movements.

3.3 Data

Figure 4 shows the data of the just explained measurements for each individual run. The goal assignments are indicated by different symbols and the averages of the three runs per algorithm are connected by lines. Figures 5(a) and 5(b) depict one individual trajectory per algorithms for reaching goals 2 and 3, showing the robot's pose at every fifth interval and its overall trajectory. We show these two tasks as they were the most difficult ones. A video at https://vimeo.com/170125350 shows the three runs for goal 3.

Both *PCONTROL* and *DWA* move mostly sidewards to reach the goals, while *HPSNAV* turns or moves in an arc, looking into its movement direction, then moves towards the goal and there corrects the remaining rotation. This strategy was not explicitly programmed into the algorithm, it emerged from the combination of the experts that try to steer the robot to the goal point, but at the same time take care to look towards the goal or in the movement direction.

We are not the first to note the strong sideward movement of the Dynamic Window Approach. The ROS implementation has a parameter "alignment costs". But we have found no publication that mentions such a parameter and the implementation seems to treat the alignment as an additional measure outside the core Dynamic Window Approach.

3.4 Discussion

In our experiment, all three algorithms reach the goals fast and reliably. For *DWA* and *HPSNAV* we had pre-tests with different configurations of proposers that failed to move from the cupboard to the table (cp. Figure 5(a)). Those configurations did not manage to turn the robot away from the cupboard to move towards the table (cp. Figure 5(a)). The robot then jiggled around the starting point. It is thus important to ensure that enough commands are proposed.

The chunks on the blackboard at the time of decision are always one for *PCONTROL* and somewhere below 300 for *DWA*, depending on the obstacles present. The only algorithm that had a "choice" was *HPSNAV* by the number of decision cycles. The maximum number of chunks it could generate in 5 decision cycles was 132, diminished by unsafe commands. For all runs the median number of decision cycles was 5, the average lies between 4 and 5, depending on the task. So *HPSNAV* usually uses all its available decision cycles, but there are some clear situations in which it can save time. One may argue that it is unfair to prescribe 300 chunks to *DWA* and only allow a maximum of 132 to *HPSNAV*. Previous tests showed that *DWA* needs a higher number of chunks to perform well and we wanted to use a configuration in which it moved nicely. *HPSNAV* also needs a lot more chunks if it is only run with the random sampling proposer, so the use of "motion primitives" indeed reduces the number of necessary chunks.

A P-controller is often used to precisely position a robot in spatially restricted environments, but it does not take into account obstacles. It is therefore no surprise that it produces the highest

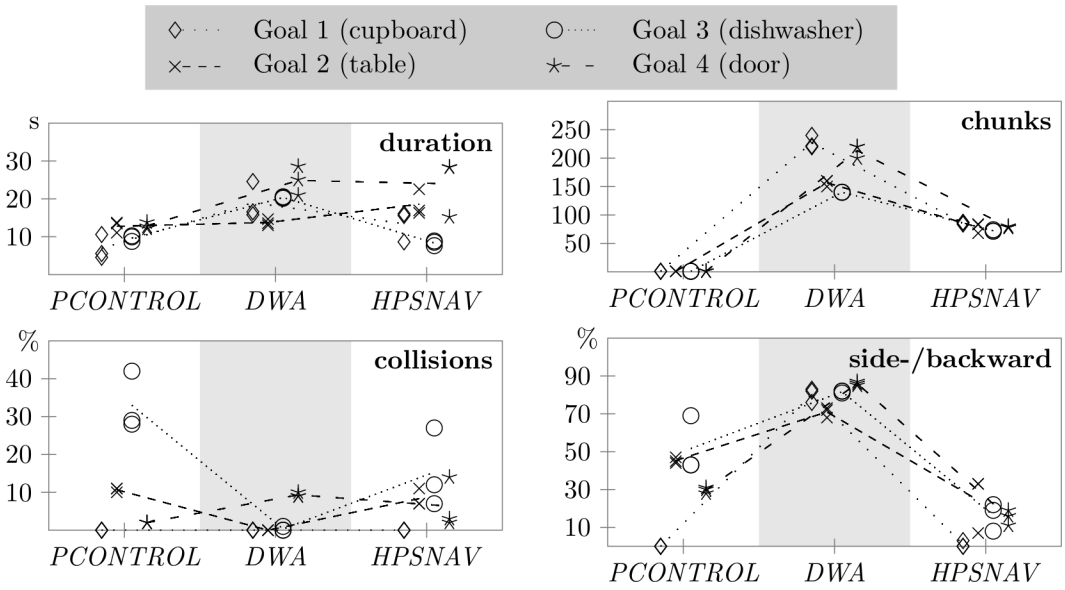


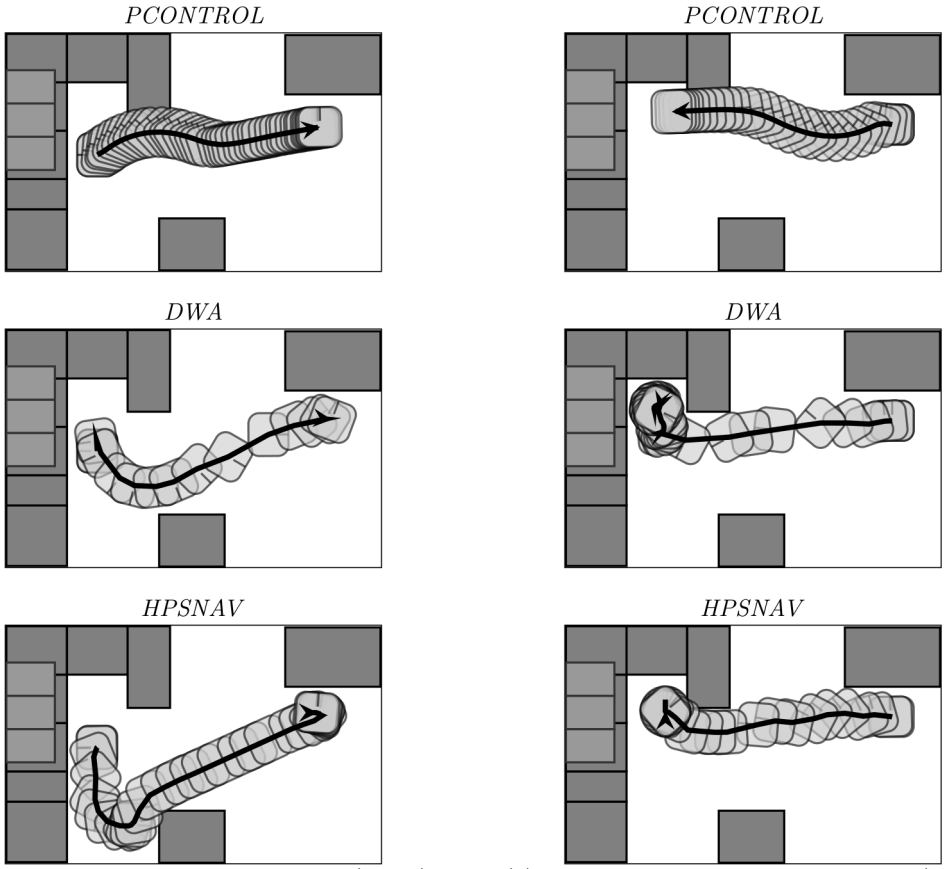
Figure 4: Performance data: The graphs show the data of each trial, the lines connect the averages from each algorithm.

number of collisions. The other two are comparable with respect to collisions. Going to goal 3, *HPSNAV* produces significantly more collisions than *DWA*, but as can be seen in Figure 5(b), the robot only slightly touches the furniture. With higher safety margins this could be resolved and the detected collisions may not have been real collisions, as they are calculated in 2D and the robot base would fit under the kitchen bar.

With respect to the robot orientation, *HPSNAV* is clearly better than *PCONTROL* or *DWA*, as can be seen both in the data in Figure 4 and the trajectories in Figure 5. Neither P-controller nor Dynamic Window Approach explicitly take into account the robot's rotation except for the goal rotation in the end. For *DWA* we have to add that the recorded trajectories come with some randomness in the choice of the command; using only the three evaluators, the joint evaluations for many chunks on the blackboard are identical or close together. The decision therefore also depends on which of the chunks happens to be in front after sorting the chunks.

HPSNAV generalizes the Dynamic Window Approach in two ways: 1) by proposing commands rather than filtering from all available commands and 2) by using more evaluators. Our evaluation shows that both are beneficial: proposing relevant commands (by using motion primitives and repeating the last command together with some randomized proposers or modifiers) reduces the number of chunks that need to be evaluated. Newell and Simon [17] described the trade-off between putting effort in choosing possible next states and the effort of evaluating those states. But in most search algorithms, states are at most filtered by rough criteria such as safety and most of the effort lies in evaluating them. *HPSNAV* shows at least for navigation that it pays off to explicitly consider the choice of states.

More evaluators lead to a better mix of aspects that are relevant for reaching the goal efficiently, safely and legibly. One concern could be the parametrization of many evaluators. In the case of navigation we had good results without particular optimization of the composition and weights of the evaluators. For other tasks this may be different.



(a) Trajectories from cupboard to table (goal 2). (b) Trajectories from table to dishwasher (goal 3).

Figure 5: Sample trajectories for goals 2 and 3.

We do not claim that our experts are the best and only ones for navigation. More important is the insight that a generalized AI search paradigm and heuristics can lead to behavior that is not only safe and efficient, but also legible for people. We also made the simplifying assumption that the robot has access to noisless, ground-truth perceptions. On a real robot, this is of course not true. Based on the findings in cognitive science that heuristic decision-making is advantageous in large worlds, we expect that the heuristic approach of HPS can cope well with uncertain perceptions and dynamic environments. But this has to be shown in future work.

4 Related Work

Nature provides many examples of well-adapted behavior that is generated by simple heuristic rules. Herbort and Butz [9] show for a grasping task that people's decisions can neither be explained by an optimization method, nor by a simple rule-based method. Instead, a heuristic combination of both methods can best explain the observed behavior.

In the domain of navigation and collision avoidance, Moussaïd et al. [16] have suggested a heuristic model to explain navigation in crowds. Different to most crowd models that are

usually based on physical models, Moussaïd et al. model each individual with heuristics for navigation decisions, ensuring both the arrival at the person's goal position and a collision-free path. Huber et al. [10] explored human collision avoidance behavior in crossing situations at different angles. The behavior of all situations cannot be explained by one single optimization technique. Rather, the behavior seems to result from a combination of different mechanisms.

We are only aware of one work in the area of AI where navigation has been implemented in a heuristic framework. Epstein [6] implements what she calls "pragmatic" navigation in her FORR architecture, which provides a heuristic decision-making mechanism, including decision rules as well as shallow search strategies. Epstein defines her navigation problems in a discrete, deterministic grid world with unknown, but static obstacles.

In the last decade it has become obvious that robot navigation in proximity of people needs special attention. In this context, several aspects have been explored, such as following a person, approaching a person, moving in crowds or moving in formations. Most approaches to human-aware navigation follow the classical two-stage approach with a global path planner and a local controller [12]. For example, Sisbot et al. [20] enhance a global A* path planner with human-aware cost functions for safety, comfort and visibility. Ohki et al. [18] consider personal space for their path plan together with a prediction of the human movement. Prediction of human paths is also a basic component in the work of Bennewitz et al. [3]. They learn motion patterns of office workers and use those for robot navigation.

HPSNAV is restricted to the role of a local planner. As we have argued in the introduction, a local planner should be able to navigate the robot safely and reliably through a normal room that contains no navigation traps; navigation between rooms would definitely need a planner. In the field of human-aware navigation, local planning has mostly focused on how to predict the movement of people and thus ensure their safety [1, 21]. In this paper, we have not even included a person, but even in static environments an observer should be able to anticipate and appreciate a robot's movements. To our knowledge the aspect of legibility has so far only been considered for global planners, not for the controller.

We have evaluated *HPSNAV* with objective measures in static environments. It remains to be tested in dynamic environments, which allow for a much higher variance of situations. Typical navigation algorithms such as the ones implemented in ROS behave different in crossing situations than people [11, 2] and are not really legible [14].

5 Conclusion

We have shown that a generalization of the Dynamic Window Approach makes robot navigation more efficient (regarding the computing resources) and more legible, while moving efficiently and safely. The approach builds on the original AI search paradigm proposed by Newell and Simon [17], generating prospective commands rather than filtering them. Both the generation and evaluation are done by heuristics.

We are convinced that this approach is valuable to many other kinds of decisions in autonomous systems, from control problems such as grasping to abstract object-related decisions like when to bring an object to a person. The Heuristic Problem Solver is an implementation of this basic concept, enabling the robot to make parallel decisions for different modalities by using several action processes.

Acknowledgements

With the support of the Bavarian Academy of Sciences and Humanities.

References

[1] D. Althoff, D. Wollherr, and M. Buss. "Safety assessment of trajectories for navigation in uncertain and dynamic environments". In: *ICRA, IEEE*. 2011.

[2] Patrizia Basili et al. "Strategies of locomotor collision avoidance". In: *Gait & Posture* 37.3 (2013), pp. 385–390.

[3] M. Bennewitz, Wolfram Burgard, G. Cielniak, and S. Thrun. "Learning Motion Patterns of People for Compliant Robot Motion". In: *International Journal of Robotics Research* 24.1 (2005).

[4] Oliver Brock and Oussama Khatib. "High-Speed Navigation Using the Global Dynamic Window Approach". In: *IEEE International Conference on Robotics and Automation*. 1999, pp. 341–346.

[5] Robert Engelmore and Tony Morgan, eds. *Blackboard Systems*. Addison-Wesley Publishing Company, 1988.

[6] Susan L. Epstein. "Pragmatic navigation: Reactivity, heuristics, and search". In: *Artificial Intelligence* 100 (1998), pp. 275–322.

[7] D. Fox, W. Burgard, and S. Thrun. "The dynamic window approach to collision avoidance". In: *Robotics & Automation Magazine, IEEE* 4.1 (1997), pp. 23–33.

[8] Gerd Gigerenzer and Wolfgang Gaissmaier. "Heuristic Decision Making". In: *Annual Review of Psychology* 62 (2011), pp. 451–482.

[9] Oliver Herbort and Martin V. Butz. "The continuous end-state comfort effect: weighted integration of multiple biases". In: *Psychological Research* 76.3 (May 2012), pp. 345–363.

[10] Markus Huber et al. "Adjustments of Speed and Path when Avoiding Collisions with Another Pedestrian". In: *PLoS ONE* 9.2 (Feb. 2014).

[11] Thibault Kruse, Patrizia Basili, Stefan Glasauer, and Alexandra Kirsch. "Legible robot navigation in the proximity of moving humans". In: *Advanced Robotics and its Social Impacts (ARSO), 2012 IEEE Workshop on*. IEEE, 2012, pp. 83–88.

[12] Thibault Kruse, Amit Kumar Pandey, Rachid Alami, and Alexandra Kirsch. "Human-aware robot navigation: A survey". In: *Robotics and Autonomous Systems* 61.12 (2013), pp. 1726–1743.

[13] S. Lemaignan et al. "Human-Robot Interaction in the MORSE Simulator". In: *Proceedings of the 2012 Human-Robot Interaction Conference (late breaking report)*. 2012.

[14] C. Lichtenthäler, T. Lorenz, and A. Kirsch. "Influence of legibility on perceived safety in a virtual human-robot path crossing task". In: *RO-MAN, 2012 IEEE*. Sept. 2012, pp. 676–681.

[15] David V. Lu and William D. Smart. "Towards more efficient navigation for robots and humans". In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2013.

[16] Mehdi Moussaïd, Dirk Helbing, and Guy Theraulaz. "How simple rules determine pedestrian behavior and crowd disasters". In: *Proceedings of the National Academy of Sciences of the United States of America* 108.17 (2011), pp. 6884–6888.

[17] A. Newell and H. Simon. *Human Problem Solving*. Upper Saddle River, New Jersey: Prentice Hall, 1972.

[18] Takeshi Ohki, Keiji Nagatani, and Kazuya Yoshida. "Collision avoidance method for mobile robot considering motion and personal spaces of evacuees". In: *Intelligent Robots and Systems (IROS)*. 2010.

[19] David Sellers. "An overview of proportional plus integral plus derivative control and suggestions for its successful application and implementation". In: *Proceedings for the 2001 International Conference on Enhanced Building Operations*. 2001.

[20] Emrah Akin Sisbot, Luis F. Marin-Urias, Rachid Alami, and Thierry Simeon. "A Human Aware Mobile Robot Motion Planner". In: *IEEE Transactions on Robotics* 23 (2007), pp. 874–883.

[21] P. Trautman and A. Krause. "Unfreezing the robot: Navigation in dense, interacting crowds". In: *Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on*. Oct. 2010.