We are IntechOpen, the world's leading publisher of Open Access books Built by scientists, for scientists

4,800
Open access books available

122,000
International authors and editors

135M
Downloads

Our authors are among the

154
Countries delivered to

TOP 1%
most cited scientists

12.2%
Contributors from top 500 universities

CLARIVATE ANALYTICS BOOK CITATION INDEX INDEXED

WEB OF SCIENCE™

Selection of our books indexed in the Book Citation Index in Web of Science™ Core Collection (BKCI)

Interested in publishing with us?
Contact book.department@intechopen.com

Numbers displayed above are based on latest data collected.
For more information visit www.intechopen.com

29

Real-Time Support of Haptic Interaction by Means of Sampling-Based Path Planning

Michael Strolz and Martin Buss
Technische Universität München
Germany

1. Abstract

Haptic feedback enables the support of a human during the interaction with an environment. A variety of concepts have been developed to achieve an effective haptic support of the user in specific scenarios, e.g. *Virtual Fixtures*. However, most of these methods do not enable an adaptive support of the motion from a user within a (real or virtual) environment, which would be desirable in many situations. Especially when dynamical obstacles are involved or when the desired motion of the human is not known beforehand, an online computation of this support is essential, which should be based on a fast and effective determination of feasible motions.
In contrast to other methods, sampling-based path planning is applicable to arbitrary interaction scenarios and enables to find a solution if it exists at all. Thus, it seems to be ideally suited for a generic framework that is able to deal with various kinematics, as e.g. a virtual prototyping test bed for the haptic evaluation of mechanisms requires. At such a test bed, the path planner could directly be coupled to the haptic rendering of a virtual scene to assist a user in approaching a target.
This motivated the development of SamPP, a sampling-based path planning library with implementations of the most important algorithms. It can be used for nearly arbitrary rigid robots and environments. By performing numerous benchmarks, we prove the effectiveness and efficiency of SamPP. It is shown that a single-threaded version of the path planning can be used for real-time support of the haptic interaction at a novel actuated car door.
Furthermore, we enhance the path planning performance for unknown or dynamical environments significantly by the *OR-Parallelization* of different path planning queries. This *Generalized OR-Parallelization* is a novel concept that to the best knowledge of the authors has not been proposed beforehand. We show that for the case of dynamic environments the likelihood of a fast path planning result is higher with our approach. Thus, even in dynamic or unknown environments, a real-time support of haptic interaction can be achieved. Finally, we highlight four promising research directions to exploit the concept of Generalized OR-Parallelization:
1) Combination of PRMs and RRTs to achieve a synergy of the advantages of both concepts, 2) concurrent use of different parameter sets of path planning algorithms, 3) online adaptation of these parameter sets and 4) online adaptation of the types and numbers of parallel executed path planning programs.

2. Introduction

2.1 Support of Haptic Interaction

Haptic feedback enables the support of a human during the interaction with a virtual, shared, and/or remote environment. This is desirable for a broad range of applications, where the limited capabilities of humans should be improved or at least should not be detrimental. Examples include training of students and employees, manipulation of complex mechanisms, robotic surgery, and teleoperation in general (Esen, 2007).

A variety of concepts have been developed to achieve an effective haptic support of the user in specific scenarios. The most important one is the concept of *Virtual Fixtures*. Initially it has been proposed as a static, *rail-like* support to reduce the DOF of human motion (Rosenberg, 1993). Since then, a lot of extensions and variations of this concept have been developed, some of which even provide a dynamic, situation-dependent haptic support (Abott et al., 2003; Ammi & Ferreira, 2007; Davies et al., 1997; Kapoor et al., 2007; Lynch et al., 2002).

However, most of these methods do not enable an adaptive support of the interaction between a user and a haptic device. Rather, based on prior knowledge, they rely on a pre-computation of parts of the support method. One example is the offline determination of a desired path for surgical tools based on MRI data, where the desired start and goal configuration of the tools are known in advance (Li et al., 2007).

2.2 Path Planning

Especially when (real or virtual) environments with dynamical obstacles are involved or when the desired motion of the human is not known beforehand, an online computation of a collision-free path is essential. Path planning has been an active field in the past 15 years, and a variety of methods have been proposed.

A major drawback of many path planning algorithms is their lack of generality in terms of e.g. the existence of local minima. For instance, artificial potential fields which have been proposed for the assistance of haptic manipulation in the nano-scale (Varol et al., 2006) are prone to produce local minima in many handling scenarios.

With the recent introduction of sampling-based methods (Kavraki et al., 1996; LaValle et al., 1998), these limitations have been overcome, and high-dimensional path planning problems have been solved efficiently. Generally in sampling-based path planning, the geometry of both robot and workspace is considered based on discrete samples of the configuration of a robot. Various methodologies exist for the creation of these samples, which greatly influence the properties of the path planner depending on a given scenario. For the generated samples, a collision check is performed, often by openly available collision detection libraries as e.g. evaluated in (Strolz et al., 2008). The result is subsequently used by a path planning algorithm, which exclusively works in the configuration space (C-space) of the robot. To find a path for the robot, a local planner has to check whether two samples can be quasi-continuously connected without a collision.

Based on this, different strategies exist to find a path in the C-space: While *Single-Query* Planners as e.g. Rapidly-exploring Random Trees (RRT) (LaValle et al., 1998) create a path specifically for a given start and goal configuration, *Multi-Query Planners* as e.g. Probabilistic Roadmaps (PRM) (Kavraki et al., 1996) proceed in two steps: In the processing step, a number of samples is connected to form a *roadmap*. In the query step, the given start and goal configurations have to be connected to the roadmap. If this succeeds and if the

roadmap is connected, a solution surely exists and a suitable, optimized path can be found by a graph search. A comprehensive overview of the state of art in sampling-based path planning is given in (LaValle, 2006).

2.3 Performance Enhancement by Parallelized Sampling-Based Path Planning

It has been noted that some sampling-based path planning algorithms are (at least partially) *embarrassingly parallel* (Arnato & Dale, 1999). This means, that the path planning time can be drastically reduced by implementing the algorithm in a parallel manner and running it on suitable hardware. Impressive demonstrations of this are given in (Challou, 1995; Plaku, 2005), where a nearly linear speedup for an increasing number of processors has been reported.

Recently, it has been shown that another way of speeding up sampling-based path planning is to run a number of path planning queries in parallel on a suitable hardware (Klasing, 2009). It has been pointed out that with this *OR paradigm* the probability that none of the n queries finds a solution within a predefined time **t** is given by

$$1 - P_n(t) = (1 - P_1(t))^n \tag{1}$$

where P_1 denotes the probability that one query finds a solution within t (Challou, 1995). Obviously, this probability decreases rapidly with an increasing number of queries.

However, as noted in (Calisi, 2008), due to variations in the path planning environment there may be no single planner that will perform well in all possible situations. It is noted that "*an ideal motion planner would be a meta-planner using a suite of more specialized planners to cooperatively solve a motion planning problem. It would automatically apply the best-suited planner for each distinct region in the planning space and would produce regional solutions that could be composed to solve the overall motion planning instance*." Furthermore, not only the choice of the algorithm itself but also its parameterization is a critical issue, because it drastically affects the performance of the path planning.

2.4 Contributions

Until the recent release of OpenRAVE (Diankov, 2008; OpenRAVE [Online]), there has been no easy-to-use, powerful generic path planning software. Thus, the sampling-based path planning library SamPP has been developed (Dömel, 2007) and subsequently improved, enhanced and evaluated. At first, we give an overview of SamPP and introduce the three most fundamental things about it: The representation of the robot and the environment, and the path planning algorithms. For the evaluation of SamPP, firstly it is successfully applied to a path planning scenario involving a robot with 10 DOF. Secondly, it is applied to different car door kinematics with 2 DOF, where it shows a superior performance (typical maximum path planning duration < 20 ms). Thirdly, the influence of the parameterization of the path planning algorithms is discussed and recommendations for the parameter tuning are given. A comparison with the state of the art, open-source sampling-based path planning libraries OpenRAVE and PP reveals that the performance of SamPP indeed is very high.

As explained in (Strolz et al., 2008), car doors with more than one DOF promise to improve the convenience of the access to cars. However, experiments on a Virtual Reality (VR) test

bed showed that users did not find the operation of car doors with several to be intuitive. We show that the single-threaded application of SamPP can be used for real-time support of the haptic interaction at such novel actuated car doors. The effectiveness of this haptic support with respect to user convenience is validated experimentally.
Furthermore, we enhance the path planning performance for unknown or dynamical environments significantly by an OR parallelization of ***different*** path planning algorithms. This is a novel concept that to the best knowledge of the authors has not been proposed beforehand. We prove that the likelihood of at least one fast path planning result is dramatically higher with our approached than with the parallel execution of several instances of a single path planner with a fixed parameter set. The reason is that the performance of a sampling-based path planner not only is subject to a stochastically process, but also depends greatly on the scenario at hand and the chosen parameters.

3. SamPP, a New Sampling-Based Path Planning Library

In this chapter, we introduce SamPP, a generic software library for Sampling-based Path Planning for rigid robots within arbitrary environments (Dömel, 2007). After describing the overall structure of the software and the representation of robot and environment, we present the implemented algorithms and an evaluation. Finally, we compare the performance to two recently released open source software libraries for sampling-based path planning, OpenRAVE (Diankov, 2008; OpenRAVE [Online]) and PP (PP [Online]).

3.1 Concept and Structure

SamPP has been intended to be part of a robotics control framework. Therefore, it was written in C++ (cross-platform) in an object-oriented manner as an API which is to be used in a client program. Using a specific parameter file which describes the path planning task at hand, a *SAMPP* object is instantiated there, and the path planning is executed.
A path planning task is fully defined by the following information:

- Description of the robot
- Description of the environment of the robot
- Start and goal configuration of the path

For solving the task, an algorithm and its parameterization have to be chosen. The algorithm needs a 3D representation of the path planning scenario to perform collision checking, which is built based on the description of robot and environment. Accordingly, we structured the software architecture in the following components:

1. ROBOT: Parametrized description of the kinematics and the 3D shape of the robot
2. ENVIRONMENT: Parametrized description of the 3D shape of all potential obstacles within the path planning scenario
3. WORLD_COLLISION and WORLD_VISUALIZATION: Structure representing the 3D scenario for the collision checking and the visualization engine
4. PLANNER: Selection and parametrization of the path planning algorithm
5. SAMPP: Path planning object based on previous five components

This way, a path planning query is performed by instatiating a SAMPP object, which will execute a path planning based on a chosen parameter set.

3.2 Representation of the Robot

The central consideration of sampling-based path planning is to find a collision free path for the robot in the configuration space (C-space) of the robot. The C-space depends on the specific kinematics of the robot, i.e. the number, the type and the limitations of its joints.
The motion of every rigid kinematics can be described by a combination of rotational and translational DOF. Even if a mechanism would exhibit different joints, e.g. a non-prismatic translational one, the position and orientation of all links could be expressed using additional *virtual* rotational and/or translational joints. This also holds for parallel links, where fewer overall DOF exist than the single joints would exhibit in sum.
To give a better understanding of the problems involved with creating a scheme for the representation of arbitrary robots, lets consider the robotic application of a car door with the two DOF ($\mathbf{q}_1$, $\mathbf{q}_2$) depicted in Figure 1. While the parts A and DOOR form an open tree-structured kinematic, due to the parallel mechanism the motion of the car door parts B and Z depend on ($\mathbf{q}_1$, $\mathbf{q}_2$). Thus, besides expressing rotational and translational DOFs and their limitations, we need to express the potential dependencies inherent to parallel mechanisms.

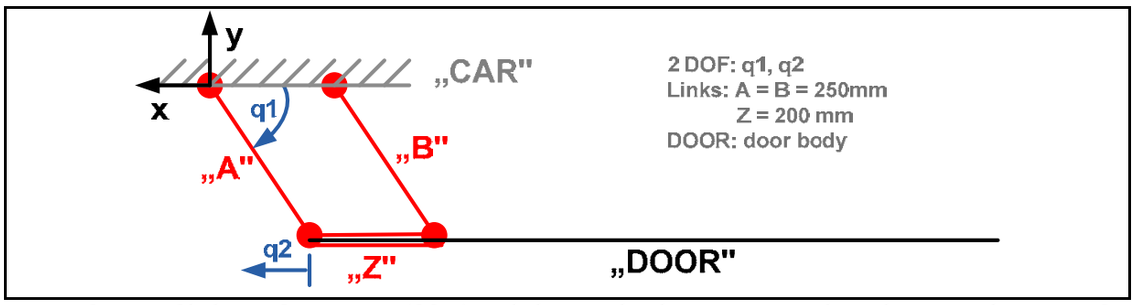

Fig. 1. Swing-Sliding Car Door (SSD) exhibiting both a rotational and a translation DOF, parallel links, and a closed kinematic chain.

By introducing dependent joints (dependent variables/dependent DOF), the problem of nonlinear and parallel kinematic configurations can be solved. For many real-world applications, a simple solution where the dependent joint configuration is calculated from the linear interpolation of predefined lookup table data is sufficient. This requires, that for every dependent joint a file has to be created that stores the lookup table.
A common way to represent kinematic chains is to use the Denavit-Hartenberg (DH) notation, where subsequently frames are created that describe a transformation from the base to the single links of the robot. This concept can be extended by introducing a parent for every frame, such that more than one child frame can be related to a frame. Thus, we build a tree structure, where each path represents a DH-like series of frames, which in combination with the dependent joints allows for representing parallel kinematics. This results in an intuitive tree-like robot description that can handle not only arbitrary open kinematic chains, but also kinematics with simple closed chains.
In robotics, it is often convenient to divide the configuration space in the configuration of the (usually) movable robot base on the one hand and the configuration of the joints of the manipulator on the other hand. The base configuration can generically be desribed by a transformation in respect to a world coordinate frame (maximum number of DOFs: 6). This equals a sequence of three translational and three rotational joints, all of which are

independent of each other. Thus, the base configuration can be expressed by a subset of the generic joint parameters.
Our generic definition of joint parameters is briefly described in the following:

- *ID*: Unique number of robot part (foundation of the overall kinematic tree)
- *PARENT*: *ID* of the predecessing robot part in the kinematic tree
- *REFJOINT*: [only if joint is of type dependent:] *ID* of reference joint
- *TYPE*: description of the DOF of the robot part
    - *RIGID*: the robot part represents no DOF
    - *TRANS*: translational DOF
    - *TRANS_CSTR*: constrained translational DOF
    - *TRANS_DPT*: no DOF, translational motion depends on *REFJOINT*
    - *ROT*: unconstrained rotational DOF
    - *ROT_CSTR*: constrained rotational DOF
    - *ROT_DPT*: no DOF, rotational motion depends on *REFJOINT*
- *MIN*: [only if joint is of type *_CSTR*:] minimum value of DOF
- *MAX*: [only if joint is of type *_CSTR*:] maximum value of DOF
- *DH_A*: Denavit-Hartenberg parameter (translation along x-axis)
- *DH_D*: Denavit-Hartenberg parameter (translation along z-axis)
- *DH_ALPHA*: Denavit-Hartenberg parameter (rotation around x-axis)
- *DH_THETA*: Denavit-Hartenberg parameter (rotation around z-axis)
- *WEIGHT*: weighting factor for this DOF

Accordingly, the base parameters are the subset without ID labels, joint type _DPT, and the Denavit-Hartenberg parameters.
Beside the kinematics, it is necessary to define parameters that take into account the influence of the single joints on the overall robot to ensure an efficient and effective path planning. The path planner needs to rate motions of the robot. A common measure for this is the change of the kinematic configuration of the robot, which results from the single joint displacements. In serial kinematic structures, a displacement of a joint near to the base usually has a significantly bigger impact on the displacement of the overall robot. This motivates to introduce a cost function that punishes big displacements, thereby allowing the path planner to work efficiently. Right now, we use a simple constant weighting WEIGHT of the individual joints to achieve this goal. The weighting has to be chosen heuristically based on the scenario at hand.
Furthermore, a discretization has to be defined for the single DOFs, because the path planning algorithm works in a discrete space, while the environment is continuous. The discretization $\mathbf{\Delta q}$ defines a lower bound which enables planning that can be considered quasi-continuously.
The distance of two states is calculated based on the simple L1 metric, thus all joint deviations are multiplied with their respective weight and summed, such that $\sum \mathbf{WEIGHT}(\mathbf{q}_k - \mathbf{q}_{k-1})$ results. If this sum is smaller than the threshold $\sum \mathbf{WEIGHT}(\mathbf{\Delta q})$ (minimum cost), no collision is considered to be feasible when moving from state k-1 to state k, and thus no collision check is performed. This means that, if the discretization is to coarse, the calculated path may not be collision-free. However, if the discretization is very fine, the efficiency of the path planning is significantly reduced. Thus, it is crucial to provide a parametrization that is appropriate for the example at hand.

For enabling collision checks and visualization, besides the kinematics a graphical 3D representation of the single robot parts is essential. Attention has to be paid to the handling of transformations and 3D data: The collision detection library and the visualization library may exhibit a different way to handle files and transformations.

3.3 Representation of the Environment

In contrast to the usually rigid and fixed robot kinematics, the environment may have to be altered during runtime because of moving obstacles. If obstacles are detected by sensors, they are often handled without semantic knowledge, i.e. shape primitives are used to describe a convex hull over their respective 3D geometry, compare e.g. (Strolz et al., 2008). Based on the assumption that only such shape primitives would be important in the modification of the environment during runtime, we defined data structures and accordingly transformation matrices for them. Thus, the environment objects can be altered, removed, or new ones can be inserted during runtime. Sometimes, it is more efficient to transform the objects rather than to create a new one.

3.4 Rapidly-exploring random Tree (RRT) Algorithms

Rapidly-exploring random Trees (RRTs, originally proposed in (LaValle, 1998)) are the most popular *Single Query* planning algorithms. A particularly successful modification is the bidirectional RRT: Instead of growing a tree from the start to the goal configuration, two trees are grown towards each other (Kuffner & LaValle, 2000).

There exist several expansion strategies for growing the tree. In the "classical" approach, the RRT is grown exactly one discretization step towards a sample. This can be extended by defining an upper bound for the expansion, e.g. five discretization steps. In contrast, the "visibility" approach iterates discretization steps towards the sample as long as the sample is not reached and no collision has been detected. We implemented these strategies, denoted as *RRT-cla* and *RRT-vis* in the following. Both algorithms require the start and the goal state as inputs. An optional parameter *timeout* enables to define a maximum duration for the path planning to quit the path planning for overly complex or even unsolvable problems. For an efficient handling of the search for nearest neighbors, we used kd-Trees and the open-source library ANN (Mount & Arya, 1997; ANN, [Online]).

3.5 Probabilistic Roadmap (PRM) Algorithms and their Parameterizations

PRM algorithms exhibit two phases: The processsing step, where the probabilistic roadmap is built, and the query step, which consists of connecting the start and the goal state to the roadmap and a consecutive search for the optimal path between them. Analogous to the RRT implementations, we implemented algorithms with "classical" and "visibility" expansion strategies: *PRM-cla* and *PRM-vis*.

The goal of building the map (*processing*) is to get a roadmap that provides a good coverage of the C-space. This can be heaviliy influenced by several parameters, which will be introduced in the following.

PrmMaxConnDist: Maximum distance between two states. This parameter defines to which extent the path planner behaves *classical* (low values, near resolution) or *visibility-based* (high values) while building the map.

As mentioned in the RRT section, the efficacy strongly depends on the environment at hand. Thus, with this parameter the path planning can be tuned for a class of scenarios. For instance, when rather dense environments have to be considered, a low value would be a good choice.

PrmInitialStates: Number of initial states which are randomly sampled before the algorithm tries to connect them. If all initial states are connected (or if the timeout condition is triggered), the building of the map is stopped.

The higher PrmInitialStates, the higher is the probability of a dense roadmap. In turn, this makes it more likely that the start and goal states can be connected with the roadmap in the query stage. However, a high number leads to a more complex roadmap which inhibits the path search. One way of finding a good setting can be to start with a rather low value. If it turns out that the start and/or goal state cannot be connected with the map, this number can adaptively be increased. Note that this functionality would have to be provided by the client program.

PrmMaxConnNumb: Maximum number of connections between a new collision free state and the roadmap. The higher this value, the better is the conjunction of the map, which tends to result in a smoother path and a longer path query.

In many applications, a value of up to 10 lead to good results.

PrmCleanMapRate: Rate of deletion of non-connected states. If a state is near an obstacle, it may be very difficult to connect it to the roadmap, which slows down the connection of the map. Therefore, the non-connected states are deleted with this rate. If this value is inappropriately low, it may be very difficult to build a map in a dense environment, because relevant states are deleted before being connected to the roadmap. Thus, this parameter be better only used if a relatively free C-space is assumed.

PrmMapRateExam: Rate of examination of the number of roadmaps. If the C-space is divided by obstacles such that not all states can be connected to one roadmap, some kind of timeout has to stop the attempt of the (infeasable) connection of the different roadmaps. This is done by examining the number of roadmaps at a constant rate. If the number did not change within one time interval, it is assumed that the different roadmaps can not be connected, and the proc. stage is terminated.

PrmMaxSampNumb: Maximum number of random samples. If this value is set, the proc. stage is terminated after this number of samples has been reached.

After the roadmap has been built, path planning *queries* can be executed. This involves firstly to connect the start and the goal state to the (same) roadmap. If this succeeded, one is sure that the states are connected, and a graph search as e.g. the famous A* algorithm can be performed to find the optimal path. Else, no path can be found, and an error is returned. Furthermore, a timeout results in an error, too.

4. Benchmark Results for SamPP

In the following, several benchmarks for SamPP are introduced and discussed. For all programs and scenarios, a PC with AMD Athlon(tm) 64 X2 Dual Core 5200+, 2 GB RAM and the operating system Linux Release 2.6.22-ipipe KDE 3.5.7 has been used. Only one of the two cores was used, and all programs were run 20 times.

In the results, **min** and **max** denote the minimum and maximum, and **σ** and **E** the standard deviation and expectation values of the path planning duration and the path length, respectively, of the 20 runs.

4.1 Application to a Robot with 10 DOF

Sampling-based path planning is superior to other path planning techniques especially if the number of DOF is high. To evaluate the efficiency of SamPP with respect to this, a scenario involving ViSHaRD10 (Ueberle et al., 2004) has been developed. ViSHaRD10 is a robot with 10 rotational DOF, as shown in Figure 2. Its special kinematic configuration does not allow a direct DH transformation from one joint to another for the joints 5, 8 and 9. Thus, for each of this joint two additional rigid joints have been used such that the robot kinematics could be described. As VRML model of these *pseudo joints*, very small cubes have been used, which are completely surrounded by neighboring joint models. Thus, they have neither influence on the path planning nor on the visualization.

The single joints of ViSHaRD10 are constrained by the wiring. We considered a restriction to **[-1.2 π 1.2π]** as appropriate to avoid damages, and applied this to every joint description. Furthermore, we had to find a suitable weighting for the joints. We did this for every single joint by using the maximum absolut worst-case displacement of all robot parts caused by a movement of this joint. These displacements were further used define the resolution for each joint.

The robot exhibits the highest versatility in the horizontal plane. Thus, a path planning scenario involving lots of motions in this plane was assumed to be most difficult, as it can constrain a high number of joints. For the evaluation, we used two scenarios.

Scenario 1 consisted of two narrow, parallel walls around the robot. The path planning task is to move the fully extended robot ($\mathbf{q}_{\{1..10\}} = \mathbf{0}$) to the opposite, fully stretched configuration ($\mathbf{q}_1 = \pi$, $\mathbf{q}_{\{2..10\}} = \mathbf{0}$).

Scenario 2 is an extension of scenario 1, where one additional short wall is placed exactly in the middle of the other two walls. It is shown in Figure 2 (r.). The start configuration is given by ($\mathbf{q}_1 = \pi$, $\mathbf{q}_2 = -\pi$, $\mathbf{q}_{\{3..10\}} = \mathbf{0}$), the goal configuration by ($\mathbf{q}_1 = -\pi$, $\mathbf{q}_2 = \pi$, $\mathbf{q}_{\{3..10\}} = \mathbf{0}$).

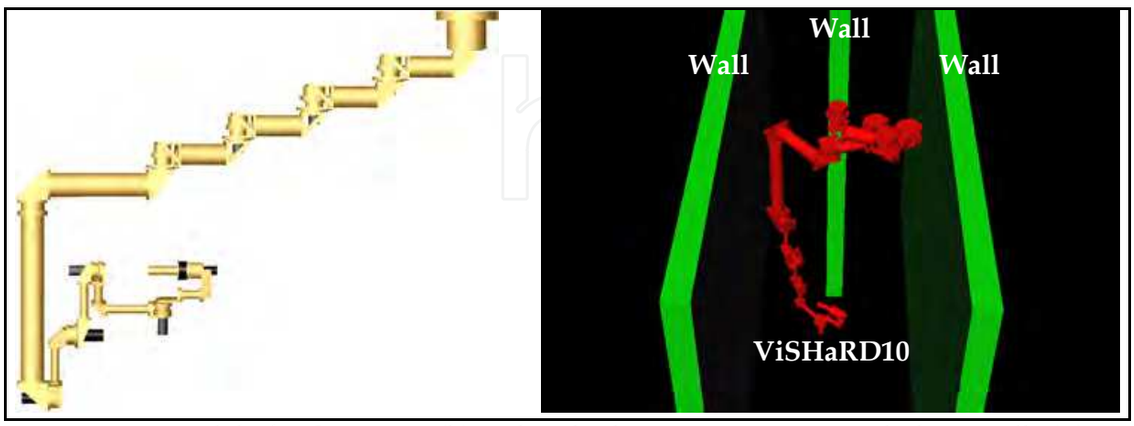

Fig. 2. VRML model of ViSHaRD10 (l.) and path planning scenario 2 with three walls (r.).

Due to the high number of dimensions, PRM algorithms are not appropriate for a fast single-shot query, because a good coverage of the C-space would require a very large

number of states, such that the graph search on the roadmap would be much slower than a single query method. Thus, we only consider the two RRT algorithms *RRT-cla* and *RRT-vis* for these scenarios. The benchmark results are given in Table 1.

		Algorithm	min	max	σ	E
Scenario 1 (two walls)	Duration [s]	RRT-cla	0.471	6.435	1.475	2.618
		RRT-vis	0.086	1.309	0.422	0.387
	Path length [NORM]	RRT-cla	65.0	97.0	9.4	77.2
		RRT-vis	112.0	360.0	61.4	218.4
Scenario 2 (three walls)	Duration [s]	RRT-cla	2.099	7.438	1.127	3.537
		RRT-vis	3.026	43.079	11.659	17.125
	Path length [NORM]	RRT-cla	92.0	188.0	35.1	125.9
		RRT-vis	164.0	432.0	62.0	290.9

Table 1. Path planning benchmark results for the two ViSHaRD10 scenarios 1 and 2.

The environment of the first scenario with two walls is not very narrow in joint space. Therefore, *RRT-vis* outperforms *RRT-cla* in the duration measures by a factor of approximately 3 to 6. The differences in the normalized path lengths clearly exhibit that despite the postprocessing of the path the faster *RRT-vis* produced costs whose average was three times higher than the *RRT-cla*. This shows one dilemma of sampling-based path planning: By choosing an appropriate algorithm and by tuning parameters, a trade-off has to be found for the scenario at hand.
In the second scenario, the third wall leads to a very narrow area in the C-space. This limits the advantage of the *RRT-vis*, and consequently leads to a rather slow path planning when compared to *RRT-cla*. Again, the *RRT-vis* produces a much shorter path. For such an environment, the classic method is the best option.
Thus, by applying SamPP to a robot with 10 DOF, we have shown that the implementations *RRT-vis* and *RRT-cla* are able to plan a path in a relatively short time. In two complex scenarios, the *RRT-cla* exhibited a maximum planning time of 7.4 s. Furthermore, it found relatively short paths when compared to the visibility based method. This has also been visually observed when executing the planned path on the robot.

4.2 Preliminary Remarks on the Application to Different Scenarios with Car Doors

We apply SamPP to some car doors with 2 DOF and investigate the effect of different environments etc. As models for the car door a VRML file with 31728 polygons has been used, the obstacles were represented as approximated spheres with 400 polygons each.
The goal of the path planning is to provide a collision free path from a fully closed position to a given open position. The following methods are investigated:

- *RRT-vis*: visibility-based RRT implementation
- *RRT-cla*: classic RRT implementation
- *PRM-vis-P*: proc. stage of PRM-vis
- *PRM-vis-Q*: query stage of PRM-vis
- *PRM-cla-5P*, *PRM-cla-10P*: proc. stage of PRM-cla with 5/10 nearest neighbours
- *PRM-cla-5Q*, *PRM-cla-10Q*: query stage of PRM-cla with 5/10 nearest neighbours

4.3 Application to a Double-Four-Links Car Door (2 DOF)

In **scenario 3**, a car door with two serial links named Double-Four-Links Door is considered. Its kinematics is depicted in Figure 3 (r.). Though exhibiting four links and six joints, it only has two rotatory DOFs. Furthermore, due to the symmetry of the links, the door performs no rotation in world coordinates.

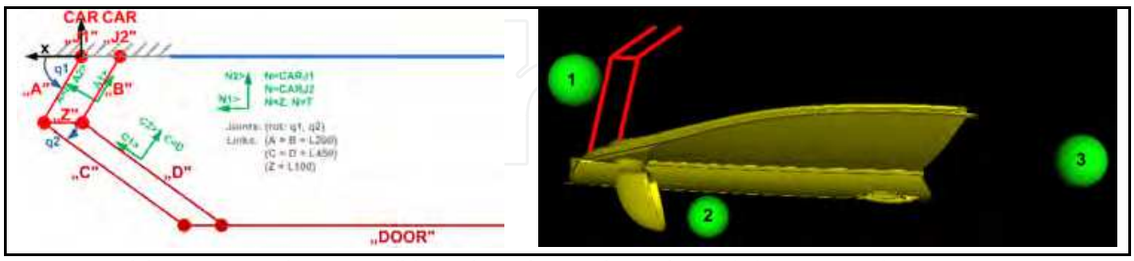

Fig. 3. Double-Four-Links Door (l.) within three obstacles (r.) (scenario 3).

We consider three different environments which consist of three spheres as is shown in Figure 3 (l.). The configuration space constrained by the environment is depicted in Figure 4 (m.). The C-space consists of 3 non-connected areas. As both the start and the goal state are located in area 2, a path can be found. Area 1 represents sphere 2 and, in combination with area 3, forms a narrow corridor. This surely is the bottleneck for the path-planning. If sphere 2 is varied only a little bit (**Δy=0.01m** nearer to the car, which has a length of **l=1.30m**), the corridor significantly narrows. In contrast, if sphere 2 is varied a little bit more (**Δy=0.10m** further away from the car), it is out of the workspace of the door and thus has no influence on the path-planning, see Figure 4 (l.). Area 2 is now a very large free space, and path planning should accordingly be very fast. This example illustrates how extremely small variations in the configuration of the obstacles can affect path planning.

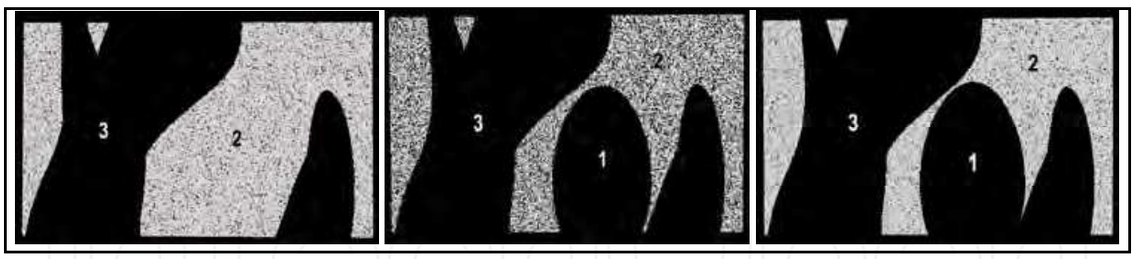

Fig. 4. Scenario 3: Broad (l.), narrow (m.) and very narrow (r.) configurations in the C-space given by slightly varying the position of obstacle 2 (see also Figure 3).

For all configurations of sphere 2 ("very narrow", "narrow", and "broad"), all path planning methods have been evaluated. The results are summarized in Table 2.

For configuration "very narrow", *RRT-cla* performs best. The PRM methods are considerably slower in the processing stage, but excel in the variations *PRM-cla-10Q* and *PRM-vis* in the query stage. If many queries are to be performed on such a kind of environment, PRM seem to be a good choice.

Interestingly, *PRM-cla-10* is faster then *PRM-cla-5* and *PRM-vis*. The reason for this must be that choosing 5 nearest neighbors leads to a roadmap which is too dense, while PRM-vis is to coarse. Thus, for every environment there is a range of connection length for which the

planning performs best. In this particular case, by chance we found a good balance, as both a higher and a lower value perform worse.
With respect to the length (cost) of the paths, there is no great difference between the planners for all three scenarios, see the example given in Table 2, scenario 3 “very narrow”.
From the results for configuration "narrow", one can see that the RRT methods give a similar expectancy value, while exhibiting a significantly different variance. The reason is, that the *RRT-vis* sometimes "by chance" quickly finds a path through the narrow passage, but besides that works less efficient in such a kind of scenario. In contrast, from the PRM methods the *PRM-vis* performs best. This is due to the funnel-shaped C-space; if this was maze like, the results most likely would have been much worse.
While there has been a strong improvement in the time duration, the path lengths seem not to significantly differ from the ones of the "very narrow" ones.

		Algorithm	min	max	σ	E
Scenario 3 (broad)	Duration [ms]	RRT-cla	20	31	3	24
		RRT-vis	3	9	2	5
		PRM-cla-5P	22	48	8	35
		PRM-cla-5Q	7	17	3	11
		PRM-cla-10P	25	49	7	34
		PRM-cla-10Q	6	18	3	10
		PRM-vis-P	41	118	21	67
		PRM-vis-Q	4	13	2	6
Scenario 3 (narrow)	Duration [ms]	RRT-cla	33	49	4	38
		RRT-vis	13	102	21	35
		PRM-cla-5P	198	396	54	266
		PRM-cla-5Q	17	48	8	25
		PRM-cla-10P	117	160	31	231
		PRM-cla-10Q	7	15	3	22
		PRM-vis-P	86	187	26	120
		PRM-vis-Q	4	17	4	9
Scenario 3 (very narr.)	Duration [ms]	RRT-cla	31	66	8	44
		RRT-vis	21	375	95	115
		PRM-cla-5P	314	517	80	404
		PRM-cla-5Q	27	60	24	114
		PRM-cla-10P	240	382	40	275
		PRM-cla-10Q	14	38	6	23
		PRM-vis-P	374	548	102	448
		PRM-vis-Q	10	23	4	16
	Path length [NORM]	RRT-cla	39	42	0.9	40.5
		RRT-vis	39	46	2.0	41.1
		PRM-cla-5P	39	43	1.1	40.3
		PRM-cla-10P	39	43	1.1	40.5
		PRM-vis-P	39	42	0.8	40.3

Table 2. Path planning benchmark results for scenario 3 with variation of obstacle position.

4.4 Application to SCARA-like Car Door (2 DOF)

In scenario 4, SamPP has to be applied to the Two-Links Door which is depicted in Figure 5. The environment consists of four spheres. The main problem in doing this is to circumvent sphere 2 and to reach the state which is near the spheres 3 and 4. The C-space of this path planning problem is very narrow, as can be seen from Figure 5 (r.). In area 1 both the start and the goal configuration is contained, thus a valid path can be found. The representation of sphere 2 forms a long and narrow passage from the start state.

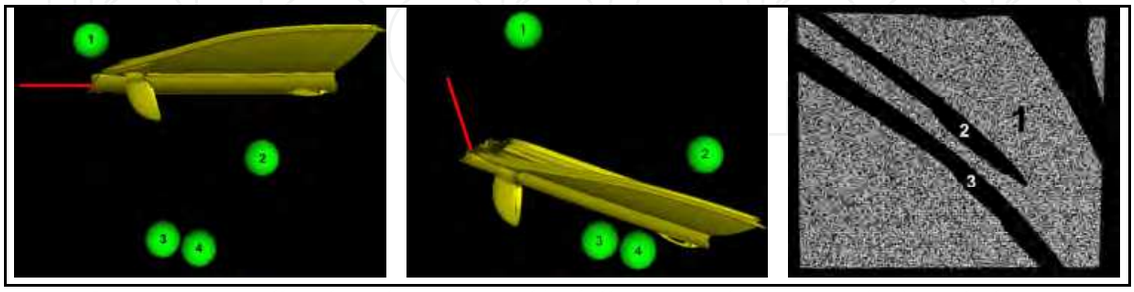

Fig. 5. Scenario 4: Fully closed position (l.), fully opened position (m.) and depiction of narrow passage in the C-space of the Two-Links Car Door.

		Algorithm	min	max	σ	E
Scenario 4 (very narr.)	Duration [ms]	RRT-cla	11	43	8	24
		RRT-vis	3	19	5	9
		PRM-cla-5P	75	168	23	103
		PRM-cla-5Q	6	17	3	11
		PRM-cla-10P	87	169	29	130
		PRM-cla-10Q	6	22	4	11
		PRM-vis-P	91	169	22	127
		PRM-vis-Q	4	19	4	9
	Path length [NORM]	RRT-cla	19	44	7.5	36.8
		RRT-vis	36	62	7.2	54.0
		PRM-cla-5P	39	47	2.2	42.3
		PRM-cla-10P	37	47	2.3	41.8
		PRM-vis-P	32	62	6.4	46.0

Table 3. Path planning benchmark results for scenario 4.

The RRT methods perform the path planning considerably faster than the PRM methods. The *RRT-vis* exhibits an expectation value of 9 ms, thereby even undercutting the expectation value of the PRM queries. If the corridor in the C-space would not have been straight but curved, the *PRM-cla* would have been better. All PRM methods require a maximum of more than 150 ms for building the map. This makes them not suited for real-time applications in scenarios like these.

The path lengths exhibit a significant variance for all methods, which is a hint that the path postprocessing performs very poor for scenarios like these. Thus, it might be beneficial to improve this algorithm.

4.5 Application to Car Doors with 2 DOF in the Presence of Many Obstacles

When interfacing the path planner with a sensor system (Strolz et al. 2009), a much higher number of primitive objects will be used to represent obstacles in the workspace of the door. This motivated to evaluate the influence of the number of obstacles on the path planner. We replaced the spheres of the environment (which represented vertical pillars) by 100 spheres each. This increase in the number of obstacles does barely affect the C-space.

From Table 4, it clearly can be seen that the RRT methods provide a much better performance than the PRMs for a single query. The reason is their reduce demand for collision checks: The PRMs suffer from the many collision queries that have to be performed when building the map. However, the maximum query time of the PRMs is significantly shorter than that of the *RRT-vis*. Thus, it is not possible to give a clear recommendation on whether to use PRMs or RRTs in a scenario with a high number of obstacles. In static scenarios, a combination might be a good choice: Two computers can be used, one running *PRM-vis*, the other *RRT-vis*. While the roadmap is built, only *RRT-vis* results are used for path planning. After that, as long as the environment does not change, both *RRT-vis* and a PRM-query a started simultaneously, and the faster result is used. For the evaluation scenarios, this would lead to a maximum time consumption for the "parallel query" of 68 ms, which might be fast enough to be used in an haptic assistance task.

		Algorithm	min	max	σ	E
Modified Scenario 3 (400 obst.)	Duration [ms]	RRT-vis	39	548	117	190
		PRM-vis-P	142	2084	382	1704
		PRM-vis-Q	16	66	11	41
Modified Scenario 4 (400 obst.)	Duration [ms]	RRT-vis	20	117	21	42
		PRM-vis-P	2497	2926	106	2643
		PRM-vis-Q	31	68	8	41

Table 4. Path planning benchmark results for modified scenarios 3 and 4 with 400 obstacles.

4.6 Short Performance Comparison to OpenRAVE

We wanted to find out whether our implementation of sampling-based path planning algorithms had a performance that is comparable to implementations of other researchers. Recently, the professional, open-source path planning library OpenRAVE (Diankov, 2008) has been released. Its RRT algorithms seemed to be suitable to benchmark our implementations of *RRT-cla* and *RRT-vis*.

At first, we installed OpenRAVE on the same Linux system that had been used for the evaluation of SamPP. We run the same scenarios which we described in the previous sections. The performance was really poor when compared to SamPP: All time measures were by approximately an order of magnitude worse than the ones for SamPP. For instance, the average time of the bidirectional RRT was 32.04 s (>> 0.39 s of our *RRT-vis*) for scenario 1 and 146 ms (>> 9 ms of our *RRT-vis*) for scenario 4. We could not explain this discrepancy, so we installed OpenRAVE on a virtual Linux system (Ubuntu) which was running on a Windows system (Windows XP, 2 GB RAM) and repeated the evaluation.

Despite the fact that the virtual Linux most likely increases the computational overhead, the results were much closer to the ones of SamPP. For instance, the average and minimum times of the bidirectional RRT was 2.45 s/0.53 s (> 0.39 s/0.09 s of our *RRT-vis*) for scenario 1 and 12 ms/5 ms (> 9 ms/3 ms of our *RRT-vis*) for scenario 4.

While these comparisons do not enable a fair overall judgement of the path planning performance (different system configuration, heavily dependence on specific scenario), they nonetheless lead to the following conclusions:

1. We were not able to identify the reason for the poor performance of OpenRAVE on the first system. Thus, we advice potential users of OpenRAVE or other complex path planning libraries to benchmark the software on different systems to minimize the risk of running it in a very suboptimal configuration.
2. SamPP is comparable to professional state-of-the-art implementations of samping-based path planning algorithms, as e.g. OpenRAVE or PP.

4.7 Remarks and Summary

We evaluated the performance of SamPP for executing path planning for a 10 DOF robot and for different 2 DOF car doors within an (in terms of the configuration space) very demanding environment. Due to the RRT and PRM algorithms, SamPP is able to solve a variety of path planning problems efficiently. For the case of 300 to 400 obstacles, nearly "worst-case" placed in the workspace of these car doors, we found typical mean values for the path planning time in the area of 50 ms for RRTs, 1500 ms for building a PRM and 30 ms for PRM queries. The evaluation results for scenarios 3 and 4 show that the performance of SamPP indeed is sufficient for the haptic real-time assistance of a human in various scenarios with 2 DOF. Independently of the planning algorithm, the path postprocessing seems to work quite well if there are no overly narrow passages in the C-space of the robot.
Note that the performance heavily depends on the environment at hand. The environments that we used for the evaluation often exhibited an uncluttered, rather free C-space. This promotes the visibility based methods. However, it has been shown that there is no "one size fits all" solution: depending on the environment at hand, variations of the parameter setting may decrease or increase the performance of the path planner.
Further, we observed that a comparison of the performance of PRM methods for fixed processing times showed that larger roadmap leads to longer query response time, and that a reduction of the number of initial states proved to give better results for our scenario. It is relatively hard to find an appropriate number of initial sample states for simple environments of the robot. The roadmap has to sufficiently cover the C-space to provide a very high probability that the start and the end goal can be connected to the map. A large and complex roadmap, in turn, cannot quickly be evaluated by a graph search algorithm. This problem cannot occur when using an RRT method, because the planner is focused on connecting a start configuration as efficiently as possible with the goal configuration, such that no "overly complex" connection structure results. For rather simple scenarios, the total planning time of *RRT-cla* is faster than a query on a roadmap. For such cases, it does make no sense to use PRMs at all.

5. Haptic User Support at a Virtual Car Door by Path Planning

5.1 System Description

In (Strolz et al., 2008), a system for the control of actuated car doors with arbitrary DOF has been introduced. This system should be augmented with an additional user support method given by an online path planning. An overview of the overall structure of the simulated system is given in Figure 6. The different modules are connected by UDP communication.

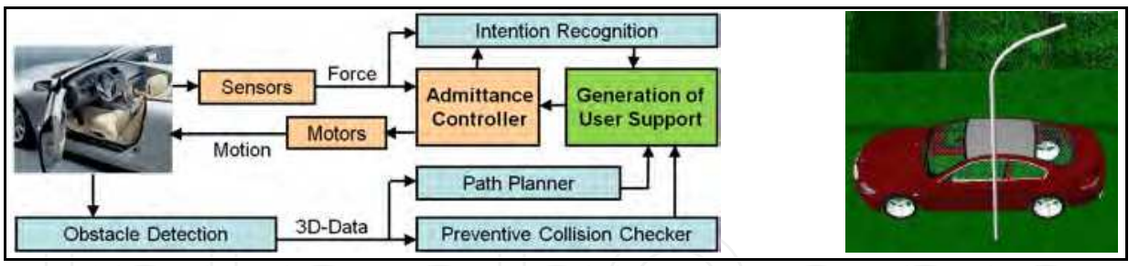

Fig. 6. Advanced car door control system with haptic user assistance by path planning, collision avoidance and intention recognition (l.) and its visual simulation (r.).

To achieve a precise path planning, a camera which monitors the workspace of the door and provides data about potential obstacles is simulated. The simulated data is continuously being sent to the path planning computer (in the form of primitive, convex shapes, e.g. spheres). Furthermore, the path planner continuously receives the start and goal configuration of the door from the door controller. For each new data packet, a path planning query trigger event occurs. As soon as the path planner finished a query and sent the collision-free path to the car door controller, it accepts such trigger events to restart path planning with the updated values.
In the car door controller, in the joint space a supportive force is calculated which points into the direction of the middle piece of the collision-free path. We chose an upper bound of 2 N for this bound, such that it predominantly does not change the motion of the mechanism itself, but rather gives motion cues to the user to achieve an intuitive interaction.

5.2 Experiment

To evaluate the effect of the haptic user assistance, an experimental user study has been conducted. We chose car door and obstacle configuration similar to scenario 4, see Figure 5.
Our hypotheses were:

1. Users can handle the door easier and more intuitively if the door is actuated nad supportive forces are displayed to them.
2. The path planning support is helpful during the haptic interaction.

We designed the experiment such that different controller configurations were displayed, some of which included the path planning. By answering a questionaire, the participants should rate these configurations with respect to a reference scenario without path planning. The duration of the experiment was approximately 30 minutes, and 20 people (12 men; in average 26 years, 70 kg, 1.75 m) participated in it.

5.3 Results and Discussion

In Figure 7, some of the results are displayed. They show a predominant approval of the implemented car door control system with path planning. A T-test revealed that the rating of the two variations of the path planner assistance (with and without end positioning support) was significant on a 5% level ($F(0.95; 38) = 2.09$, $p = 0.017 < 0.05$) and ($F(0.95; 38) = 2.09$, $p = 0.0004 < 0.05$)). Thus, the path planner indeed brings a significant advantage to users when they handle a novel car door.

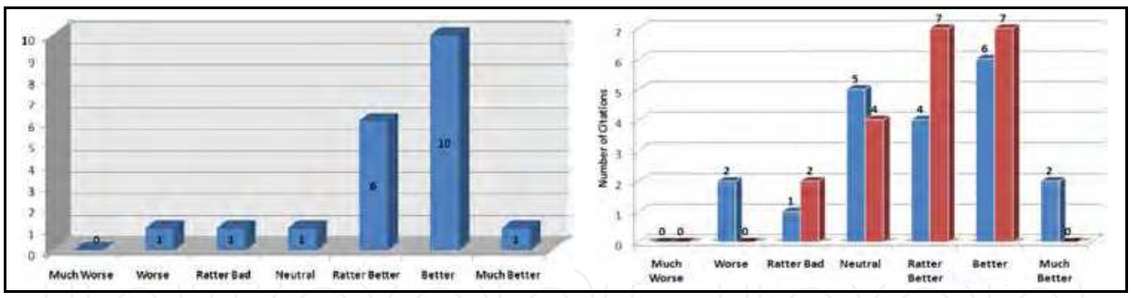

Fig. 7. Evaluation of the advanced car door control system: Comparison against reference scenario for the assistance in general (l.) and for two variations of the path planning assistance (r.) where the red bars represent an additional haptic support (Solhjoo, 2009).

6. Further Enhancement: Parallel Execution of Different Path Planners

6.1 Problem: There is no Best Algorithm

In the introduction and the evaluation section, it was highlighted that there is no overall best-performing path planning algorithm, because the kinematics of the robot and the structure of the environment have a huge impact on the level of difficulty of the path planning task. To clarify this, in Table 5 a composition of the fastest planners is given for slight modifications of scenario 3.

		Algorithm	min	max	σ	E
Scenario 3 (broad)	Duration [ms]	RRT-cla	20	31	3	24
		RRT-vis	3	9	2	5
		PRM-vis-P	41	118	21	67
		PRM-vis-Q	4	13	2	6
Scenario 3 (narrow)	Duration [ms]	RRT-cla	33	49	4	38
		RRT-vis	13	102	21	35
		PRM-vis-P	86	187	26	120
		PRM-vis-Q	4	17	4	9
Scenario 3 (very narr.)	Duration [ms]	RRT-cla	31	66	8	44
		RRT-vis	21	375	95	115
		PRM-vis-P	374	548	102	448
		PRM-vis-Q	10	23	4	16
Modified Scenario 3 (400 obst.)	Duration [ms]	RRT-vis	39	548	117	190
		PRM-vis-P	142	2084	382	1704
		PRM-vis-Q	16	66	11	41

Table 5. Composition of the fastest planners for modifications of scenario 3.

6.2 Solution: Parallelization of Different Algorithms (Generalized OR paradigm)

As already explained in the introduction, two research directions have been proposed in the past to speed up complex path planning problems:

1. Parallelization of subtasks of path planning algorithm:
   Decreasing the time consumption of specific path planning algorithms:
2. OR-parallelization of a specific path planning algorithm:
   Increasing likelihood of a fast result by executing several instances of one planner

We propose a promising third alternative:

3. OR-parallelization of different path planning algorithms:
   Increasing likelihood of a fast result by executing a number of instances of different planners and/or planner parametrizations

To prove this principle mathematically, we extend Equ. (1) (Challou, 1995) to the **Generalized OR paradigm**: Be $P_{1,2,..,k}(t)$ the probability that the different path planning programs 1, 2, ..., k do not find a collision-free path within the time t. Then, the probability that a path is found within t is

$$P(t) = 1 - P_{n+o+...+q}(t) = (1 - P_1(t))^n(1 - P_2(t))^o \ldots (1 - P_k(t))^q \quad (2)$$

where n, o, ..., q denote the number of the parallel executed instances of the respective programs. The programs might be different in respect of the algorithm and/or the parametrization of the algorithm.

6.3 General remarks to the Generalized OR paradigm

The effect of this approach can be shown by the evolution of the probabilities of some random processes and their combinations. Several sequences of random numbers were generated based on an exponential distribution function. They are characterized by an exponential coefficient (8, 10, 9, 11 in our case) and a static time offset (0.30s, 0.15s, 0.25s, 0.18s) to represent the characteristics of different path planner evaluations.

Exemplary, in Figure 8 the probability of finding a collision-free path is depicted as a function of time and of number of programs. The arrow in the upper left axis indicates that for an increasing number of parallel path planning programs, the probability approaches a step function at time $t = t_{Offset} + t_{calc,\ min}$ which due to the probabilistical completness of sampling-based path planning would be achieved for an infinite number of simultaneously starting programms. The upper and lower axes show four different occurrences of path planning probability functions for 1 to 66 parallely running programms. In the middle axes, the combinations of 33 of the upper and 33 of the lower algorithms is depicted. Note that in both cases, a speedup with respect to the worse performing algorithm is achieved.

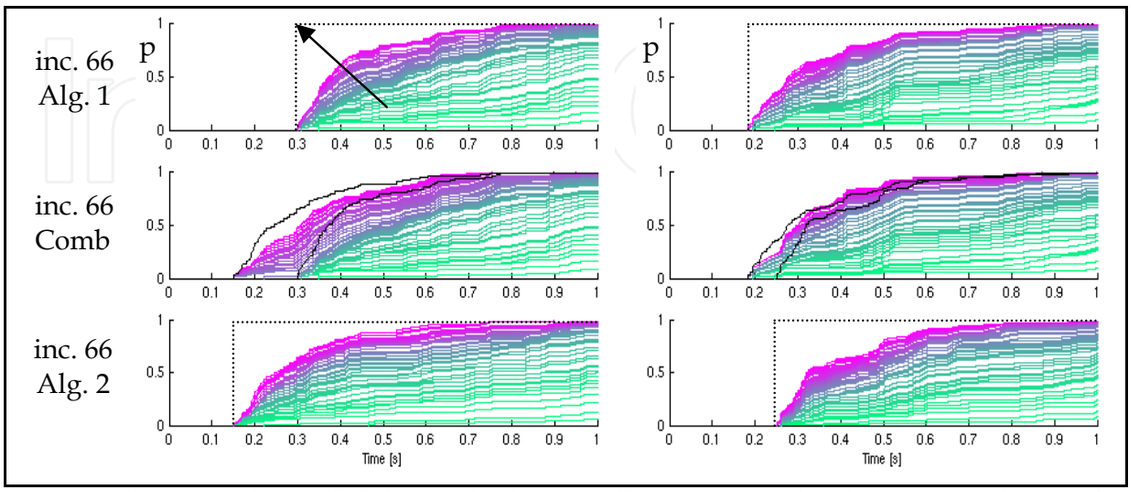

Fig. 8. Evolution of the probability of finding a collision-free path. The arrow indicates that for an increasing number of programs, the probability approaches a step function.

Based on Equ. (2), the general conclusion can be drawn that from an algorithmic point of view the performance of the overall sampling-based path planning will *always* increase if additional planners are started, because each planner contributes to the overall probability. In the following, we point out four advantages and research directions arising from this.

6.4 Potential Advantage 1: Synergy by combining PRMs and RRTs

Often, path planning queries can be faster calculated for existing PRMs than for single-shot RRTs. However, building the PRM requires a significant amount of time, which limits their application. The best option might be to build one ore more roadmaps while path planning queries are answered by other algorithms. Then, as long as the environment doesn't change significantly, the typically very efficient PRM queries can be performed. This way, both the advantages of PRMs and RRTs can be utilized. For the example given in Table 5, combinations of *RRT-cla*, *RRT-vis* and *PRM-vis* could drastically reduce the worst-case maximum duration of path planning both during and after building a PRM.

In Figure 9, the performance of the parallel execution of *RRT-cla* and *RRT-vis* is given for scenario 3. As had been expected from the results of Table 5, the *RRT-vis* was better in the broad configuration space and the RRT-cla in the very narrow one. Due to this combination, the poor performance of the *RRT-cla* in the very narrow case are barely noticable when compared to parallel executions of only *RRT-vis*. This underlines the increase of the reliability which is inherently achieved by the Generalized OR-parallelization.

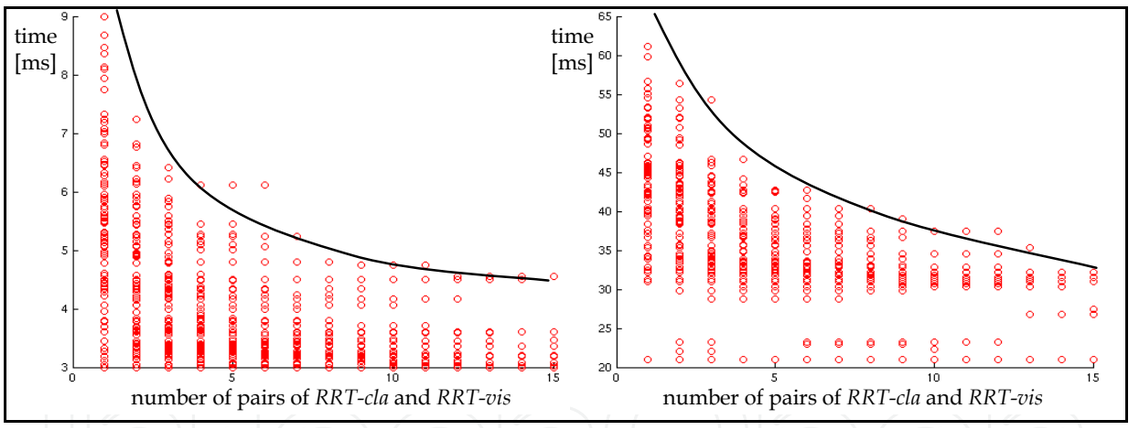

Fig. 9. Decrease of the shortest computation time per run with increase of the number of RRT-based path planner pairs for scenario 3 ("broad", l. and "very narrow", r.).

6.5 Potential Advantage 2: Utilization of Different Parameterizations of Algorithms

The choice of the parameters of an algorithm drastically influences its performance, see e.g. Section 3.6. One of the big problems with the parameterization is that due to the infinite combinations of robots and environments, most planners will perform badly for at least some "pathological" cases, where e.g. the C-space is extremely dense. However, the default parameter set of e.g. a PRM planner might not be designed for solving this particular case, but to perform well in the majority of the planning tasks. Using our approach, well-proven default and purpose-built parameter sets can be used for arbitrary scenarios.

6.6 Potential Advantage 3: Adaptive Parameterization of the Algorithms

Additionally to the utilization of different parameter sets for path planning algorithms, these parameters should be adapted online. In the previous sections, we pointed out that especially the performance of PRM planners relies on appropriate parameters such as the number of initial states or the desired density of the map. Based on PRM performance criteria such as query time and query success, these parameters can be adaptively balanced.

6.7 Potential Advantage 4: Advanced Adaptive OR-Parallelization Scheme

If there are enough processing resources that all relevant planning algorithms can be executed simultaneously, an advanced adaptive OR-parallelization can be realized: Based on the evolution of the path planning duration of the individual algorithms, the candidate(s) with the highest likelihood for fast path planning results is identified online and subsequently started more often than the other planners. Thus, based on the definition of specific criteria, an optimization of the OR-parallelization can be performed. This optimization should take into account the quality of the estimation of the path planning durations, e.g. it has to take care that sufficiently “non-optimal” algorithms are running.

7. Conclusion and Future Work

We have developed SamPP, a generic sampling-based path planning library and successfully applied to a variety of robots and environments. Due to the implementation of RRT and PRM algorithms, SamPP is able to solve low- as well as high-dimensional problems efficiently.

The ability to solve a high-dimensional path planning scenarios has been shown by the example of ViSHaRD10, a robot with 10 DOF.

Furthermore, we evaluated the performance of SamPP for executing path planning for different car doors with 2 DOF within an (in terms of free configuration space) very demanding environment. For the case of 300 to 400 obstacles, nearly "worst-case" placed in the workspace of these car doors, we found typical mean values for the path planning time in the area of 50 ms for RRTs, 1500 ms for building a PRM and 30 ms for PRM queries. The evaluation results show that the performance of SamPP indeed is sufficient for the haptic real-time assistance of a human in various scenarios with 2 DOF. Independently of the planning algorithm, the path postprocessing seems to work quite well if there are no overly narrow passages in the C-space of the robot.

Based on these results, we developed a “real-time” haptic support method and applied it to a virtual car door. An experimental user study revealed that the haptic support is appreciated by the users.

Furthermore, we enhanced the path planning performance for unknown or dynamical environments significantly by the *OR-Parallelization* of different path planning queries. This *Generalized OR-Parallelization* is a novel concept that to the best knowledge of the authors has not been proposed beforehand. We showed that for the case of dynamic environments the likelihood of a fast path planning result is higher with our approach.

Finally, we highlight four promising research directions to exploit the advantages of the concept of *Generalized OR-Parallelization*: 1) Combination of PRMs and RRTs to achieve synergy of the advantages of both concepts, 2) concurrent use of different parameter sets of

path planning algorithms, 3) online adaptation of these parameter sets and 4) online adaptation of the types and numbers of parallel executed path planning programs.

Acknowledgement

This work has been supported by BMW Group in the framework of CAR@TUM.
First of all, the authors would like to thank Andreas Dömel for his valuable contributions during and after his Studienarbeit (Dömel, 2007). Furthermore, the authors would like to thank Klaas Klasing for his constant support and advice. Finally, the authors would like to thank Amir Solhjoo for his contributions in the user study (Solhjoo, 2009).

8. References

Abbott, J.J.; Hager, G.D. & Okamura, A.M. (2003). Steady-Hand Teleoperation with Virtual Fixtures. *Proceedings of the IEEE International Workshop on Robot and Human Interactive Communication*, Millbrae, California, USA, 2003.

Ammi M. & Ferreira, A. (2007). Robotic Assisted Micromanipulation System using Virtual Fixtures and Metaphors, *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pp. 454-460, 2007.

ANN [Online]. *http://www.cs.umd.edu/~mount/ANN/*. Accessed on September 19th, 2009.

Arnato, N. & Dale, L. (1999). Probabilistic roadmap methods are embarrassingly parallel. *Proceedings of 1999 IEEE International Conference on Robotics and Automation*, 1999, pp. 688-694.

Calisi, D. (2008). Motion Planning, Reactive Methods, and Learning Techniques for Mobile Robot Navigation, *www.dis.uniroma1.it/~dottoratoii/db/relazioni/relaz_calisi_2.pdf* Accessed on September 19th, 2009.

Challou, D.; Boley, D.; Gini, M. & Kumar, V. (1995). A parallel formulation of informed randomized search for robot motion planning problems. *Proceedings of 1995 IEEE International Conference on Robotics and Automation*, 1995, pp. 709-714.

Davies, B.L.; Harris, S.J.; Lin, W.J.; Hibberd, R.D.; Middleton, R. & Cobb, J.C. (1997). Active compliance in robotic surgery--the use of force control as a dynamic constraint. *Proc Inst Mech Eng H.*, Vol 211, No. 4 (1997), pp. 285-292.

Diankov, R. and Kuffner, J.J. (2008). OpenRAVE: A Planning Architecture for Autonomous Robotics. *Technical Report CMU-RI-TR-08-34*, Robotics Institute, Carnegie Mellon University, Pittsburgh, USA, 2008.

Dömel, A. (2007). Entwicklung eines Pfadplaners für einen Virtual Reality-Versuchsstand. *Studienarbeit*, TU München.

Esen, H. (2007). Training in Virtual Environments via a Hybrid Dynamic Trainer Model. *PhD Thesis*, TU München, 2007.

Fischer, M.; Braun, S.C.; Hellenbrand, D.; Richter, C.; Sabbah, O.; Scharfenberger, C.; Strolz, M.; Kuhl, P. & Färber, G. (2008). Multidisciplinary Development of New Door and Seat Concepts as Part of an Ergonomic Ingress/Egress Support System. *FISITA 2008, World Automotive Congress*, Munich, 2008.

Kapoor, A.; Li, M. & Taylor, R.H. (2007). A Constrained Optimization Approach to Virtual Fixtures for Multi-Handed Tasks. *Proceedings of the 2007 IEEE International Conference on Robotics and Automation*, Orlando, Florida, May 2009.

Kavraki, L.; Svestka, P.; Latombe, J.-C. & Overmars, M. (1996). Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *IEEE Transactions on Robotics and Automation*, Vol. 12, No. 4., pp. 566-580.

Klasing, K. (2009). Parallelized Sampling-based Path Planning for Tree-structured Rigid Robots, *Technical Report TR-LSR-2009-03-01-Klasing,* Institute of Automatic Control Engineering, TU München, 2009.

Kuffner, J.J. & LaValle, S.M. (2000). RRT-Connect: An Efficient Approach to Single-Query Path Planning. *Proceedings of 2000 IEEE International Conference on Robotics and Automation (ICRA 2000)*, Vol. 2, pp. 995-1001.

LaValle, S.M. (2006). *Planning Algorithms*. Cambridge University Press.

LaValle, S.M. (1998). Rapidly-exploring random trees: A new tool for path planning. *Technical Report TR 98-11*, Computer Science Dept., Iowa State University, Oct. 1998.

Li, M.;Ishii, M. & Taylor, R. H. (2007). Spatial Motion Constraints Using Virtual Fixtures Generated by Anatomy. *IEEE Transactions on Robotics*, Vol. 23, No. 1, pp 4-19.

Lynch, K.M.; Liu, C.; Rensen, A.S.; Kim, S.; Peshkin, M.; Tickel, T.; Hannon, D. & Shiels, K. (2002). Motion Guides for Assisted Manipulation. *International Journal of Robotics Research*, Vol. 21, No. 1 (2002), pp. 27-43.

OpenRAVE [Online]. *http://openrave.programmingvision.com/*. Accessed on September 19th, 2009.

Mount, D.M. & Arya, S. (1997). ANN: A Library for Approximate Nearest Neighbor Searching," *Proceedings of Center for Geometric Computing Second Ann. Fall Workshop Computational Geometry*, 1997.

Plaku, E.; Bekris, K.E.; Chen, B.Y.; Ladd, A.M. & Kavraki, L.E. (2005). Sampling-based roadmap of trees for parallel motion planning. *IEEE Transactions on Robotics*, Vol. 21, No. 4 (2005), pp. 597-608.

PP [Online]. *http://www.lsr.ei.tum.de/research/software/pp/*. Accessed on September 19th, 2009.

Rosenberg, L. (1993). Virtual fixtures: Perceptual tools for telerobotic manipulation. *Proceedings of IEEE Virtual Reality Annual International Symposium*, pp. 76–82, Sept. 1993.

Solhjoo, A. (2009). Further Development and Implementation of the Control of a Car Door with Two Actuated Degrees-of-Freedom. Master´s Thesis, TU München.

Strolz, M.; Mörtl, A.; Gräf, M. & Buss, M. (2009). Development, Control and Evaluation of an Actuated Car Door, *IEEE Transactions on Haptics*, Vol. 2, No. 3 (2009), pp. 170-180.

Strolz, M.; Mühlbauer, Q.; Scharfenberger, C.; Färber, G. & Buss, M. (2008). Towards a generic control system for actuated car doors with arbitrary degrees of freedom. *Proceedings of IEEE Intelligent Vehicles Symposium (IV 2008)*, pp. 391-397, Eindhoven, The Netherlands, June 2008.

Ueberle, M.; Mock, N. & Buss, M. (2004). ViSHaRD10, a novel hyper-redundant haptic interface. *Proceedings of the 12th International Symposium on Haptic Interfaces for Virtual Environment and Teleoperator Systems*, pp. 58-65, 2004.

Varol, A.; Gunev, I. & Basdogan, C. (2006). A Virtual Reality Toolkit for Path Planning and Manipulation at Nano-Scale. *Proceedings of the 14 th IEEE Symposium on Haptic Interfaces for Virtual Environments and Teleoperator Systems*, pp. 485-489, Washington D.C., USA, March 2006.

Advances in Haptics

Edited by Mehrdad Hosseini Zadeh

ISBN 978-953-307-093-3
Hard cover, 722 pages
Publisher InTech
Published online 01, April, 2010
Published in print edition April, 2010

Haptic interfaces are divided into two main categories: force feedback and tactile. Force feedback interfaces are used to explore and modify remote/virtual objects in three physical dimensions in applications including computer-aided design, computer-assisted surgery, and computer-aided assembly. Tactile interfaces deal with surface properties such as roughness, smoothness, and temperature. Haptic research is intrinsically multi-disciplinary, incorporating computer science/engineering, control, robotics, psychophysics, and human motor control. By extending the scope of research in haptics, advances can be achieved in existing applications such as computer-aided design (CAD), tele-surgery, rehabilitation, scientific visualization, robot-assisted surgery, authentication, and graphical user interfaces (GUI), to name a few. Advances in Haptics presents a number of recent contributions to the field of haptics. Authors from around the world present the results of their research on various issues in the field of haptics.

InTech Europe
University Campus STeP Ri
Slavka Krautzeka 83/A
51000 Rijeka, Croatia
Phone: +385 (51) 770 447
Fax: +385 (51) 686 166
www.intechopen.com

InTech China
Unit 405, Office Block, Hotel Equatorial Shanghai
No.65, Yan An Road (West), Shanghai, 200040, China
中国上海市延安西路65号上海国际贵都大饭店办公楼405单元
Phone: +86-21-62489820
Fax: +86-21-62489821

IntechOpen