Motion Primitive Based Random Planning for Loco–Manipulation Tasks

Alessandro Settimi[1,2], Danilo Caporale[1], Przemyslaw Kryczka[2], Mirko Ferrati[1], Lucia Pallottino[1]

***Abstract*— Several advanced control laws are available for complex robotic systems such as humanoid robots and mobile manipulators. Controls are usually developed for locomotion or for manipulation purposes. Resulting motions are usually executed sequentially and the potentiality of the robotic platform is not fully exploited.**

In this work we consider the problem of loco–manipulation planning for a robot with given parametrized control laws known as primitives. Such primitives, may have not been designed to be executed simultaneously and by composing them instability may easily arise. With the proposed approach, primitives combination that guarantee stability of the system are obtained resulting in complex whole–body behavior.

A formal definition of motion primitives is provided and a random sampling approach on a manifold with limited dimension is investigated. Probabilistic completeness and asymptotic optimality are also proved. The proposed approach is tested both on a mobile manipulator and on the humanoid robot Walk-Man, performing loco–manipulation tasks.

I. Introduction

One of the main problems when planning motion of complex robotics systems is the curse of dimensionality. In case of robots able to move in and interact with the environment, such as humanoid robots (see Fig. 1 for an example) and mobile manipulators, the problem is even worse. For loco–manipulation tasks several approaches have been proposed in the literature with the goal of reducing such complexity.

The majority of reduction techniques may be classified as whole–body ([1]–[6]) and primitive based planning ([8],[9]).

It is well known that whole–body planning considers the whole dynamical system resulting in high dimensional, numerically intractable problems. Sample based approaches that usually perform better than other classical approaches (such as potential fields, exact and approximated decomposition methods, see e.g., [1]) in higher dimensional spaces are, however, prohibitive. The main idea behind whole–body approaches is the limitation of the solution search space based on system and tasks constraints. For example, in [2], [3] the sampling procedure is biased on constrained submanifolds of the configuration space of lower dimension.

A whole–body contact sampling algorithm has been proposed in [4], where the transitions between contacts are planned to switch from a stance (a robot configuration in space) to another, hence reducing the complexity of the problem. In [5] the whole–body jacobian based method *Stack of Tasks* is used to tackle the problem of grasping an object while walking using visual servoing. Once the priorities of the tasks (*i.e.,* balancing, locomotion, manipulation) are defined, the developed controller tries to execute lower priority tasks while guaranteeing execution of higher priority ones. This is done by trying to execute lower priority tasks in the *nullspace* of the higher priority tasks. Similarly, in [6], a jacobian based approach is used to decouple the manipulation from the locomotion tasks.

[1]Centro di Ricerca "E. Piaggio", Università di Pisa, 56122 Pisa, Italy.
[2]Dep. of Advanced Robotics, Istituto Italiano di Tecnologia, via Morego, 30, 16163 Genova, Italy
Corr. auth.: `alessandro.settimi@for.unipi.it`

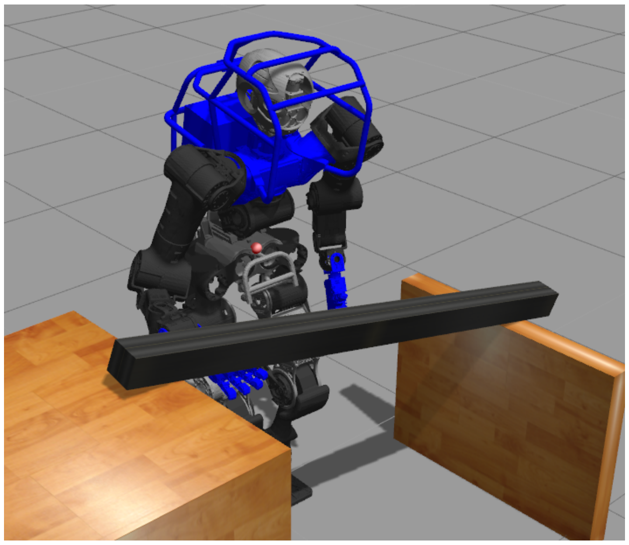


Fig. 1: The Walk-Man robot walking and manipulating an object in the Gazebo simulation environment.

A completely different approach is to develop control laws for particular tasks that do not fully exploit the potentiality of the robot. For example, advanced control laws are available to perform objects manipulation or to walk on rough terrains. Indeed, a variety of not integrated different behaviors is available while there is still a lack of complex whole–body controls. A remarkable approach is hence to exploit and integrate such available control laws to create complex behaviors. In this way such manipulation, locomotion, balancing, and other control laws can be combined without dramatically increasing the complexity of the problem. Such techniques are known as primitive based approaches. In [7], high-quality motion primitives are exploited in order to avoid sampling in the whole configuration space. Contact planners, such as those proposed in [4], [7] can be seen as particular whole–body primitives that can be used with other control laws in the herein proposed approach.

In [8] the problem of pushing an object on a flat surface is

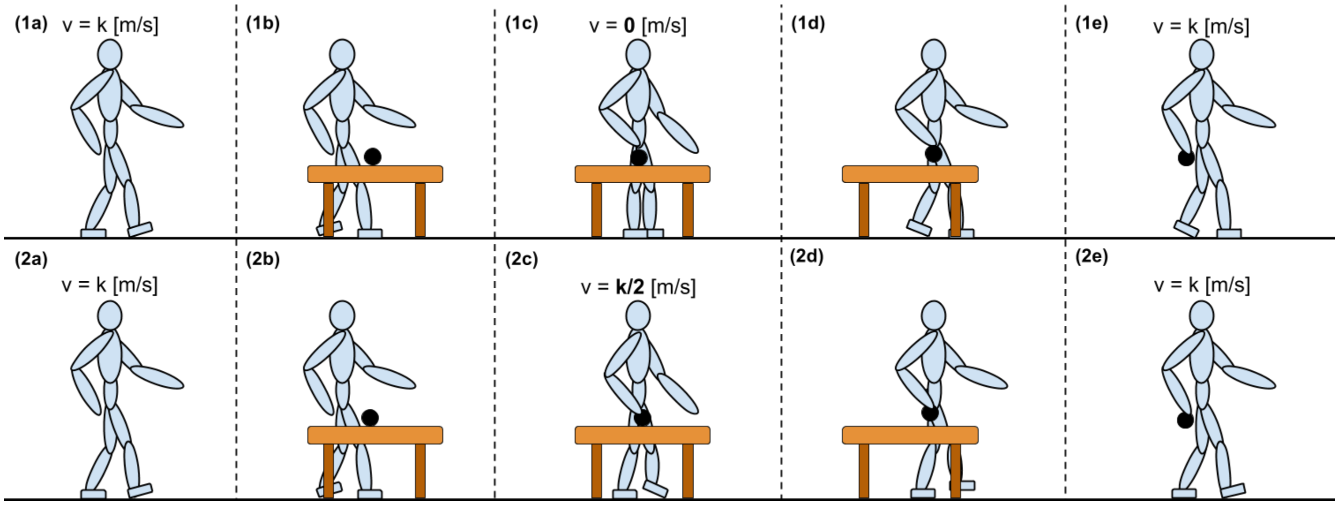


Fig. 2: The P-Search* basic idea. The task is to go from the start position to the goal one loading an object placed on a table. In subfigures 1(a-b-c-d) a humanoid robot approaches an object, stops, loads it, and then starts moving again. Subfigures 2(a-b-c-d) show the same task executed without stopping but just slowing down near the object.

faced. The possible actions (walking, reaching and pushing) are considered separately and executed in a sequential way, i.e., each action is performed only after the previous one is terminated.
Similarly to the Stack of Tasks proposed in [5], methods in [9] exploit motion primitives in a hierarchical framework where the priority of the various primitive has to be chosen in order to execute a certain task. System stability and safety depend on these priorities whose choice is hence critical and task dependent.

In this paper we propose a primitive-based approach to combine primitives. The implementation shown here is derived from RRT* [10]. With respect to RRT [11] and other random graph based algorithms, such as Informed RRT [12], we do not change the sampling space with heuristics but we use the information available from the primitives design to structure a sampling space with desirable properties. Differently from other similar methods, given a set of primitives we provide a technique to combine them in parallel, thus we aim at finding plans that use more than one primitive simultaneously, without jeopardizing the stability of the system. Our method is able to jointly use different primitives that were not, in principle, designed to be combined. With respect to the literature, the proposed approach is independent on how primitives are designed and executed. Only basic assumptions on the stability and robustness of the considered control laws are required. Requested properties on the primitives make the proposed approach, named *P-Search** in the following, independent from the robotic platform it is applied to (such as humanoids, mobile manipulators, mobile wheeled robots). Motion primitives for humanoid robots may involve manipulation or locomotion, and may use different control approaches. Some examples of manipulation primitives are: turn a valve, open a door, grasp an object, bi-manually manipulate an object; while locomotion primitives can represent dynamic walking, static walking, crawling gaits, stairs climbing.

The *P-Search** algorithm samples on the union of the primitives image spaces that correspond to submanifolds of the configuration space. *P-Search** will be proved to be able to provide complex plans in which primitives are combined. For example, let's consider a humanoid robot on which two primitives have been developed: a locomotion primitive to move in the environment, and a manipulation primitive to grasp objects with the hand. In Fig. 2(1a–1e) the two primitives are used sequentially, the robot stops in front of the object, grasps it and then moves away. An idea of the *P-Search** outcome is reported in Fig. 2(2a–2e), where the robot slows down without stopping its motion to safely catch the object while guaranteeing stability. The obtained behaviour is not possible with a sequential combination of the primitives nor with a direct parallel combination of the primitives that can easily provide unstable behaviours. Hence a whole–body control approach (a single complex primitive) or a *P-Search**-like algorithm should be used.

In Section II the class of considered systems is characterized and primitives and primitives image space are formally defined. In Section III the *P-Search** algorithm is described and proved to be probabilistic complete and asymptotic optimal in Section IV. Finally, experimental results are reported in Section V to show the validity of the approach.

II. Formalization

Consider the dynamical system Σ expressed by equations

$$\left\{ \begin{array}{lcl} \dot{x}(t) & = & f\left(x(t), u(t)\right) \\ y(t) & = & h\left(x(t)\right), \end{array} \right. \tag{1}$$

where $x \in X \subseteq \mathbb{R}^n$ is the state of the system, $u \in \mathcal{U} \subseteq \mathbb{R}^m$ is the control vector, $y \in \chi \subseteq \mathbb{R}^p$ is the output vector, $f : \mathbb{R}^n \times \mathbb{R}^m \to X$ and $h : \mathbb{R}^n \to \chi$ are respectively the dynamics and the output transformations of the system. We address the problem of finding a trajectory between two given states minimizing a given cost function, such as time of execution or energy efficiency. Control inputs associated to the obtained trajectory must also be determined.

Definition 1: We define a generic *motion primitive* π as a 6-tuple $\pi(q,\, \chi,\, \sigma,\, T,\, \xi, C)$ with

- $q \in Q$: the parameters that characterize the primitive;
- χ: the image space of the primitive that corresponds to the image space of the output function of the dynamical system;
- $\sigma\ :\ X \times Q\ \rightarrow\ \chi$: the **steering function** of the primitive that is a set–valued function based on the system dynamics from the primitive space to the image space; it can be a map on $(0,1)^d$, with $d \geq 2$;
- $T \in \mathbb{R}_{\geq 0}$: the duration of the execution of the primitive;
- $\xi = \rho(t,y), \rho : \mathbb{R}_{\geq 0} \times \chi \rightarrow \Xi = \{0,1\}$: a trigger that enables the execution of the primitive, where t is the time variable;
- $C : \mathbb{R}_{\geq 0} \times X \times Q \rightarrow \mathbb{R}$: the cost function associated with the primitive.

Let $\mathbb{P}$ be the finite set of the generic primitives. The parameters q, C, χ and σ characterize the control law u associated to the motion primitive. A similar definition of a simpler control entity can be found in [13] where an *atom* (u, ξ, t) is characterized by a control law, a trigger and an execution time.

In case of complex robotics systems such as humanoids or mobile manipulators two approaches can be used to determine a motion primitive for the whole system: a highly complex (whole–body) primitive based approach or a concatenation of simpler primitives defined on decoupled subsystems of lower dimension. The first approach suffers from the curse of dimensionality, hence several simplifications are usually made to tackle the problem obtaining motion primitives that do not fully exploit the system potentiality. In the latter case, for example, a decoupled locomotion and manipulation approach does not take advantage of the whole–body capabilities of the robotic platform.

For this reason, in this paper we consider systems that can be partitioned in N coupled subsystems as: $x = [x_1'\, x_2'\, \dots\, x_N']$, $u = [u_1'\, u_2'\, \dots\, u_N']$ and $y = [y_1'\, y_2'\, \dots\, y_N']$, where the dynamics can be expressed as:

$$\left\{ \begin{array}{rcl} \dot{x}_1 & = & f_1(x, u_1) \\ y_1 & = & h_1(x) \in \chi_1 \\ \dot{x}_2 & = & f_2(x, u_2) \\ y_2 & = & h_2(x) \in \chi_2 \\ \vdots & & \\ \dot{x}_N & = & f_N(x, u_N) \\ y_N & = & h_N(x) \in \chi_N . \end{array} \right. \tag{2}$$

It is worth noting that, even if each sub-system $\Sigma_i = \{\dot{x}_i = f_i(x, u_i),\, y_i = h_i(x) \in \chi_i\}$ depends on a different control variable it also depends on the whole state variable $x \in \mathbb{R}^n$. A motion primitive π_i is defined for each subsystem Σ_i since it has direct effects on Σ_i and undirected effects on all other the subsystems Σ_j with $j \neq i$, through the evolution of the state variable x. Another important aspect of this formalization is that the primitives set associated to a system can also involve only a subset of its degrees of freedom.

The main idea of the proposed approach is to use motion primitives π_i for the N subsystems Σ_i as local steering functions in classical sample based planning algorithms to obtain a plan for the whole system Σ. One of the basic assumption on the motion primitive π_i is that the associated control law u_i is a stabilizing policy for system Σ_i while all other $N-1$ controls u_j are null (null controls generate steady motions or *trim trajectories*, see [14]). Moreover, feasibility conditions on the primitives concurrency must be carefully checked (e.g., in the case of humanoid robots, a check on the ZMP-condition [15] can be used for this purpose).

One of the main strengths of our approach is that we do not sample in the whole state space X of Σ, as in RRT, RRT*, PRM [16], but only in each primitive image space, thus reducing dimensionality.

A tree $\mathcal{T}$, whose vertices are points $z \in \chi$, will iteratively grow on the union of the primitive image spaces that must hence be connected. Indeed, in order to connect samples on an image space χ_j to the growing tree, χ_j must intersect at least one of the image spaces on which the tree lays. In other words, given the set $\{z_i\}_1^k \in \mathcal{T}$ of samples in the tree, it is possible to sample in the image space χ_j only if there exists $z_h \in \mathcal{T}$ laying on the image space χ_h with $\chi_j \cap \chi_h \neq \emptyset$.

At each iteration i, there exist a non-empty set of active primitives (whose trigger conditions ξ_i are verified), denoted as $\mathbb{P}_A$, and sampling is done in the union of the image spaces, denoted as χ_A. Sampling in each image space is allowed after a trigger condition ξ_i has been activated, which corresponds to connect a sample in $\chi_i \cap \chi_A$ to the tree $\mathcal{T}$. For example, a grasp primitive is not activated until the object to be grasped is sufficiently close.

Remark 1: Let $\mathcal{P} = (V, E)$ be the graph whose nodes in V are associated to motion primitives π_i. An edge in E between nodes π_i and π_j exists if $\chi_i \cap \chi_j \neq \emptyset$. A motion planning problem can be solved with the primitive-based sampling algorithm P-Search* only if the graph $\mathcal{P}$ is connected. Indeed a tree $\mathcal{T}$ connecting any start and goal configurations can be constructed based on samples only if the union of the primitive image spaces is connected. This condition corresponds to a connected graph $\mathcal{P}$.

III. The P-Search* algorithm

In this Section, we describe the *P-Search** planning algorithm to connect initial and goal states z_I and z_G respectively, which lie in the obstacle-free region of two possibly different image spaces. For example from a standing robot in a position to a pose in which the robot has an object in its hands.

As in many other planning algorithms, alternative versions could be obtained to grow the tree $\mathcal{T}$ starting from the goal and back–chaining the primitives to the initial state or, alternatively, starting two branches from the beginning and the end state and trying to connect them (see [17]). In this paper, only the forward growth is described.

Let $\chi_{i,obs}$ be the obstacle region in the primitive image space, such that $\chi_i \setminus \chi_{i,obs}$ is an open set. In the following, for notation convenience, we refer to the generic obstacle free image space $\chi_{i,free}$ as χ_i.

The initial state z_I is assumed to be in a feasible condition, which means that it exists an active primitive $\pi_i \in V$ with an associated image space χ_I and $z_I \in \chi_I$.

A. P-Search*

Algorithm 1: $\mathcal{T} \leftarrow$ `P-Search`$^*(z_I, z_G)$

```
Data: $\mathcal{P} = (V, E), z_I, z_G$
Result: $\mathcal{T}$ a tree whose vertices are points $z \in \chi$. Given
        two vertices $z_i$, $z_j \in \chi_k$ an edge $(z_i, z_j)$ is an
        instantiation of the primitive $\pi_k \in V$ that steers
        $z_i$ toward $z_j$ in $\chi_k$.
1 $\mathcal{T} \leftarrow$ InsertNode($\emptyset, z_I, \mathcal{T}$);
2 $z_{new} = z_I$;
3 for $i = 1$ to $N$ do
4     $\mathbb{P}_A \leftarrow$ ActivePrimitives($z_{new}$);
5     $\chi_i \leftarrow$ SamplePrimitive($\mathbb{P}_A$);
6     $z_{rand} \leftarrow$ Sample($\chi_i$);
7     $(z_{new}, \mathcal{T}) \leftarrow$ LocalRRT*($\chi_i, z_{rand}, \mathcal{T}$);
8 return $\mathcal{T}$;
```

The P-Search* algorithm, given initial and goal states z_I, z_G, provides a tree $\mathcal{T}$ whose vertices are points $z \in \chi$. Given two vertices z_i, $z_j \in \chi_k$ an edge (z_i, z_j) is an instantiation of the primitive $\pi_k \in V$ that steers z_i toward z_j in χ_k.

Line 1: An empty tree $\mathcal{T}$ is created and populated with the initial point.

ActivePrimitives: Given the newly added point z_{new}, the `ActivePrimitives` function computes all the primitives π_j such that $z_{new} \in \chi_j$ and such that z_{new} satisfies π_j trigger conditions ξ_j. The set of such primitives is returned and stored in $\mathbb{P}_A$.

Primitive Sampling: The `SamplePrimitive` function return a random primitive from the set $\mathbb{P}_A$.

Sampling: The `Sample` function takes as argument the obstacle free image space χ_k from where the random sample $z \in \chi_k$ is extracted.

LocalRRT:* Given an image space χ_k, a sample $z \in \chi_k$ and the tree $\mathcal{T}$, the `LocalRRT`* algorithm tries to add the sample z to the tree. A detailed description of the procedure is reported in Section III-B. The outcome of this function is a node z_{new} and an updated tree $\mathcal{T}$ where z_{new} has been added if a collision–free path in χ_k from an existing node in $\mathcal{T}$ to z_{new} is found. Moreover, the path is inserted in the tree set of edges with its associated cost.

B. `LocalRRT*`

The procedure `LocalRRT`* is now described in detail.

Nearest: Given an image space χ_k a sample $z \in \chi_k$ and the tree $\mathcal{T}$, the `Nearest` function returns the vertex $z_{nearest} \in \chi_k$ of the tree, which is the nearest to the random point z. By construction, for each point in χ_k there exists at least a vertex of $\mathcal{T}$ inside the same image space.

Algorithm 2: $(z_{new}, \mathcal{T}) \leftarrow$ `LocalRRT`$^*(\chi_k, z, \mathcal{T})$

```
1  $z_{nearest} \leftarrow$ Nearest($\chi_k, z, \mathcal{T}$);
2  $(z_{new}, x_{new}) \leftarrow$ Steer($z_{nearest}, z$);
3  if Unfeasible($x_{new}$) then
4      return $(-, \mathcal{T})$;
5  if ObstacleFree($x_{new}$) then
6      $Z_{near} \leftarrow$ Near($z_{new}, \mathcal{T}$);
7      $z_{min} \leftarrow$ ChooseParent($\mathcal{T}, Z_{near}, z_{nearest}, z_{new}$);
8      $\mathcal{T} \leftarrow$ InsertNode($z_{min}, z_{new}, \mathcal{T}$);
9      $\mathcal{T} \leftarrow$ Rewire($\mathcal{T}, Z_{near}, z_{min}, z_{new}$);
10     return $(z_{new}, \mathcal{T})$;
11 else
12     return $(-, \mathcal{T})$;
```

Steer: Given the sample node z and the nearest vertex $z_{nearest} \in \mathcal{T}$ the `Steer` function first computes a point z_{new} between $z_{nearest}$ and z. Moreover, it attempts to connect $z_{nearest}$ with z_{new}. Note that the function is image space–dependent, hence it attempts to connect two points inside the same image space χ_k. If any, the path between points $z_{nearest}$ and z_{new} is returned. The `Steer` function is the implementation of the steering function σ of the primitive π_k in Definition 1.

Unfeasible: the function `Unfeasible` checks whether the trajectory x_{new} computed by function `Steer` violates additional conditions such as kinematic or dynamics constraints (e.g., ZMP stability conditions). Such function guarantees that the parallel primitives composition does not lead to unstable behaviours.

ObstacleFree: a check on the trajectory returned by the `Steer` function is performed by the function `ObstacleFree`. In case of failure the sample z_{new} is discarded, otherwise the `Near` routine is called.

Near: the `Near` function returns the set Z_{near} of the tree nodes that are in a ball centered in the current sample z_{new} of radius dependent on the current cardinality of the nodes of the tree (see. [10] for details). Such nodes must lay in one of the image spaces where z_{new} lays on.

ChooseParent: the set Z_{near} is used to choose the node in $\mathcal{T}$ that will become the parent node of z_{new} (named z_{min}). This is done by applying the `Steer` function from any node in Z_{near} toward z_{new}. z_{min} is the starting point of the minimum cost path.

Rewire: this function performs in the same manner as in [10], considering all the possible primitives for rewiring. The rewiring procedure allows to add and remove arcs in the tree to obtain lower cost paths from the start point z_I to existing nodes in Z_{near} through z_{new}.

IV. Probabilistic Completeness and Asymptotic Optimality

Here we investigate on the properties of asymptotic optimality and probabilistic completeness, which are fundamental for sample-based algorithms to achieve optimal results. Since our algorithm is implemented as a modified version of RRT* [10] we retain many properties from previous work and we can use them to demonstrate the aforementioned properties.

It is well known that probabilistic completeness holds for RRT and RRT* algorithms, [18], [19]. Here we provide the same result for P-Search*. In what follows, we indicate with μ the probability measure over a sampling space χ. Probabilistic completeness and optimality properties of RRT* follow from the property of robust feasibility of paths (i.e., obtained paths are at least at $\delta > 0$ distance from the obstacles). The robust feasibility property is verified also by P-Search* algorithm thanks to a proper implementation of the `ObstacleFree` function (if a sample or a path is closer than δ from an obstacle it is discarded). Hence we are now able to prove the following properties.

Theorem 1: If the primitive graph $\mathcal{P}$ is connected, the P-Search* is probabilistically complete.

Proof: First we note that, using just one primitive, the algorithm reduces to RRT* since the whole sampling space is coincident with the image space of that primitive, χ_I. Hence, given any two samples in the same image space the probability to find a path between the samples by applying P-Search* tends to 1.

Consider now a pair of samples (z_i, z_j) in different image spaces, χ_i and χ_j. Since the primitive graph $\mathcal{P}$ is connected, there always exists a sequence of pairwise intersecting image spaces from χ_i to χ_j. For the probabilistic completeness on any image space the thesis holds. □

Theorem 2: P-Search* is asymptotically optimal.

Proof: Asymptotic optimality of RRT* follows from the choice of the radius γ used in the RRT* Near function. Since the number of primitives is finite, it is sufficient to consider the largest radius while implementing the P-Search* function `Near`. □

V. Results

Both simulation and experimental tests have been conducted on a mobile manipulator (consisting of an iRobot Create 2 mobile platform, with a 4 joints OWI-535 Robotic Arm mounted on board) and on the simulated model of the humanoid robot Walk-Man [20], demonstrating the capability of our approach on different robots with their own primitives (see Fig. 3).

P-Search* has been implemented in MATLAB to validate the effectiveness of the algorithm when performing off-line planning for the aforementioned robots.

The developed code for the P-Search* implementation is available online[1], together with all the open source code related to the Walk-Man controllers[2] and the mobile manipulator ones[3].

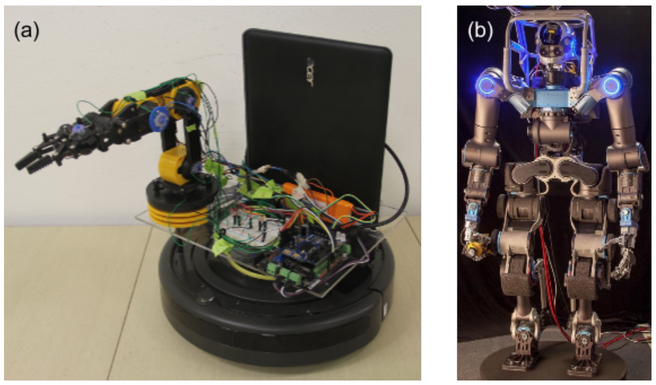


Fig. 3: The robotic platforms used in the experiments.

Consider the problem of moving a mobile robot in the environment and to interact with an object through manipulation, thanks to one or more robotic arms equipped with hands or grippers. For this problem the full state space is $(x, y, \theta, v, q_1, \ldots q_N)$, where (x, y) is the position on the plane, θ is the heading of the robot, v is the robot walking speed and q_i, $i = 1 \ldots N$ are the arm joint coordinates.

For this purpose two primitives have been defined:
L: the locomotion primitive, to move the robot on the XY plane, characterized by the following parameters:

- $q_L = \emptyset$,
- $\chi_L \ni [x\, y\, \theta\, v]^T$,
- σ_L, an optimization routine, applied on a simplified dynamics, minimizes the time variable t subject to state and control constraints and returns the robot desired trajectory,
- $C_L = t$,
- T_L, is the duration of execution of the steering function σ_L,
- $\xi_L = 0$ until a sample laying in χ_L is added to $\mathcal{T}$.

M: the manipulation primitive, to move the end-effector towards the object and manipulate it, characterized by the following parameters:

- $q_M = o$, where o is the object pose,
- $\chi_M \ni [x\, y\, \tau]^T$,
- σ_M, the inverse kinematics of the robotic arm, giving the joints desired values corresponding to a certain value of o and τ,
- $C_M = t$,
- T_M, is the duration of execution of the steering function σ_M,
- $\xi_M = 1$ when $\|o - r\| \leq \delta$, with r the pose of the robot and $\delta > 0$, otherwise it is 0.

τ can be set as the rate of completion of the manipulation task in terms of distance between the end–effector and the object. For a grasping task $\tau = 1$ means that the end–effector is gripping the object, and τ is equal to 0 when the

[1]https://github.com/CentroEPiaggio/primitives

[2]https://gitlab.robotology.eu/walkman-drc
[3]https://github.com/MirkoFerrati/irobotcreate2ros

end–effector is beyond the maximum distance to perform a successful grasping. This is formalized by the M primitive trigger condition.

Note that, with the proposed approach it is possible to sample in a 4-dimensional space for the locomotion primitive and in a 3-dimensional space for the manipulation one instead of sampling in the full $4+N$-dimensional configuration space of the robot, as τ is used to sample only one scalar instead of the entire joint space of the robotic arm. This is even more convenient when dealing with a higher number of DoFs. This is one of the advantages of our method, where the trajectory generator computes the joint values corresponding to the value of τ and the *P-Search** algorithm checks for feasible primitives composition.

In Fig. 4 the image spaces related to the proposed scenario are depicted, note that they can represent any generic loco–manipulation task. Let's suppose that the problem is to reach a location where to execute a manipulation task (e.g. grasp an object) and come back close to the starting position. As we can see in the figure, χ_L is represented both for $\tau = 0$ and $\tau = 1$, and the tree connecting the starting point to the goal is made by combination of L primitives (solid lines) and M primitives (dotted lines). It is worth noting that dotted lines can involve both locomotion and manipulation resulting in a loco–manipulation behaviour. Also, note that performing a manipulation action with the M primitive while moving in the (x, y) plane is allowed only when a sample in χ_L is connected to a sample in χ_M which had nonzero walking speed v. This is coherent with the idea that the manipulation primitive does not affect the locomotion image space directly, but indirectly through the effect of a constant control (*trim trajectory*) on the locomotion primitive.

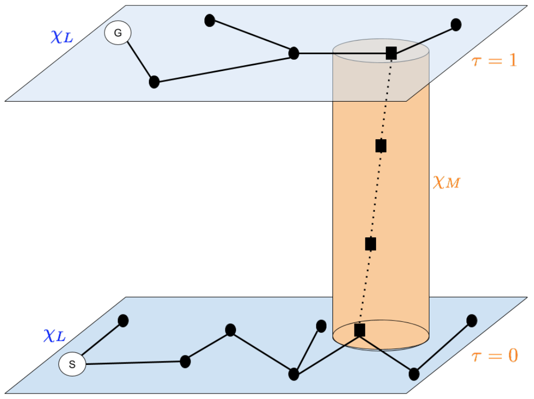


Fig. 4: The image spaces representation of a generic loco–manipulation task.

Although the aforementioned primitives are equivalent in term of planning, the implemented control laws are different and depend, of course, on the robot type. More details about the primitives related control laws are given in the following.

Mobile Manipulator - Locomotion: Given the plan solution's waypoints (x, y, θ, v, τ), in case of activation of the L primitive, x, y, θ and v are used to generate the mobile robot trajectory, optimizing the execution time for a unicycle model.

Mobile Manipulator - Manipulation: Given the plan solution's waypoints (x, y, θ, v, τ), in case of activation of the M primitive, τ is used to generate the end–effector desired position with respect to the target object pose, then Inverse Kinematics is applied to obtain the joints desired values.

Humanoid Robot - Locomotion: In this case x, y, θ and v are used to generate the foot placement and feet trajectories necessary to arrive at the desired position. Based on the foot placement the ZMP reference trajectory is generated and used by the Preview Controller [21] to compute the COM reference trajectory. During execution we employ an implementation of the feedback controller described in [22] to additionally stabilize the robot around the precalculated pattern.

Humanoid Robot - Manipulation: This primitive is equivalent to the one of the mobile manipulator, except for the kinematics of the arm. The Walk-Man robot arm has 7 DoF, thus redundancy can be exploited.

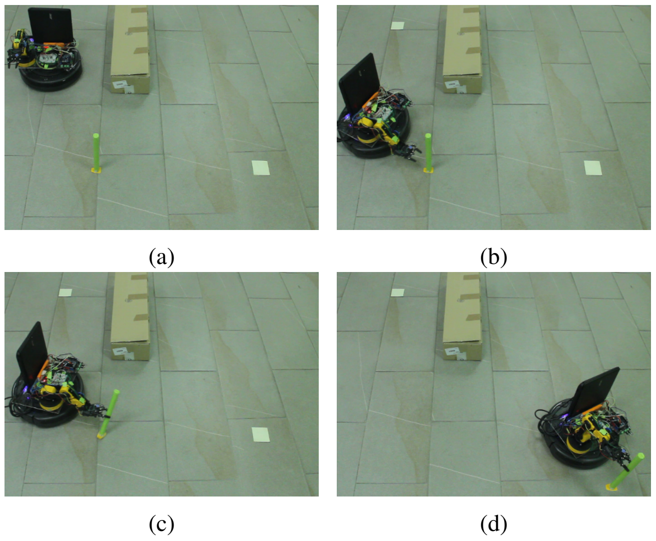
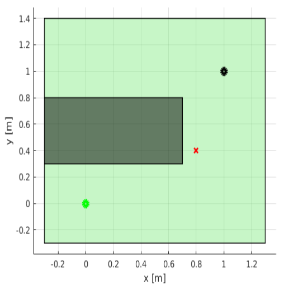


Fig. 5: Snapshots of the performed loco–manipulation experiment.

In Fig. 5 an experiment for the mobile manipulator involving an obstacle is reported. The robot approaches the object using the locomotion primitive (Fig. 5a), then it slows down and maintains this primitive active at constant velocity while using the manipulation primitive to grab the object (Fig. 5b,5c). At the end, the manipulation primitive is maintained at constant (null) velocity and the locomotion primitive is used to bring the object to the desired position (Fig. 5d).

In Fig. 6 the corresponding image spaces seen from above are reported. The obstacle is reported as the grey zone in the sampling space.

In Fig. 6b the sampling space of the related experiment and the primitives image spaces are shown. As we can see, to perform the task, the tree grows from the locomotion primitive image space with $\tau = 0$ to the one at $\tau = 1$, passing


(a) Sampling space of experiment depicted in Fig.5. The obstacle has been enlarged to take the robot dimension into account.

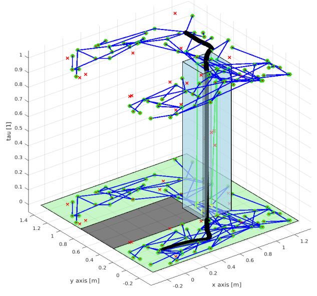

(b) 3D view of the sampling space of experiment depicted in Fig. 5.

Fig. 6

through the manipulation primitive image space, represented as a cuboid. The black line represents the final plan generated for the robot.

To further test our approach we implemented the scenario shown in Fig. 7, to perform a complex loco–manipulation task using the Walk-Man robot in the Gazebo simulation environment. Here the task consists in removing an object while the robot is walking avoiding obstacles.

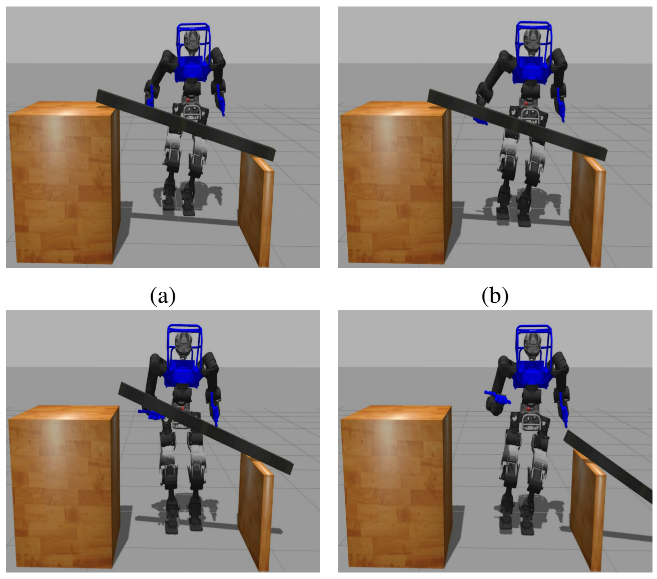

(a) (b) (c) (d)

Fig. 7: Walk-Man robot performing a loco–manipulation experiment in the Gazebo simulation environment. The robot first walks to the object, and then starts to manipulate it while walking until it is removed.

The representation of the environment is reported in Fig. 8, where the green, blue, and black dots represent respectively the starting position, the object to manipulate and the goal position.

In Fig. 9 the plan found by *P-Search** is shown. The manipulation task accomplishment is represented by the tree growing in the vertical axis, in fact, the start position is in $(0, 0, 0)$ while the goal position is in $(1.5, 0, 1)$.

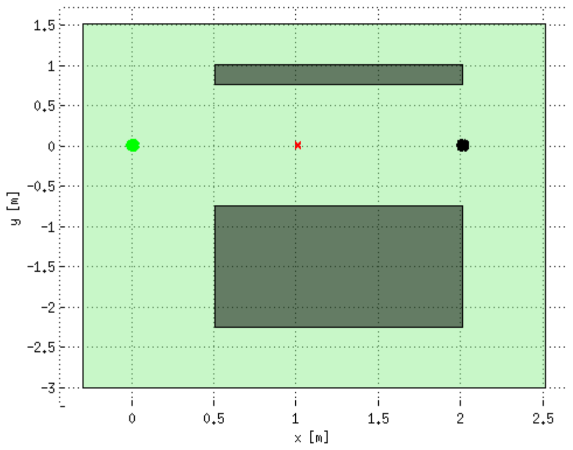


Fig. 8: Sampling space of loco–manipulation experiment reported in Fig. 7.

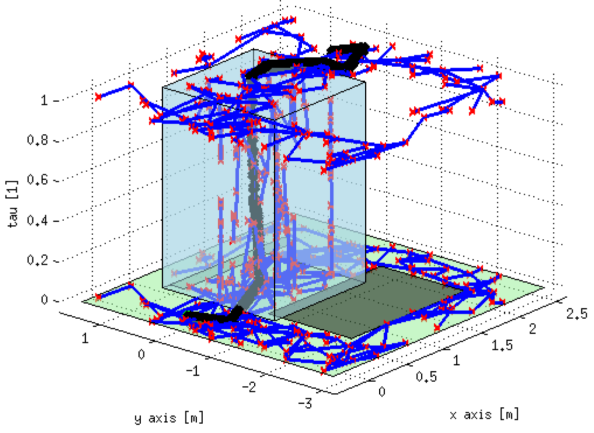


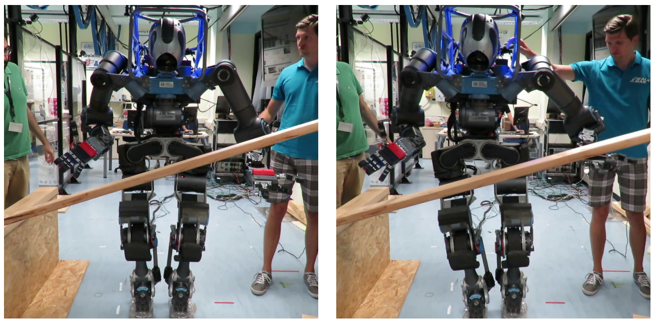
Fig. 9: 3D view of the sampling space of experiment depicted in Fig. 7.

As we can see in Fig. 7, the robot executes the task similarly to the mobile manipulator. Walk-Man walks to reach the object, then it starts to manipulate it while walking until the object is removed and the passage can be safely crossed.

The same experiment has been performed successfully on the Walk-Man robotic platform and it is reported in Fig. 10.

VI. Conclusions

In this work we proposed a novel planning approach based on motion primitives. The parallel stable composition of primitives potentially not designed to work simultaneously leads to complex whole–body behaviors. Future work is devoted to a hierarchical planning framework capable to modifying dynamically the set of available primitives depending on the surrounding environment (recognized objects,


(a) (b)

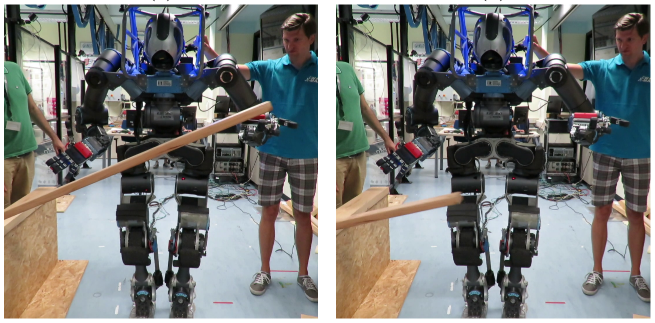

(c) (d)

Fig. 10: Walk-Man robot performing a loco–manipulation experiment. The robot performs the experiment in the same way as the simulated one.

type of terrain, ...). Further tests on the real Walk-Man robot are being conducted, moreover current work is focused on combining primitives with overlapping degrees of freedom. In this case the stability condition is not enough to guarantee the feasibility of the simultaneous execution of different primitives. Additional properties have to be investigated to allow the combination of such primitives.
The planner is now used to generate offline reference trajectories, and robustness is provided by the single primitive controllers. An efficient implementation of the code is being tested in order to allow for on-line re-planning in case of failures.

Acknowledgments

This work is supported by the European commission project Walk-Man EU FP7-ICT no. 611832 and the ECs Horizon 2020 robotics program ICT-23-2014 under grant agreement 644727 (CogIMon). Authors would like to thank Salvatore Gerratana and Gaspare Santaera for their contribution for the mobile manipulator experimental setup. Authors would like to thank Corrado Pavan for his contribution in *P-Search**. Finally, authors would like to thank Federico Spinelli for his help in the perception part.

References

[1] Steven M LaValle. *Planning algorithms*. Cambridge university press, 2006.
[2] Felix Burget, Anja Hornung, and Maren Bennewitz. Whole-body motion planning for manipulation of articulated objects. In *Robotics and Automation (ICRA), 2013 IEEE International Conference on*, pages 1656–1662. IEEE, 2013.
[3] Sébastien Dalibard, Antonio El Khoury, Florent Lamiraux, Alireza Nakhaei, Michel Taïx, and Jean-Paul Laumond. Dynamic walking and whole-body motion planning for humanoid robots: an integrated approach. *The International Journal of Robotics Research*, page 0278364913481250, 2013.
[4] Karim Bouyarmane and Abderrahmane Kheddar. Humanoid robot locomotion and manipulation step planning. *Advanced Robotics*, 26(10):1099–1126, 2012.
[5] Nicolas Mansard, Olivier Stasse, François Chaumette, and Kazuhito Yokoi. Visually-guided grasping while walking on a humanoid robot. In *Robotics and Automation, 2007 IEEE International Conference on*, pages 3041–3047. IEEE, 2007.
[6] Marco Cognetti, Pouya Mohammadi, Giuseppe Oriolo, and Marilena Vendittelli. Task-oriented whole-body planning for humanoids based on hybrid motion generation. In *Intelligent Robots and Systems (IROS 2014), 2014 IEEE/RSJ International Conference on*, pages 4071–4076. IEEE, 2014.
[7] Kris Hauser, Timothy Bretl, Kensuke Harada, and Jean-Claude Latombe. Using motion primitives in probabilistic sample-based planning for humanoid robots. In *Algorithmic foundation of robotics VII*, pages 507–522. Springer, 2008.
[8] Kris Hauser, Victor Ng-Thow-Hing, and Hector Gonzalez-Baños. Multi-modal motion planning for a humanoid robot manipulation task. In *Robotics Research*, pages 307–317. Springer, 2010.
[9] Luis Sentis and Oussama Khatib. Synthesis of whole-body behaviors through hierarchical control of behavioral primitives. *International Journal of Humanoid Robotics*, 2(04):505–518, 2005.
[10] Sertac Karaman, Matthew R Walter, Alejandro Perez, Emilio Frazzoli, and Seth Teller. Anytime motion planning using the rrt*. In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, page 1478. IEEE, 2011.
[11] Steven M LaValle. Rapidly-exploring random trees: A new tool for path planning. 1998.
[12] Jonathan D Gammell, Siddhartha S Srinivasa, and Timothy D Barfoot. Informed rrt*: Optimal sampling-based path planning focused via direct sampling of an admissible ellipsoidal heuristic. *arXiv preprint arXiv:1404.2334*, 2014.
[13] Dimitrios Hristu-Varsakelis, Magnus Egerstedt, and Perinkulam S Krishnaprasad. On the structural complexity of the motion description language mdle. In *Decision and Control, 2003. Proceedings. 42nd IEEE Conference on*, volume 4, pages 3360–3365. IEEE, 2003.
[14] Emilio Frazzoli, Munther Dahleh, Eric Feron, et al. Maneuver-based motion planning for nonlinear systems with symmetries. *Robotics, IEEE Transactions on*, 21(6):1077–1091, 2005.
[15] Miomir Vukobratović and Branislav Borovac. Zero-moment point—thirty five years of its life. *International Journal of Humanoid Robotics*, 1(01):157–173, 2004.
[16] Lydia E Kavraki, Petr Švestka, Jean-Claude Latombe, and Mark H Overmars. Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *Robotics and Automation, IEEE Transactions on*, 12(4):566–580, 1996.
[17] James J Kuffner and Steven M LaValle. Rrt-connect: An efficient approach to single-query path planning. In *Robotics and Automation. Proceedings. ICRA'00. IEEE International Conference on*, volume 2, pages 995–1001. IEEE, 2000.
[18] Sertac Karaman and Emilio Frazzoli. Sampling-based algorithms for optimal motion planning. *The International Journal of Robotics Research*, 30(7):846–894, 2011.
[19] Steven M LaValle and James J Kuffner. Randomized kinodynamic planning. *The International Journal of Robotics Research*, 20(5):378–400, 2001.
[20] F. Negrello, M. Garabini, M. G. Catalano, P. Kryczka, W. Choi, D. G. Caldwell, A. Bicchi, and N. G. Tsagarakis. Walk-man humanoid lower body design optimization for enhanced physical performance. In *IEEE International Conference of Robotics and Automation (ICRA2016)*, pages 1817 – 1824, Stockholm, Sweden, May 16-21, 2016, 2016. IEEE, IEEE.
[21] Shuuji Kajita, Fumio Kanehiro, Kenji Kaneko, Kiyoshi Fujiwara, and Kensuke Harada Kazuhito Yokoi. Biped walking pattern generation by using preview control of zero-moment point. In *Robotics and Automation (ICRA), IEEE International Conference on*, pages 1620–1626, 2003.
[22] P. Kryczka, P. Kormushev, N. G. Tsagarakis, and D. G. Caldwell. Online regeneration of bipedal walking gait pattern optimizing footstep placement and timing. In *Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on*, pages 3352–3357, Sept 2015.