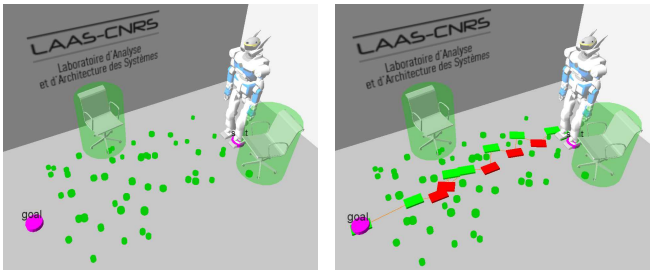
Towards Reactive Whole-Body Motion Planning in Cluttered Environments by Precomputing Feasible Motion Spaces

Andreas Orthey, Olivier Stasse

Towards Reactive Whole-Body Motion Planning in Cluttered Environments by Precomputing Feasible Motion Spaces

Andreas Orthey[1,2], Olivier Stasse[1]

***Abstract*— To solve complex whole-body motion planning problems in near real-time, we think it essentials to precompute as much information as possible, including our intended movements and how they affect the geometrical reasoning process. In this paper, we focus on the precomputation of the feasibility of contact transitions in the context of discrete contact planning. Our contribution is twofold: First, we introduce the *contact transition and object* (CTO) space, a joint space of contact states and geometrical information. Second, we develop an algorithm to precompute the decision boundary between feasible and non-feasible spaces in the CTO space. This boundary is used for online-planning in classical contact-transition spaces to quickly prune the number of possible future states. By using a classical planning setup of A* together with a l_2-norm heuristic, we demonstrate how the prior knowledge about object geometries can achieve near real-time performance in highly-cluttered environments, thereby not only outperforming the state-of-the-art algorithm, but also having a significantly lower model sparsity.**



(a) **(b)**

Fig. 1: Left: A highly-cluttered environment with 30 randomly placed objects, where the robot has to avoid stepping onto objects, while reaching a goal region. Right: Our algorithm finds a feasible footstep path in $0.3s$ by (1) approximating objects with simple geometrical shapes and (2) adding this geometrical information to the precomputation of the feasibility of motions.

I. Introduction

Consider the problem in Fig. 1a: A highly-cluttered environment has to be traversed by a humanoid robot, without stepping onto obstacles on the ground. The goal is to find a set of footsteps, which allows us to move the robot towards the goal region. This problem is problematic from different point of views: First, for each footstep we want to take, we have to compute a control law for each degree of freedom of the robot, such that we fulfill certain constraints like joint limits, dynamics and stability. Second, we have to check if the body of the robot is in collision with objects in the environment. Due to the large number of objects and the nearness of the robot to the objects, this is generally not possible in real-time. In this paper, we provide an algorithm, which generates a footstep path for a humanoid robot which is faster than the state-of-the-art approach, and runs in real-time even for challenging environment like the one in Fig. 1a, where 30 objects are randomly placed.

The underlying problem is the one of real-time planning of motions for a high-dimensional degrees of freedom robot. We approach this problem by using an approximation method to precompute if a motion between two contact points will be feasible. Our contribution is twofold: First, we introduce the *contact transition and object* (CTO) space: The union of a reduced set of contact points and the parameters of approximated objects in the environment. Second, we perform an analysis of the decision boundary between feasible and non-feasible subspace within the CTO space. We hereby focus on a sparse and approximate representation of this boundary, which allows us to discriminate very fast between feasible and non-feasible contact points.

This work can be seen as an additional simplification of planning in the contact space of a robot [1]. It further develops the ideas of [2], which used the swept volume – defined as *"the space, occupied by a robot during the execution"* [3] – to precompute if a motion between two contact points will be feasible. We further advance this precomputation idea by including the geometrical information of objects in the environment.

The paper is organized by first considering related work Section II. Section III will focus on background in contact planning, motion generation and swept volume approximation. In Section IV we introduce the CTO space, Section V discusses the sampling and approximation of the feasibility function, and Section VI demonstrates the applicability of our approach in a highly-cluttered environment.

II. Related Work

We provide in this section a basic overview about contact planning, with emphasis on footstep planning but also on how to construct a general contact space framework. For each approach, we focus on its relation to our work.

Chestnutt et al. [4], [5] pose the problem of footstep planning as a discrete search problem, and are approximating its heuristic by a mobile robot planner. Complementary to their work we use a simple heuristic, and instead focus solely on a fast decision about which steps will be feasible in the present of obstacles.

Escande et al. [1] provide a complete framework for multi-contact planning, in which they investigate how to choose

[1]CNRS, LAAS, 7 avenue du colonel Roche, F-31400 Toulouse, France
[2]Univ de Toulouse, INP, LAAS, F-31400 Toulouse

contact points, and how to generate paths between them. Our work focuses on the first point, for which we provide an approximate solution.

Hauser et al. [6] are planning general multi-contact points for a humanoid climbing robot. Their approach focuses on using motion primitives of contact points as an initial trajectory for a sampling based algorithm. While their work is concerned with finding a probabilistically complete algorithm, we focus on simplifications for real-time planning.

Hornung et al. [7] are using a anytime variant of the A* algorithm to plan footsteps for the Nao robot. This can be seen complementary to our work, in which we try to approximate the feasibility of footstep transitions.

Perrin et al. [2] are using swept volumes to approximate the contact transition between footsteps. While they require the storage of complete swept volumes for collision checking, we devised an approximate mapping from contact points to feasibility by incorporating the object geometry directly into the precomputation process.

III. Background

Our approach is based on three core topics, which will be explained in the following sections: First, we introduce the concept of contact-space planning to reduce the dimensionality of a robotic system in Section III-A. Second, we discuss how the whole-body motion of a robot is generated between two contact points in Section III-B, and finally, we introduce approximation via swept volumes in Section III-C.

A. Planning in Contact Space

Planning a movement for a robotics system, with many degrees of freedom (dof), is commonly seen as intractable, because their complexity is exponential in the number of dof [8]. A simplification, which reduces the planning dimensions, is the contact-space planning approach [4], [1], [6]. Planning is posed as a discrete search problem of finding a sequence of contact-points, which can be used to reach a desired goal region. For transitions between contact-points, local optimization methods can be used. In our work, we will make the further simplification, that contact-points are restricted to footsteps. The long-term goal of our research is the inclusion of hand-environment contacts, which is why we formulate our approach in terms of general contact-points, rather than foot-contacts. We also note, that we are interested in fast real-time planning methods, which is contrary to algorithms which try to find a complete trajectory in the general contact-point space [6], [1]. Earlier research in motion planning made this distinction explicit by dividing algorithms into coarse and fine motion planning [9] — whereby our work can be considered coarse motion planning.

B. Optimal whole-body motion between contact points

For finding a trajectory between two *contact points* $\mathbf{x}_I$ and $\mathbf{x}_G$, we assume that there is an optimization function $p : \mathbb{R}^M \times \mathcal{K} \rightarrow \mathbb{R}^d$, which maps a contact point $\mathbf{x}$, of dimension M, into a joint configuration q, of dimension d, which we will call a *contact configuration*. The space $\mathcal{K}$ defines a behaviour of the robot, i.e. how the rest of the body is positioned. Given one behaviour, and assuming zero noise, the mapping p is uniquely defined, so that we can further operate on contact configurations, without loss of generality. Between two contact configurations q_I and q_G, we then utilize a local optimization function formalized as a classical optimal control problem

$$\begin{aligned} \underset{u}{\text{minimize}} \quad & \int_{t_0}^{t_f} C(u(t), q(t))dt \\ \text{subject to} \quad & \dot{q}(t) = f(q(t), u(t)) \end{aligned}$$

whereby $q(t)$ is the configuration at time t, $u(t)$ is the control input, f is the dynamics of the robot, and C is the cost function, which could depend on the task and the behaviour we want to achieve. We now assume the existence of an algorithm g, which solves the whole-body generation problem between two contact configurations:

$$q_{q_I \rightarrow q_G} = g(q_I, q_G, C) \tag{1}$$

whereby $q_{q_I \rightarrow q_G}$ is the final trajectory of the robot, q_I and q_G are the start and the goal configurations, and C is the above mentioned cost function. Besides being a non-chaotic system, we make no restrictions on the optimization algorithm g and the cost function C. Therefore, we can make use of potential functions [8], nonlinear attractors like the dynamical motion primitive [10], stochastic optimal control solvers [11], or – as in our case – a hierarchical null space control framework, called the stack of task [12]. In this case, we use costs depending on distance to self collisions, distance to joint limits, and dynamical stability.

In the absence of noise, we assume that the optimization problem is uniquely defined, i.e. for a pair of q_I,q_G, optimizer g, and cost function C, g returns one unique trajectory.

C. Swept-Volume Approximations

The unique trajectory from Eq. (1) defines directly a swept volume of the robot body [2], which we will denote as $\mathcal{SV}_{q_I,q_G}$. The number of possible contact transitions is infinite and needs to be reduced to make planning computationally tractable. We therefore use a set of N contact points, which are a discretization of all mechanically feasible footsteps of the robot. This implies the computation of $\binom{N}{2}$ swept volumes (one for each transition pair). By adding a waypoint, as reported in [2], one can assume, that each transition will have a common end point, which prunes the number of swept volumes to N. Using this setting, Perrin et al. [2] have demonstrated real-time motion planning in a constrained environment with fixed upper body and stepping capabilities. Our goal in the next section is to show, how to speed up this approach by (1) introducing the geometry of objects directly into the precomputation algorithm and (2) approximating the decision boundary between feasible and non-feasible space in the joint space of objects and contact points.

IV. Contact Transition and Object (CTO) Space

To plan a discrete set of contacts for a robot, we want to precompute if the transition between two contact points is *feasible*. The feasibility is a function of the environment and the underlying controller. It is therefore necessary to represent the environment, which we do by using an object-centered approach and by fitting generalized geometrical shapes to those objects.

To decide if a contact transition will be feasible, a common approach [2] is to use precomputed swept volumes for each contact transitions and check each swept volume for collisions with all visible objects in scene. In this work, we go one step back and analyse directly the joint space of contact points $\mathcal{X}$ and objects O. Instead of recalling the swept volume and doing collision checking to determine feasibility, our goal is to approximate a feasibility function $f : \mathcal{X} \times O \rightarrow \mathbb{R}$ directly by learning a discriminative function of the form $\hat{f} : \mathcal{X} \times O \rightarrow \mathbb{R}$, such that we minimize the distance between them.

For making this tractable, we apply two simplifications: First, we use a discrete set of contact points $\tilde{\mathcal{X}}$, which was obtained from all possible contact points $\mathcal{X}$ by (A) utilizing the symmetries of the robot body and a waypoint contact as discussed in section III-C, (B) uniformly discretizing contact points from $\mathcal{X}$, and (C) pruning contact points not satisfying internal constraints — like joint limits and self collisions. This provides us with a set of N contact points, which all have the same common goal contact point $\mathbf{x}_G$. For example to go from an arbitrary contact $\mathbf{x}_0$ to another contact $\mathbf{x}_2$, we concatenate $\mathbf{x}_0$ to $\mathbf{x}_G$, and $\mathbf{x}_G$ to $\mathbf{x}_2$. The contact points are a set with an underlying structure, in this case an geometrical ordering (position of contacts) and a metric (distance between contacts). Set and structure define together a mathematical space, such that we can define:

Definition 1 (Reduced Contact-Transition Space): A discrete set of contact points $\mathbf{x}_0, \cdots, \mathbf{x}_N$, which have a common goal contact point $\mathbf{x}_G$

$$\tilde{\mathcal{X}} = \{\mathbf{x}_0, \cdots, \mathbf{x}_N\} \tag{2}$$

In this paper, one contact point is defined as $\mathbf{x} = \{(x, y, \theta, \bar{q})^T | x, y, \theta \in SE(2), \bar{q} \in \{L, R\}\}$, whereby x, y are the middle point of one foot, and θ is the inclination, and $\bar{q}$ is the support foot.

Second, we observe that the detailed shape of an object is not important for coarse motion planning [9], where one is interested in a first reasonable guess of the trajectory. We therefore build the reduced object space $\tilde{O}$ from the complete object space O by assuming that objects can be approximated by basic geometrical shapes. As an intermediate representation between a set of basic shapes (cylinder, sphere, box) and a complete mesh triangle representation, we utilize a generalization of basic shapes, called the superellipsoid. The superellipsoid allows us to describe different basic shapes by one formula with a sparse set of parameters [13]

$$S(x, y, z; \vec{\theta}, \vec{\lambda}) = \left(\left(\frac{x}{\lambda_1} \right)^{\frac{2}{\theta_2}} + \left(\frac{y}{\lambda_2} \right)^{\frac{2}{\theta_2}} \right)^{\frac{\theta_2}{\theta_1}} + \left(\frac{z}{\lambda_3} \right)^{\frac{2}{\theta_1}} \tag{3}$$

whereby $\vec{\theta} > 0$ specifies the shape (e.g. a cylinder), and $\vec{\lambda} > 0$ specifies the elongations along the axes (e.g. the height and radius of a cylinder). Eq. (3) is called the inside-outside function, referring to points x, y, z as being outside the object for $S(x, y, z) > 1$ and inside or on the surface for $S(x, y, z) \leq 1$. Examples include the ellipsoid ($\theta_1 = 1, \theta_2 = 1$), cylindroid ($\theta_1 \ll 1, \theta_2 = 1$) and the quader ($\theta_1 \ll 1, \theta_2 \ll 1$). For this work, we restrict objects to the cylindrical space by defining

Definition 2 (Reduced Object Space): The set of objects $\mathbf{o}$, which can be approximated by a superellipsoid in the form

$$\begin{aligned} \tilde{O} = \{(x, y, \phi, \vec{\theta}, \vec{\lambda})^T | x, y, \phi \in SE(2), \\ \vec{\theta} = (0.01, 1)^T, \\ \vec{\lambda} \in \mathbb{R}_+)\} \end{aligned} \tag{4}$$

Together with the contact points, we can now define the CTO space:

Definition 3 (Contact Transition and Object Space): The union of reduced contact space and reduced object space

$$\mathcal{C}_{\text{CTO}} = \{\tilde{\mathcal{X}} \cup \tilde{O}\} \tag{5}$$

Having defined the $\mathcal{C}_{\text{CTO}}$ space, the rest of the paper is devoted to the computation of the decision boundary between the feasible subspace and the non-feasible subspace. This is formulated as finding a discriminative function $\hat{f}$, which minimizes an optimization problem of the form

$$\begin{aligned} \underset{\hat{f}}{\text{argmin}} \quad & ||f(\mathbf{x}, \mathbf{o}) - \hat{f}(\mathbf{x}, \mathbf{o})||_2 \\ \text{s.t.} \quad & \mathbf{o} \in \tilde{O}, \mathbf{x} \in \tilde{\mathcal{X}} \end{aligned}$$

Whereby f and $\hat{f}$ are computing the feasibility of a contact transition as depicted in Fig. 2: f first optimizes a controller to traverse the contact points, then computes the swept volume along its trajectory and finally conducts collision checking with objects in the environment; $\hat{f}$ simplifies this computation by acting as a discriminative function for the $\mathcal{C}_{\text{CTO}}$ space, to directly decide if a contact transition and an object are in the feasible subspace. In the next section, we will focus on the sampling of f and its approximation $\hat{f}$.

V. Precomputation of Decision Boundary in CTO Space

To estimate $\hat{f}$, we first generate samples from the true feasibility function f. This requires the definition of a probability distribution, which provides us samples near the decision boundary, such that objects have a distance of $d \approx 0$ to the swept volume. A particularity of this distribution is its

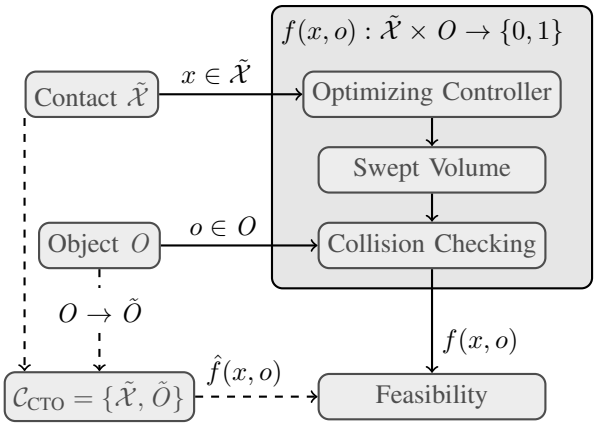


Fig. 2: From Contact Transitions to Feasibility. Dashed lines present the precomputation functions, which form a shortcut for efficient online planning

elongated shape, which requires the usage of a momentum variable to efficiently sample the distribution.

After acquiring samples, we finally discuss the estimation procedure for $\hat{f}$ by using nonlinear discriminative analysis [14].

A. Sampling of the feasibility function

We divide the sampling stage of $\hat{f}$ into two phases: First, we acquire N contact points by using an uniform discretization. We recall, that every contact point has a unique goal, and together with a controller defines implicitly a unique trajectory. The unique trajectory in turn defines a swept volume by using a function $\mathcal{S} : \tilde{\mathcal{X}} \to \mathcal{T}$, whereby $\mathcal{T}$ will be a triangle mesh. The complete set of swept volumes can then be defined as

$$\mathcal{SV}_{1:N} = [\mathcal{S}(\mathbf{x}_1), \ldots, \mathcal{S}(\mathbf{x}_N)] \tag{6}$$

For each swept volume, we start obtaining samples $\mathbf{o}_i \in \tilde{O}$, by defining a probability distribution, which provides us with the properties we want: High probability around the decision boundary, low probability otherwise. One possible choice is the normal distribution, defined as

$$p(x_j, o_i) = \mathcal{N}(d[\mathcal{S}(\mathbf{x}_j), M(\mathbf{o}_i)]; \mu = 0, \sigma) \tag{7}$$

whereby M computes the triangle meshes of the object i at position $\mathbf{o}_i$, $\mathcal{S}(\mathbf{x}_j)$ is the swept volume from contact position $\mathbf{x}_j$, and d is defined as the norm between the nearest points on the object and on the swept volume – or the farthest points inside the swept volume, if the object is in collision. Finally the standard deviation σ is a measurement of how much we tolerate samples away from the boundary.

Eq. (7) is an elongated probability distribution, which can be very narrow, depending on our choice of σ. Therefore, standard sampling techniques like the Markov Chain Monte Carlo (MCMC) algorithm will generally be inefficient, i.e. require too many samples before converging to the stationary distribution [14]. One algorithm, which can handle elongated distributions is the *Hamiltonian Monte Carlo* (HMC) algorithm [15], which adds a momentum variable to the sampling process, in order to faster converge to the stationary distribution. The most important feature of HMC is its ability to follow the contour curve of the distribution by simulating the hamiltonian dynamics. In our case, this translates to following the decision boundary of the $\mathcal{C}_{\text{CTO}}$ space. The underlying algorithm to simulate this contour-curve following behaviour is called the leap-frog algorithm, and progresses by using a number of steps τ and step width ϵ. The initial momentum in a certain direction is defined by a proposal distribution. In our experiments, we used $\epsilon = 0.3$, $\tau = 13$ and a multivariate normal distribution $\mathcal{N}(\mu, \mathbf{\Sigma}) = \mathcal{N}(\mathbf{0}, 0.09 \cdot \mathbf{I})$ as the proposal. For Eq. (7), we have chosen $\sigma = 0.17$.

For simplifications, we consider in our experiments objects approximated by a cylindrical representation, by setting the θ parameters of Eq. (3) to $\theta_1 = 0.01, \theta_2 = 1$. The λ parameters are allowed to vary, and are defined for a cylindrical representation as $\lambda_1 = \lambda_2 = r$ and $\lambda_3 = h$, whereby r is the radius of the cylinder and h the height. Sampling is then conducted explicitly in the space of $\tilde{O} = \{(x, y, r, h)^T | x, y \in \mathbb{R}, r, h \in \mathbb{R}_+\}$.

B. Nonlinear Discriminative Analysis

After obtaining the samples from the true function f, we have to select a model to approximate f by $\hat{f}$. The choice of this model is mainly determined by its online performance: The more often we can call the function per second, the better will be our planning performance. One widely used choice is the *multilayer perceptron* (MLP), which can lead to compact models and faster evaluation, but is harder to train than common kernel machines, because its objective function is non-convex [14]. Because we need to reduce the time for online performance as much as possible, we have chosen the MLP with one hidden layer and trained the network from the sampled data by utilizing the FANN[1] library.

1) Network Optimization: We applied several standard machine learning tricks to obtain a robust and stable approximation of $\hat{f}$. First, we splitted our training data into a training set (70%) and a validation set (30%) and used an early stopping criterion by observing the model error on the validation set. Second, we used multiple restarts with random initializations. Third, we combined two samplers to avoid spurious non-feasible regions: A uniform coarse sampling technique to avoid spurious non-feasible regions, and the aforementioned HMC algorithm to accentuate the decision boundary.

Finally, we summarized the essentials steps of the precomputation in Algorithm 1. For each contact point $\mathbf{x} \in \tilde{\mathcal{X}}$, we first compute the whole-body motion to the waypoint $\mathbf{x}_G$, by using the optimizer g and cost function C. The resulting trajectory $q_{q_I \to q_G}$ defines a swept volume $\mathcal{SV}$, which we approximate by using a function $\mathcal{S}$. For the class of objects $\tilde{O}$, we acquire M samples $\mathbf{o}_{1:M}$ by using the HMC sampling algorithms with parameters τ and ϵ. Afterwards, we split our sampling data and start the nonlinear

[1] http://leenissen.dk/fann/

Algorithm 1 Precomputing feasible motion space

Require: $\mathcal{C}, \tilde{O}, \tau > 0, \epsilon > 0, H > 0, M > 0$
  function PRECOMPUTE($\mathcal{C}, \tilde{O}, M, H, \tau, \epsilon$)
    $\mathcal{F} \leftarrow \varnothing$
    for all $\mathbf{x} \in \tilde{\mathcal{X}}$ **do**
      $q_{q_I \to q_G} \leftarrow g(\mathbf{x}, \mathbf{x}_G, \mathcal{C})$ [12]
      $\mathcal{SV} \leftarrow \mathcal{S}(q_{q_I \to q_G})$ [2]
      $\mathbf{o}_{1:M} \leftarrow$ Sampler$(\mathcal{SV}, \tilde{O}, M, \tau, \epsilon)$ [15]
      $\mathbf{o}_{train}, \mathbf{o}_{validation} \leftarrow$ split$(\mathbf{o}_{1:M})$
      $\hat{f} \leftarrow$ NDA$(H, \mathbf{o}_{train}, \mathbf{o}_{validation})$ [14]
      $push(\mathcal{F}, \hat{f})$
    end for
  end function

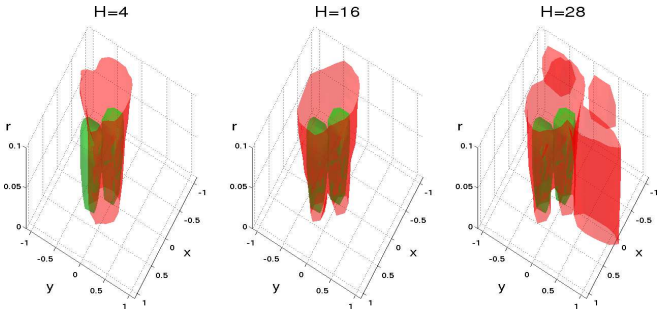


Fig. 3: Influence of the model complexity on the approximated feasibility function: Each graph shows the object space $\tilde{O}$ for the same swept volume with changed complexity parameter H. For visualization, we have shown the non-feasible regions for the $(x, y, r)^T$ parameters of a cylinder, whereby we fixed $h = 0.03$. Shown is the isosurface of the feasibility function for the zero value, first the real feasibility function (green), and second the approximated function $\hat{f}$ (red). Depending on the complexity parameter H of the model we can observe different performances: a low complexity like $H = 4$, leads to an underfitting of $\hat{f}$, while a high complexity $H = 28$ leads to overfitting, visible by several spurious non-feasible regions. The goal is to find a parameter, like $H = 16$, which balances model complexity and error rate.

discriminative algorithm to approximate $\hat{f}$. $\hat{f}$ is finally saved in our complete feasibility structure $\mathcal{F}$.

C. Algorithmic analysis

The offline-precomputation of the feasibility function required ~6 hours on a 8 core, 3.0Ghz PC with 8GB working memory. The online performance requires the computation of two matrix multiplications in our MLP, and therefore scales with $\mathcal{O}(H * (N + 1))$, whereby H is the complexity parameter and N the number of dimensions of $\tilde{O}$. At the moment, we have no theoretical guarantee that the algorithm is strictly conservative, i.e. that it declares a footstep as valid, if it is not. We could approach this by proving that the derivative of the feasibility function is bounded, i.e. K-lipschitz continuous, and using this as a hard constraint during the optimization of the approximated model.

VI. Experiments

In our experiments, we use a feasibility function $\hat{f}$ with a reasonable model complexity of $H = 16$, which avoids under- and overfitting, as discussed in Fig. 3. We refer to this

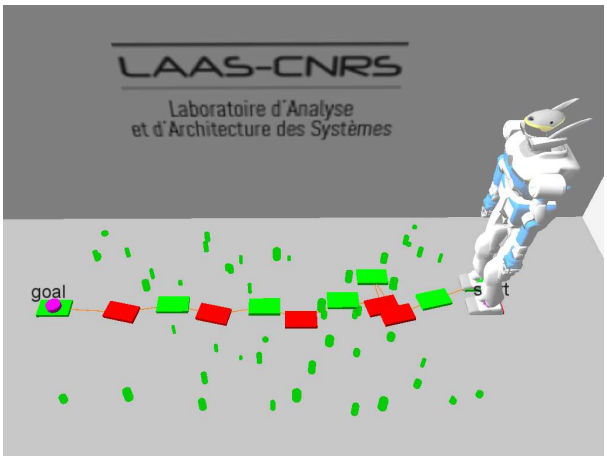


Fig. 4: A cluttered environment, which we consider in our experimental verification. A number of cylinders are used as obstacles, and determines the complexity of the scene. In a real world setting, those cylinders would correspond to approximations of objects, similar to the chair in Fig. 1.

algorithm as $FP(16)$, whereby FP stands for *feasibility precomputation*. For comparison, we use a reimplementation of the swept volume approximation (SVA) algorithm, proposed by [2], which stores swept volumes for each action, and afterwards used a collision checking algorithm for feasibility checks [16]. Both algorithms are integrated into our planning framework, and tested in a challenging environment, where we randomly place objects.

A. Planning

For planning, we utilize a standard A* algorithm [8] with a classical euclidean l_2-norm heuristic to the goal. The heuristic is complementary to our work: We focus not on the heuristic, but on approximating the extension of nodes in the graph search. The choice of the heuristic can further speed up planning [4], but is beyond the scope of this work.

B. Walking in Cluttered Environment

To evaluate and compare the performance of our feasibility precomputation, we consider a highly-cluttered and constrained environment, where K small objects are located randomly over a flat, horizontal floor, as visualized in Fig. 4. We generate the objects by using a uniform sampler $\mathcal{U}$ and bounding cylinders in the form of $x = \mathcal{U}(-0.8m, 0.8m), y = (0.2m, 2.8m), r = \mathcal{U}(0.01m, 0.03m), h = \mathcal{U}(0.01m, 0.1m)$.

The robot is allowed to set footsteps, which are constrained to be between $x = [-0.8m, 0.8m]$ and $y = [-0.2m, 3.2m]$.

The planning tasks is to move the feet, starting with the left foot at coordinates $(x_I, y_I) = (0, 0)$, towards the goal at $(x_G, y_G) = (3, 0)$, i.e. having one foot in the vicinity $(< 0.3m)$ of the goal. We compare the two approaches, mentioned above: For the SVA algorithm, we obtain the cylinders as triangle meshes from the simulator and store them offline for efficient collision checks. For $FP(16)$, we use the x, y, r, h values as the input for $\hat{f}$. Before each execution, we apply a homogeneous transformation to move the object into the coordinate system with the support foot as origin, such that they coincide with the precomputation stage.

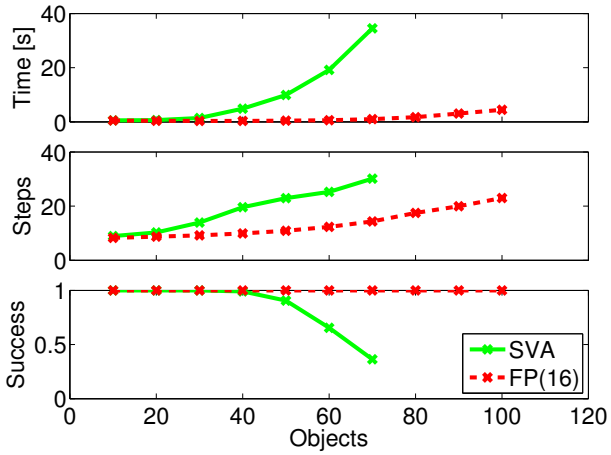


Fig. 5: Comparison between swept volume approximations (SVA) [2] (green) and the precomputation of the feasibility function (red). Each point represents the average over 100 trials in the cluttered environment situation, where the robot had to traverse a distance of $3.0m$, while avoiding M objects, randomly distributed on the floor.

Moreover we prune all objects, which have a certain distance to the robot ($< 1.1m$) before we apply the algorithms.

Fig. 5 shows the performance of the two algorithms on this task: In the first row, we show the average planning time for successful plans versus the number of objects in the scene. It can be seen, that the time for planning with the SVA algorithm (green) increases rapidly with the number of objects. In comparison, our algorithm (red) increases only marginally and stays bounded by $< 1s$ even for $N = 60$ objects. Also, we obtain a lower number of steps toward the goal as seen in the second row. The last row shows the success rate of the planner, i.e. after a fixed time T, we stop the planner and consider the task unsuccessful. Those tasks are cleared from our system and are not considered for the time and step graphs.

VII. Conclusion

In this work, we presented the *contact transition and object* space, a joint space of contact points and geometrical information of approximated objects. We developed an algorithm to precompute the feasibility of specific objects and contact points, by approximating the decision boundary between feasible and non-feasible subspaces. As a result we obtain a sparse discriminative function, which allowed us to quickly prune non-feasible contact-points – while at the same time preserving the important stepping-over capability of humanoid robots.

In our simulated experiments, we demonstrated that our approach can be used to generate whole-body motions for a humanoid robot in highly-cluttered environments in near real-time, thereby outperforming a state-of-the-art algorithm, which used swept volume approximations. Moreover, our algorithm has a significantly lower memory fingerprint: instead of saving the complete swept volumes, we require only the model parameters of our discriminative function to be saved. This comes at the price of a lower accuracy at run-time: due to the approximation of the objects by simple geometrical shapes, we lose the ability to move close to objects and conduct for example fine-manipulation planning. However, for certain behaviors like walking, fine-manipulation is per se not required. Also, we see our method as a first reasonable guess of the trajectory, which could be further refined locally.

Possible future research directions are the incorporation of object velocities into the precomputation, the estimation of the decision boundary for the general superellipsoid space of objects, the augmentation of the action space and the verification on our robotics platform using vision systems. As a natural extension we are going to consider multi-contact point planning and the implementation of different control strategies, such that we can switch very fast between behaviors like walking, crouching and holding objects while walking.

Acknowledgment

The research in this paper was supported by the KoroiBot EU-FP7 Project, OSEO/Romeo2 and the French Ministry of Higher Education and Research.

References

[1] A. Escande, A. Kheddar, and S. Miossec, "Planning contact points for humanoid robots," *Robotics and Autonomous Systems*, vol. 61, no. 5, pp. 428 – 442, 2013.

[2] N. Perrin, O. Stasse, L. Baudouin, F. Lamiraux, and E. Yoshida, "Fast humanoid robot collision-free footstep planning using swept volume approximations," *IEEE Transactions on Robotics*, vol. 28, no. 2, 2012.

[3] P. Jiménez, F. Thomas, and C. Torras, "Collision detection algorithms for motion planning," in *Robot Motion Planning and Control*, J.-P. Laumond, Ed. Berlin: Springer-Verlag, 1998, pp. 1–53.

[4] J. Chestnutt, "Navigation and gait planning," in *Motion Planning for Humanoid Robots*, K. Harada, E. Yoshida, and K. Yokoi, Eds. Springer London, 2010, pp. 1–28.

[5] J. Chestnutt, M. Lau, K. M. Cheung, J. Kuffner, J. K. Hodgins, and T. Kanade, "Footstep planning for the honda asimo humanoid," in *(ICRA)*, 2005.

[6] K. K. Hauser, T. Bretl, K. Harada, and J.-C. Latombe, "Using motion primitives in probabilistic sample-based planning for humanoid robots," in *WAFR*, ser. Springer Tracts in Advanced Robotics, S. Akella, N. M. Amato, W. H. Huang, and B. Mishra, Eds., vol. 47. Springer, 2006, pp. 507–522.

[7] A. Hornung, D. Maier, and M. Bennewitz, "Search-based footstep planning," in *ICRA Workshop on Progress and Open Problems in Motion Planning and Navigation for Humanoids*, Karlsruhe, Germany, 2013.

[8] S. M. LaValle, *Planning Algorithms*. Cambridge, U.K.: Cambridge University Press, 2006.

[9] Y. K. Hwang and N. Ahuja, "Gross motion planning – a survey," *ACM Comput. Surv.*, vol. 24, no. 3, pp. 219–291, 1992.

[10] A. J. Ijspeert, J. Nakanishi, H. Hoffmann, P. Pastor, and S. Schaal, "Dynamical movement primitives: Learning attractor models for motor behaviors," *Neural Computation*, vol. 25, no. 2, pp. 328–373, 2013.

[11] M. Toussaint, "Robot trajectory optimization using approximate inference," in *ICML*, 2009, p. 132.

[12] N. Mansard, O. Khatib, and A. Khedar, "A unified approach to integrate unilateral constraints in the stack of tasks," *IEEE Transaction on Robotics*, vol. 25, no. 3, June 2009.

[13] A. Barr, "Superquadrics and angle-preserving transformations," *Computer Graphics and Applications, IEEE*, vol. 1, no. 1, pp. 11–23, 1981.

[14] C. M. Bishop, *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag New York, 2006.

[15] R. M. Neal, "MCMC using Hamiltonian dynamics," in *Handbook of Markov Chain Monte Carlo*, S. Brooks, A. Gelman, G. Jones, and X. Meng, Eds. Chapman and Hall/CRC Press, 2010.

[16] E. Larsen, S. Gottschalk, M. C. Lin, and D. Manocha, "Fast proximity queries with swept sphere volumes," Technical Report TR99-018, Department of Computer Science, University of North Carolina, Tech. Rep., 1999.