Integrating task and PRM motion planning: Dealing with many infeasible motion planning queries

Kris Hauser[1] and Jean-Claude Latombe[2]

(1) School of Informatics and Computing, Indiana University, Bloomington, USA
(2) Computer Science Department, Stanford University, California, USA

Abstract

To accomplish a task an autonomous robot must break this task into "primitive" subtasks and order them to satisfy precedence constraints. Each subtask requires performing a motion. The existence of a feasible trajectory is an additional precondition for the subtask, but a very expensive one to test. Probabilistic RoadMaps (PRM) are an effective approach to plan feasible trajectories when these exist. However, PRM planners are unable to detect that no solution exists. On the other hand, a task/motion planner must often consider many subtasks, a fraction of which, only, admit feasible trajectories. This paper proposes a general algorithm (I-TMP) that specifically addresses this issue. This algorithm interweaves task and motion planning, and allows distributing computational effort where it is most useful. It is probabilistically complete in the following sense: if I-TMP can generate a sequence of subtasks that admits a feasible trajectory, such a trajectory will eventually be found with high probability. An application of I-TMP to multi-limbed robots navigating on rough terrain is presented.

I. Introduction

Consider an autonomous multi-limbed robot (e.g., a humanoid robot) that must accomplish a complex task, like gardening, repairing a bicycle, performing a scientific experiment, navigating on rugged terrain. The robot must break this task into primitive subtasks (e.g., move right hand to grasp a screwdriver, bring grasped screwdriver in contact with screw tip, move left foot onto a ladder rung) and order them to satisfy logical precedence constraints. Each primitive subtask requires performing a motion that eventually creates or breaks contacts with the environment. The existence of a feasible trajectory for this motion is an additional precondition for the subtask, but a very expensive one to test.

Here we consider the case where the robot's planner must search a huge (possibly infinite) space of subtasks represented by a graph – the *subtask graph* – whose nodes are subtasks and arcs are precedence constraints. We assume that the robot has many degrees of freedom, so that the only available viable motion planning approach to compute feasible trajectories is the Probabilistic RoadMap (PRM) approach (Kavraki et al, 1996; Hsu et al., 1999; LaValle and Kuffner, 1999; Sanchez and Latombe, 2002; Akinc et al., 2005). A PRM planner approximates the connectivity of the robot's feasible motion space by a network of simple trajectories connecting configurations sampled according to some probability distribution (Hsu et al., 2006). If properly implemented, this planner is probabilistically complete and has fast convergence, i.e., the probability that it fails to find a solution trajectory when one exists converges toward 0 exponentially in the number of sampled configurations. However, it is unable to detect that no solution exists. So, a PRM planner terminates with failure after some cutoff time.

A significant fraction of subtasks are often infeasible because no trajectories exist to perform them (e.g., the robot would collide with obstacles, lose balance, or lose sight of a key object). If the cutoff time of the PRM planner for each motion-planning query is set too high, then much time will be wasted on infeasible subtasks; if it is set to low, critical subtasks may be incorrectly labeled as infeasible, and the overall task planner may eventually fail to find a plan for the task at hand.

We propose a general algorithm (I-TMP, for Incremental Task/Motion Planner) that specifically addresses this issue. I-TMP interweaves task and motion planning, and allows distributing computational effort where it is the most useful. The algorithm is probabilistically complete in the following sense: if I-TMP can generate a sequence of subtasks that admits a feasible trajectory, such a trajectory will eventually be found with high probability. We also describe heuristic techniques to distribute computational effort in order to speed up the overall planner. Finally, we present an application of I-TMP to multi-limbed robots navigating on rough terrain.

II. Related Work

The integration of task planning and PRM motion planning has been recently studied in (Cambon, 2009) for manipulation tasks. In this study the structure of the search space is similar to the one considered here, but slightly more specific. However, the key issue of the unfeasibility of many subtasks is not directly addressed.

The integration of task/motion planning has also been considered for mobile robots navigating among movable obstacles (van den Berg et al., 2008). In this context, the robot must displace movable obstacles in order to open passageways to reach a specified goal position. However, movable obstacles may interact in a complex way; for instance, displacing an obstacle may require the prior displacement of other obstacles. The proposed approach tries to solve this problem by searching for a continuous trajectory in a large configuration space that not only encodes the parameters defining the placement of the robot in the workspace, but also those of the movable obstacles. Only planar problems and simple robots are considered, so that a trajectory can be broken into segments, each of which lies in a low-dimensional space.

The issue of the unfeasibility of motion planning queries in a task/motion planner was introduced in (Bretl et al. 2005). Independently, other researchers have proposed algorithms, called "disconnection planners", to detect that a motion planning problem has no solution (Basch et al., 2001; Hirsch and Halperin, 2004; Zhang et al., 2008)). However, these algorithms are only applicable to simple robots with few degrees of freedom (2 or 3) and rely on coarse approximation of the robot's shape. In general, they can only detect certain infeasibility. A more general method based on semi-algebraic techniques is proposed in (Bretl et al., 2005); however, it usually takes prohibitive time to run.

The work described in this paper is an extension of the planning framework proposed in (Bretl, 2006).

III. Structure of the Search Space

A. Subtask Graph

We assume that each possible subtask is described by preconditions and effects using a logic-based language (Russel and Norvig, 2003) similar to STRIPS (for the purpose of this paper the details of this language are not important). Ignore for a moment that feasible trajectories are needed to perform subtasks. In a given state of the world, a subtask is feasible if its preconditions (precedence constraints) are satisfied in that state. The initial state of the world, the conditions defining a goal state and the subtask descriptions implicitly define a directed graph G, which we call the *subtask graph*:

- A subtask is associated with each node of G.

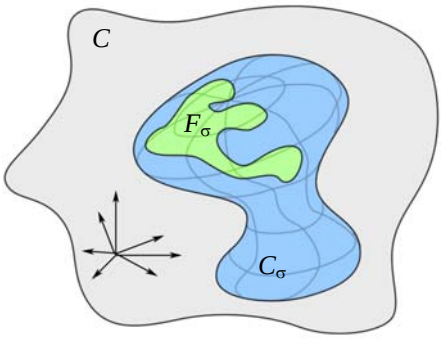

Figure 1: Configuration space C, submanifold C_σ defined by dimensionality-reducing constraints, and feasible space F_σ defined by volume-reducing constraints.

- The preconditions of each subtask in G with no parent are verified in the initial state of the world. The corresponding node is a *start node*.
- Every continuous sequence of subtasks in G starting at a start node is feasible as far as the preconditions of the successive subtasks are concerned.
- Every subtask in G that achieves the goal conditions is a *goal node*.
- So, every continuous sequence of subtasks between the start node and a goal node is a plan for the task at hand.

G may, or may not be finite. The same subtask may be associated with several nodes of G.

B. Feasible Spaces

Let us now consider the requirement that feasible trajectories are needed to perform subtasks.

A robot trajectory can be represented by a continuous curve segment in a parameter space C called the robot's *configuration space* (Lozano-Pérez, 1983). This space is usually parameterized by the robot's degrees of freedom (e.g., its joint angles). For a humanoid robot it typically has between 30 and 50 dimensions. C is a manifold, meaning that it is locally similar to a linear space (Latombe 1991).

Each subtask σ in G determines a feasible subset F_σ of the robot's configuration space C – the subtask's *feasible space*. F_σ consists of all the configurations that achieve the constraints imposed by the subtask, e.g., avoiding collision, maintaining certain contacts with the environment, maintaining balance, and keeping certain objects in view. For example, if the subtask σ requires the robot to move a box held with both hands with its two feet making fixed contacts with the terrain, then F_σ is the subset of all configurations where (1) neither the robot nor the box collide with obstacles, (2) the joint angles in the robot are such that the torso, the two arms and the box form a closed kinematic chain, (3) the pelvis, the two legs and the terrain form another closed chain, and (4) the robot's center of

mass (taking into account the box) is above the support polygon.

F_σ is a subset of a sub-manifold C_σ embedded in the configuration space C (Fig. 1). The constraints defining C_σ within C are dimensionality-reducing constraints of the form $H_\sigma(q) = 0$; they usually are constraints requiring that some contacts with the environment be maintained (Hauser and Latombe, 2008). The constraints defining F_σ within C_σ are volume-reducing constraints of the form $K_\sigma(q) > 0$ (e.g., collision avoidance). F_σ is usually high-dimensional and geometrically complex. It may also be made of several connected components. In some cases, it is empty, that is, no configurations satisfy all the constraints.

In general, two distinct subtasks admit different feasible spaces. For example, in a subtask σ' different from the above subtask σ, the robot may not hold any object and may be standing on a single foot. Then, both the dimensionality-reducing and volume-reducing constraints are different, so that $F_{\sigma'}$ has greater dimensionality than F_σ and a different geometric shape.

In the following, we assume that primitive subtasks are chosen such that robot-environment contacts stay fixed during any given subtask, except at the start and the end of the subtask where new contacts may be created or existing contacts may be broken (e.g., by grasping or un-grasping an object, or making a new foot contact with a rugged terrain or breaking one). Therefore, any feasible space F_σ is contained in a sub-manifold C_σ of fixed dimensionality. Two successive feasible spaces in a task plan often have different dimensionalities. Subtasks that require multiple changes of contact are encoded as a series (or if order is unspecified, a subgraph) of primitive subtasks.

C. Transitions

Let σ and σ' be two adjacent subtasks in G and let F_σ and $F_{\sigma'}$ be their respective feasible spaces. We call the intersection $F_\sigma \cap F_{\sigma'}$ the *transition space* between σ and σ'. Any point in this space is a *transition configuration* between σ and σ' (Fig. 2). It follows from the assumption stated above that contacts can only be created or broken at transition configurations. The execution of a task plan by a robot can now be seen as a motion along a continuous trajectory concatenating successive sub-trajectories, each contained in the feasible space of a subtask. The endpoint of a trajectory in one feasible space F_σ is the start of the trajectory in the next feasible space $F_{\sigma'}$, hence a transition configuration in $F_\sigma \cap F_{\sigma'}$ (this does not require, however, that a trajectory from F_σ to $F_{\sigma'}$ crosses $F_\sigma \cap F_{\sigma'}$ at a single configuration). So, for a task plan to be feasible, a continuous trajectory must exist that connects the initial robot configuration s to a goal configuration g through the feasible spaces associated with the successive subtasks in the plan.

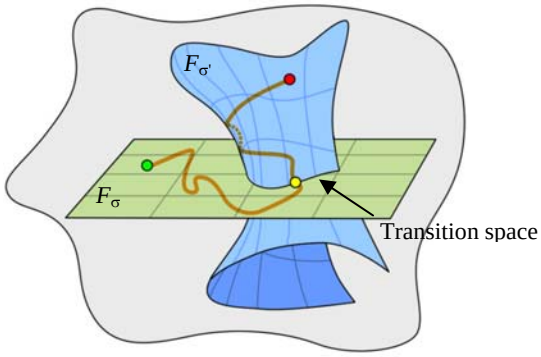

Figure 2: Two intersecting feasible spaces F_σ and $F_{\sigma'}$. To switch between them the robot must cross the transition space $F_\sigma \cap F_{\sigma'}$. F_σ and $F_{\sigma'}$ often have different dimensionalities.

Transitions play an important role, as they usually constitute bottlenecks for motion planning. Indeed, the transition space between two subtasks must satisfy the constraints associated with both subtasks. For example, when a robot un-grasps an object, two constraints, at least, must be simultaneously satisfied: (1) the object must be at a location where it will stay in equilibrium after it is un-grasped and (2) the robot must hold the object at this location. Similarly, when a legged robot breaks a contact to move one of its feet to a new footfall, it is must both satisfy the kinematic constraint that the foot is still at its initial contact position and achieve balance without applying any force at this contact position. So, transition spaces are more likely to be empty than feasible spaces, or at least smaller.

IV. Task/Motion Planning

A. PRM Planning

For most robots, computing an exact explicit geometric representation of a single feasible space F_σ is prohibitively time consuming, due to the high dimensionality and the geometric complexity of that space. Instead, a PRM planner approximates the connectivity of such a space by a network of simple trajectories, called a *roadmap*. The nodes, called *milestones*, are configurations sampled using some probabilistic distribution.

A property – *expansiveness* – was introduced to characterize how quickly a PRM planner can solve motion planning problems (Hsu et al., 1999). Roughly speaking, a feasible space is expansive if it does not contain arbitrarily narrow passages (regions of zero relative volume connecting other regions). In that case, it can be shown that the probability that the planner fails to find a solution when one exists converges toward 0 exponentially in the number of sampled configurations. However, a PRM planner is unable to detect that no solution exists.

Smoothed analysis shows that narrow passages are unstable under small perturbations of the geometry of the robot and its environment (Chaudhuri and Koltun, 2007)

and therefore are unlikely to occur in practice when the robot maintains a fixed set of contacts with its environment. This is the case for every feasible space F_σ, under the assumption that each subtask keeps contacts fixed (except at the start and end). However, the union of all the feasible spaces contains regions of varying dimensionalities as the robot change contacts between subtasks. So, this union is non-expansive, with subspaces of lower dimensionalities forming arbitrarily narrow passages between subspaces of higher dimensionalities. PRM planning must be adapted to handle these passages.

B. Addressing the Varying Dimensionality Issue

A natural approach to deal with the varying dimensionality issue is to sample each feasible space individually and construct a separate roadmap in it. An aggregate roadmap can then be constructed over the union of the feasible spaces by connecting the individual roadmaps at milestones specifically sampled in transition spaces. This is the approach of the TMP (for Task/Motion Planner) algorithm presented below.

Like in any PRM planner we need two functions to define TMP, respectively to sample milestones (SAMPLE) and to create connections between milestones (CONNECT):

- SAMPLE(X), where X is a feasible space F_σ (or a transition space $F_\sigma \cap F_{\sigma'}$), first samples a configuration q in the $C_\sigma \supset F_\sigma$ (or in the subspace $C_\sigma \cap C_{\sigma'} \supset F_\sigma \cap F_{\sigma'}$) and then tests whether $q \in X$. It returns q if $q \in X$ and failure otherwise. C_σ can always be parameterized using a set of charts forming an atlas (Latombe, 1991). To sample q in C_σ one may pick a chart from the atlas and independently sample each coordinate in that chart (Cortés and Siméon, 2005). Another way is to (1) define a manifold M containing C_σ that is easier to parameterize than C_σ (for example, M a linear space), (2) sample a configuration in M, and (3) use a numerical method to move this configuration into C_σ (Yakey et al., 2001). The same techniques can be used for sampling from $C_\sigma \cap C_{\sigma'}$.

- CONNECT(q,q',F_σ) tries to connect two milestones q and q' in F_σ. It picks a sequence of charts in the atlas of C_σ such that the first chart in the sequence covers q and the last one covers q'. Then, for every two successive charts in this sequence it picks a configuration contained in both charts. Finally, it connects q to q' by a trajectory made of successive line segments (one in each successive chart) joining these intermediate configurations. If this trajectory fully lies in F_σ, then CONNECT returns the trajectory. Alternatively, CONNECT may compute the trajectory by joining q and q' with a straight line in M and numerically deforming this line into F_σ (Hauser, 2008).

Suppose that an explicit representation of the subtask graph G can be pre-computed (in that case, G must have finite size). TMP concurrently builds roadmaps in all the feasible spaces associated with the subtasks in G, connecting them at transition milestones:

TMP(G,s,g,N)

1. For each subtask σ associated with a node of G, initialize R_σ to the empty roadmap.
2. For each subtask σ associated with a start node of G add the start configuration s as a milestone of the roadmap R_σ.
3. Similarly, for each subtask σ associated with a goal node of G add the goal configuration g as a milestone of the roadmap R_σ.
4. Repeat N times:
   a. For every subtask σ in G, if SAMPLE(F_σ) succeeds, add the returned configuration q to R_σ as a new milestone and try to connect q to each previously existing milestone q' in R_σ using CONNECT(q,q',F_σ).
   b. For every pair of adjacent subtasks σ and σ' in G, if SAMPLE($F_\sigma \cap F_{\sigma'}$) succeeds, add the returned q as a new milestone to both R_σ and $R_{\sigma'}$, and try to connect q to each previously existing milestone q' in R_σ and $R_{\sigma'}$ with CONNECT(q,q',F_σ) and CONNECT(q,q',$F_{\sigma'}$).
5. Let R be the aggregate roadmap obtained by connecting the roadmaps R_σ at matching transition milestones. If s and g are connected by a path in R, then return a solution trajectory; otherwise return failure.

It is shown in (Hauser, 2008; Hauser and Latombe, 2008) that, when there actually exists a solution trajectory, the probability that TMP returns failure converges toward 0 exponentially with N, under the following two (usually satisfied) assumptions: (1) each non-empty feasible space F_σ is expansive and (2) SAMPLE(X) succeeds with non-zero probability whenever X is non-empty. It is also shown that if TMP iterates until a solution is found, the mean and variance of the running time to find a solution, when one exists, are bounded.

However, the running time of TMP is linear in the size of G, which in most practical cases is huge or even infinite. So, TMP can only be used on small problems.

C. Incremental Planning

In practice, even when G is huge, most tasks can be solved by plans made of a relatively small number of subtasks (typically a few dozens, or less). The underlying idea of the incremental planning algorithm I-TMP is to restrict TMP to a small portion of G containing these subtasks. Since these subtasks are not known in advance, I-TMP iteratively executes two successive steps: subtask selection and roadmap construction.

At each cycle i = 1, 2, …, I-TMP builds a subgraph G_i of G such that $G_{i-1} \subset G_i$ and creates roadmaps in each of the feasible spaces F_σ associated with the subtasks in G_i. Specifically, it performs the following:

For i = 1, 2, … do:

1. *Subtask selection*: Construct G_i
2. *Roadmap construction*: TMP(G_i,s,g,N)

In practice, the roadmaps built at cycle i–1 are not forgotten, but instead incremented with new milestones and connections at cycle i. I-TMP returns failure if it has not found a solution after a certain number of cycles or a given cutoff time. When G is finite, I-TMP is probabilistically complete and has the same asymptotic convergence as TMP because the subgraphs G_i will eventually grow to G in its entirety. However, I-TMP does not require G to be pre-computed (the subgraphs G_i can be built by searching G), so it can work even when G is infinite. In this case, the completeness of I-TMP depends on the completeness of the algorithm searching G. If I-TMP can generate a sequence of subtasks that admits a feasible trajectory, this trajectory will eventually be found with high probability.

However, for I-TMP to be efficient, it must select the successive subgraphs G_i so that a trajectory joining s and g can be found at a cycle i where G_i is still reasonably small. A simple heuristic is to extract, at each cycle i, one or a few new paths joining start to goal nodes in G and insert these new paths into G_{i-1} to produce G_i. Thus, I-TMP will not waste time sampling feasible and transition spaces in subtask sequences that do not eventually lead to a goal node in G. However, this heuristic alone does not necessarily lead to generating sequence of subtasks that admit feasible trajectories. We now address this critical issue.

D. Dealing with Infeasible Queries

As mentioned before, transition spaces are the bottlenecks for motion trajectories as each such space must satisfy the constraints associated with two subtasks. So, the non-emptiness of every transition space $F_\sigma \cap F_{\sigma'}$ along a sequence of subtasks in G_i is not only a necessary condition for the existence of a feasible trajectory through these tasks; it is also a good indication that such a trajectory actually exists. Moreover, they are not a computational bottleneck, because it is usually much faster to sample transition spaces than to build roadmaps in feasible spaces. These two observations lead to constructing G_i by searching G using a heuristic function that estimates the non-emptiness of the transition spaces.

More precisely, let Σ be the search tree constructed while searching G and let Q be a priority queue of transition spaces T sorted by decreasing values of a priority function $p(T)$ discussed below. At the beginning of cycle i=1, Σ is initialized to contain only the initial nodes in G and Q is initialized to contain all the transition spaces associated with the arcs stemming from these nodes in G. Each transition space is inserted into Q with some initial priority. At cycle i the search algorithm is then as follows:

1. Repeat until either Σ contains a path to a goal node such that this path is not contained in G_{i-1}, or a cutoff time has been reached:
   a. Remove the first transition space T from Q. Perform SAMPLE(T).
   b. On failure, reduce $p(T)$ and reinsert T into Q.
   c. On success, let $T = F_\sigma \cap F_{\sigma'}$. Add σ' to Σ as a child of σ. For each transition space T' stemming from σ' in G, insert T' into Q with some initial priority.
2. If the cutoff time has been reached, then return failure.
3. If a new path has been found, then insert this path into G_{i-1} to produce G_i.

Note that:

- Transition configurations returned by SAMPLE are stored to be later inserted into the appropriate roadmaps R_σ.
- The algorithm can easily be modified to perform backward or bi-directional search.

There are multiple ways to define the priority function $p(T)$. One approach is to consider $p(T)$ as an estimate of the probability that T is not empty. In (Hauser et al., 2005) supervised learning is used to acquire a statistical model $p(T)$, as a function of features of the two subtasks meeting at T (e.g., the locations of the contacts made by the robot). This model can then be used to initialize $p(T)$ whenever a new transition space T is inserted into Q. When SAMPLE(T) returns failure, $p(T)$ is reduced to reflect the posterior probability, in such a way that $p(T) \to 0$ with the number of unsuccessful runs of SAMPLE(T).

In addition, the running time of I-TMP can be significantly reduced by tuning appropriately the number of sampling operations in each feasible and transition space in G_i at each cycle i. This requires adapting slightly the TMP algorithm given above. For example, it is suboptimal to run SAMPLE the same number of times in feasible and transition spaces. Because it usually takes much more work to build connected roadmaps in feasible spaces than to sample transition spaces, feasible spaces should be more heavily sampled. In addition, since PRM planning in a feasible space has fast convergence, SAMPLE should be run a smaller number of times in spaces that have already been highly sampled. In particular, since roadmaps existing at cycle i–1 are re-used at cycle i, the numbers of SAMPLE runs performed at cycle i should be greater in feasible and transition spaces that were not in G_{i-1} than in those that were already in G_{i-1}. Finally, at any stage of cycle i if the roadmap in a feasible space F_σ is fully connected, there is no immediate need to perform further runs of SAMPLE in F_σ (however, the transition space $F_\sigma \cap F_{\sigma'}$ should still be sampled further if the roadmap in $F_{\sigma'}$ is not fully connected).

For most parameter values in the heuristics, I-TMP embedding the above modifications has the same probabilistic completeness as the original TMP, but with different convergence constants.

V. Robot Navigation on Rough Terrain

We have used variants of I-TMP to plan the motion of multi-limbed robots on rough terrain, in particular:

- Capuchin, a 4-limbed climbing robot (Zhang, 2008),
- ATHLETE, a 6-legged robot designed to navigate on natural, possibly steep terrain (Hauser et al., 2008; Hauser, 2008),
- HRP-2, a humanoid robot (Hauser et al., 2008; Hauser, 2008).

With Capuchin, tests were conducted on the real robot (Fig. 3). With ATHLETE and HRP-2, they were only conducted on simulation (Fig. 4-6).

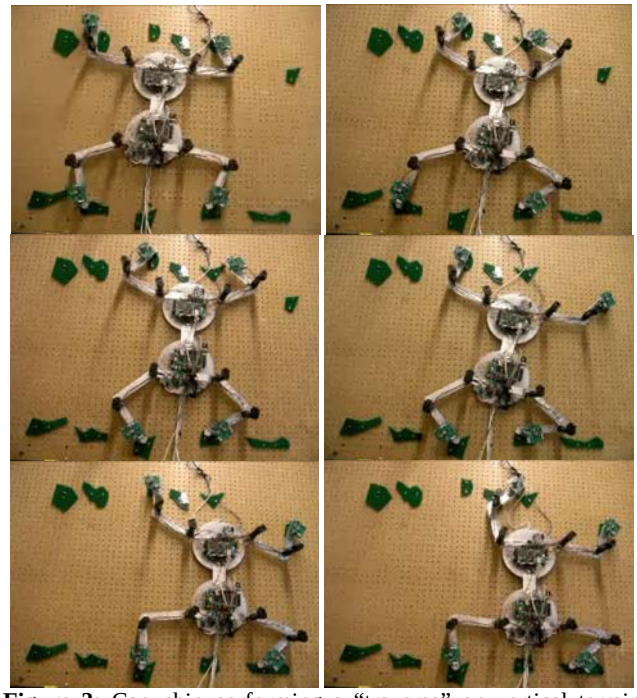

Figure 3: Capuchin performing a "traverse" on vertical terrain with sparse features.

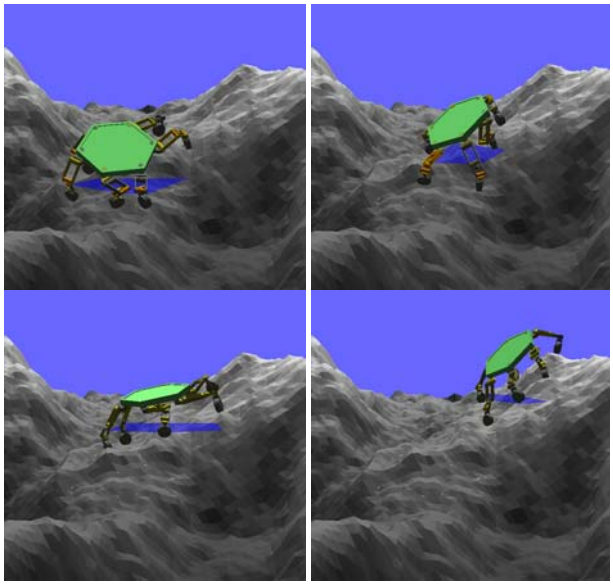

Figure 4: ATHLETE navigating on rough terrain.

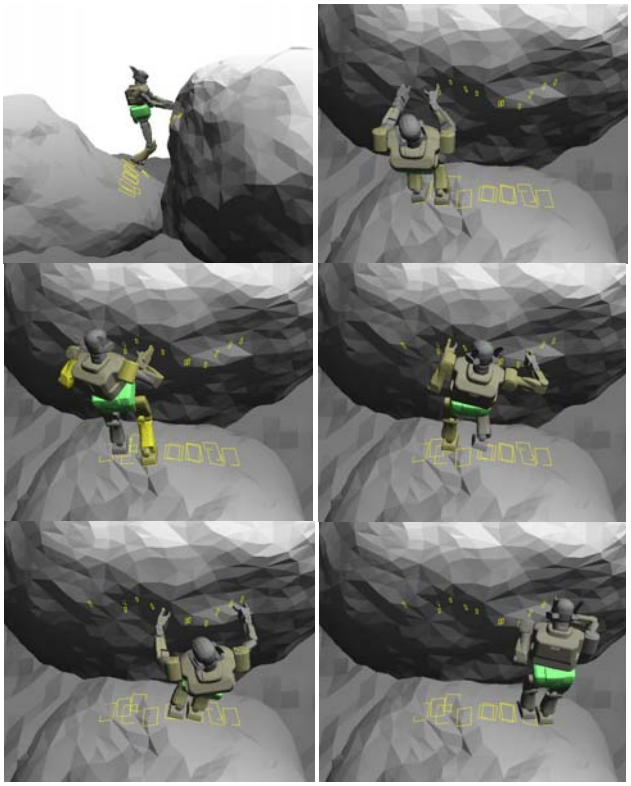

Figure 5: HRP-2 navigating on complex rock terrain requiring hand contacts to maintain stability.

ATHLETE consists of 6 identical legs attached to a hexagonal chassis. Each leg has 6 actuated revolute joints. So, the robot's configuration space has 42 dimensions. Contacts between ATHLETE and the terrain are restricted to wheel-shaped feet. Each contact, which is modeled as a point, removes 3 degrees of freedom. So, when the robot makes 6 contacts, its motion lies in a submanifold of dimensionality 42 – 18 = 24. When it makes only 3 contacts (the minimum number of contacts that allows quasi-static stability), its motion lies in a submanifold of dimensionality 42 – 9 = 33. In our work we model HRP-2 so that its configuration space is 36-dimensional. We allow the robot to contact the terrain with both its feet and hands, and contacts can be described by points, edges, or faces. Finally, Capuchin's configuration space is 11-dimensional; its contacts with the terrain are restricted to the endpoints of it limbs and are modeled as points.

For each robot, a subtask consists of moving to a configuration where the robot achieves a new contact or breaks an existing one. For instance, consider subtask σ in which ATHLETE makes 6 fixed contacts with the terrain. Let the goal of σ be to break a contact c. So, the robot must move to a configuration in F_σ where the force exerted at c is zero, while remaining stable. Once σ is completed, the robot switches to another task σ' that may, for example, be to bring the foot at the broken contact to another footfall c'.

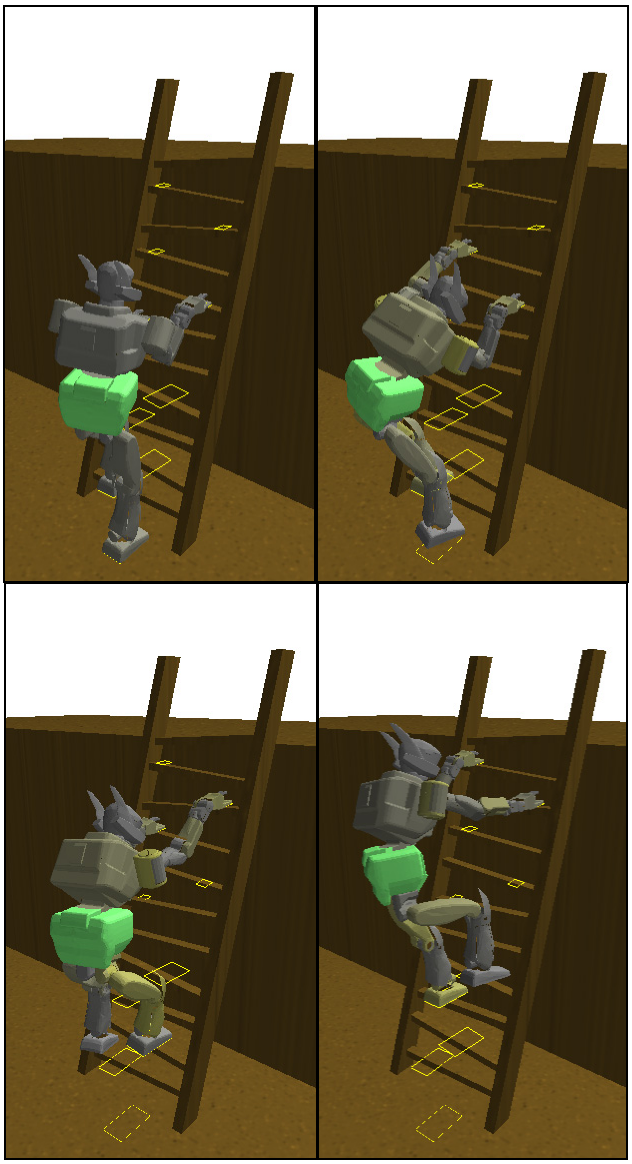

Figure 6: HRP-2 climbing a ladder with uneven rungs

For all these robots, the graph G is non-directed and consists of all the contact state (we call then *stances*) that might be feasible, i.e., where all the contacts are within a disc or sphere whose diameter is approximately the limb span of the robot. In most cases, there are several 100,000s of stances in G. Two stances are connected by an edge of G if they differ by only one contact (hence, we allow a single contact to be created or broken in each transition). The constraints on the motion are collision avoidance (including self-collision avoidance), quasi-static stability using a Coulomb friction model at the contacts (Bretl, 2006), torque limits in the joints, and kinematic closures to maintain contacts. We assume all motions to be quasi-static. Possible contact locations on the terrain are either given (e.g., in the case of Capuchin), extracted from the terrain geometry, or sampled at random. In many examples, over 70% of the transition spaces in G are empty.

Our experiments (Hauser, 2008; Hauser and Latombe, 2008) confirm that TMP is impractical, except for small problems. I-TMP is both reliable and reasonably fast (it still takes from minutes to hours to solve a planning problem). The heuristics used to tune the number of sampling operations in each feasible and transition spaces greatly improve running time, sometimes by several orders of magnitude. Certain simplifications result in significant speedups, but their reliability depends on the robot. For example, we found out that for HRP-2 it is usually sufficient to sample a single milestone per transition space without affecting the reliability of the planner. But the same simplification does not work for ATHLETE; the planner then fails consistently to solve many problems that I-TMP can actually solve when the number of milestones in each transition space is not limited. The reason for this (shown by other tests) is that the feasible spaces of HRP-2 tend to be connected, while those of ATHLETE often consist of multiple components.

VI. Conclusion

A key problem in integrating task and motion planning is to deal with the fact that many motion planning queries are infeasible. In high-dimensional configuration spaces no effective motion planning techniques exist today to detect that a query is infeasible. PRM planners can solve feasible queries efficiently, but their running time is variable. Allocating them too much time would lead the integrated planner to waste time on many infeasible queries. But too little time could lead them to fail on critical feasible queries. The algorithm I-TMP and the heuristics proposed in this paper provide an approach that allocates computational time adaptively where it is likely to be the most useful. Experiments with several limbed robots navigating on rough terrain show that this framework is effective, at least for this type of application.

Several issues remain for future research. In particular, the running time of I-TMP is still too long in many cases to achieve close-to-real-time performance without major simplifications that could impact completeness. More work should be done to learn a probabilistic model of the feasibility of subtasks. This model would then be used to better select the subtasks added to the graphs G_i at each cycle of I-TMP. Decision-theoretic planning techniques might also improve the way computational time is allocated and eventually reduce total planning time (Hauser, 2008).

Another possible improvement would be to design a motion planner that can suggest subtasks to be included in a task plan – e.g., displace an obstructive movable obstacle to clear a passage (van den Berg, 2008). In a similar vein, certain subtasks could be automatically inserted by the

motion planner based on the connectivity of the feasible space. For example, an object may not be grasped and placed at a goal location without un-grasping it at an intermediate location where it can be re-grasped in a way that makes it possible to eventually place it at the goal location. Detecting the need for intermediate un-grasp and re-grasp operations may be easier at the motion planning level than at the task planning level.

References

M. Akinc, K. E. Bekris, B. Y. Chen, A. M. Ladd, E. Plaku, and L. E. Kavraki 2005. Probabilistic roadmaps of trees for parallel computation of multiple query roadmaps. *Algorithmic Foundations of Robotics VI*, Springer Tracts in Advanced Robotics, Vol. 17, p. 80-89.

J. Basch, L. Guibas, D. Hsu, and A. T. Nguyen 2001. Disconnection proofs for motion planning. *Proc. IEEE Int. Conf. Rob. Aut.*, p. 1765–1772, Seoul, Korea.

T. Bretl 2006. Motion planning of multi-limbed robots subject to equilibrium constraints: The free-climbing robot problem. *Int. J. Rob. Res.*, 25(4):317–342.

T. Bretl, S. Lall, J.C. Latombe, and S. Rock 2005. Multi-step motion planning for free-climbing robots. *Algorithmic Foundations of Robotics VI*, Springer Tracts in Advanced Robotics, Vol. 17, p. 59-74.

S. Cambon, R. Alami, and F. Gravot 2009. A hybrid approach to intricate motion, manipulation and task planning. *Int. J. Rob. Res.*, 28(1):104-126.

S. Chaudhuri and V. Koltun 2007. Smoothed analysis of probabilistic roadmaps. *Proc. 4th SIAM Workshop on Analytic Algorithmics and Combinatorics* (ANALCO07), New Orleans.

J. Cortés and T. Siméon 2005. Sampling-based motion planning under kinematic loop-closure constraints. *Algorithmic Foundations of Robotics VI*, Springer Tracts in Advanced Robotics, Vol. 17, p.75-90.

K. Hauser 2008. *Motion Planning for Legged and Humanoid Robots*. Ph.D. Thesis, Stanford University.

K. Hauser, T. Bretl, and J.C. Latombe 2005. Learning-assisted multi-step planning. *Proc. IEEE Int. Conf. Rob. Aut.*, Barcelona, Spain.

K. Hauser, T. Bretl, J.C. Latombe, K. Harada, and B. Wilcox 2008. Motion planning for legged robots on varied terrain. *Int. J. Rob. Res.*, 27(11-12):1325-1349.

K. Hauser and J.C. Latombe 2008. Multi-modal motion planning in non-expansive spaces. *Workshop on Algorithmic Found. of Rob.* (WAFR), Guanajuato, Mexico.

S. Hirsch and D. Halperin 2004. Hybrid motion planning: Coordinating two discs moving among polygonal obstacles in the plane. *Algorithmic Foundations of Robotics V*, Springer Tracts in Advanced Robotics, Vol. 7, p. 239-255.

D. Hsu, J.C. Latombe, and H. Kurniawati 2006. On the probabilistic foundations of probabilistic roadmap planning. *Int. J. Rob. Res.*, 25(7):627–643.

D. Hsu, J.C. Latombe, and R. Motwani 1999. Path planning in expansive configuration spaces. *Int. J. of Comp. Geometry and Applications*, 9(4-5):495-512.

L.E. Kavraki, P. Svetska, J.C. Latombe, and M. Overmars 1996. Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *IEEE Tr. Rob. and Autom.*, 12(4):566–580.

J.C. Latombe 1991. *Robot Motion Planning*. Kluwer Acedemic Publishers.

S. LaValle and J. Kuffner 1999. Randomized kinodynamic planning. *Proc. IEEE Int. Conf. Rob. and Autom.*, p. 473-479.

T. Lozano-Pérez 1983. Spatial planning: a configuration space approach. *IEEE Tr. On Comp.*, C-32(2):108-120.

S. Russell and P. Norvig 2003. *Artificial Intelligence: A Modern Approach*. Prentice Hall.

G. Sanchez and J.C. Latombe 2002. On Delaying Collision Checking in PRM Planning – Application to Multi-Robot Coordination. *Int. J. Rob. Res.*, 21(1):5-26.

J. van den Berg, M. Stilman, J. Kuffner, M. Lin, and D. Manocha 2008. Path planning among movable obstacles: a probabilistically complete approach. *Workshop on Algorithmic Found. of Rob.* (WAFR), Guanajuato, Mexico.

J.H. Yakey, S.M. LaValle, and L.E. Kavraki 2001. Randomized path planning for linkages with closed kinematic chains. *IEEE Tr. Robot. and Autom.*, 17(6):951–958.

L. Zhang, Y.J. Kim, and D. Manocha 2008. Efficient cell labeling and path non-existence computation using C-obstacle query. *Int. J. Rob. Res.*, 27(11-12):1246-125.

R. Zhang 2008. Design of a climbing robot: Capuchin. *Proc. 5th Intl. Conf. on Computational Intelligence, Robotics, and Autonomous Systems*, Linz, Austria.