Integrating Physics-Based Prediction with Semantic Plan Execution Monitoring*

Sebastian Rockel[1], Štefan Konečný[2], Sebastian Stock[3,4], Joachim Hertzberg[3,4], Federico Pecora[2], Jianwei Zhang[1]

***Abstract*— Real-world robotic systems have to perform reliably in uncertain and dynamic environments. State-of-the-art cognitive robotic systems use an abstract symbolic representation of the real world for high-level reasoning. Some aspects of the world, such as object dynamics, are inherently difficult to capture in an abstract symbolic form, yet they influence whether the executed action will succeed or fail. This paper presents an integrated system that uses a physics-based simulation to predict robot action results and durations, combined with a Hierarchical Task Network (HTN) planner and semantic execution monitoring. We describe a fully integrated system in which a Semantic Execution Monitor (SEM) uses information from the planning domain to perform *functional imagination*. Based on information obtained from functional imagination, the robot control system decides whether it is necessary to adapt the plan currently being executed. As a proof of concept, we demonstrate a PR2 able to carry tall objects on a tray without the objects toppling. Our approach achieves this by simulating robot and object dynamics. A validation shows that robot action results in simulation can be transferred to the real world. The system improves on state-of-the-art AI plan-based systems by feeding simulated prediction results back into the execution system.**

I. Introduction

Robots are set to move from well-structured factory floors to a much more chaotic environment: private homes occupied by naive users with little understanding of the strengths and limitations of robotic systems. To operate safely (for all involved parties) and efficiently in such adverse circumstances, the robot needs to understand the environment.

To illustrate some of these problems, we employ the following example throughout the paper. A robot has to deliver a pepper mill. It can place the pepper mill on a tray, but the mill might topple over during driving. One solution to this problem is to drive 'slowly'. But how does the abstract qualitative term 'slow' translate into precise velocity and acceleration values that the robot can process? What are the odds of the pepper mill toppling nonetheless? How much more likely is toppling when driving 'fast'? How should the 'drive' action be parametrized to bring an object safely to its destination? These questions are hard to answer in general terms as they depend upon the platform and environment.

Any approach employing symbolic abstraction to capture the real world is likely to require discretization of inherently continuous domains (time, space) and will reduce the amount of detail considered. But how much detail can be omitted before the representation becomes unrealistic in the target application? The same issues are relevant for research into realistic physics simulation.

In this work, we propose to combine a realistic physical simulation with symbolic reasoning. The main focus is on how to integrate *functional imagination* [1] using the HIRES framework [2] with an off-the-shelf task planner and using Semantic Execution Monitoring (SEM) to simulate an action before it is executed. A plan created by the task planner determines which actions are to be simulated and provides a causal precondition-effect relation for each action. These plan action relations will be used by a SEM to generate the context of the action relevant for imagination – the initial state imagined. From functional imagination we obtain approximate information about expected action duration and toppling likelihood. To use imagination, we must decide how to express abstract qualitative terms (i.e. 'slow', 'topple') in sufficient detail for simulation and how the simulation is to be performed. The SEM uses the expected duration obtained from imagination for temporal reasoning and scheduling – both beyond the scope of this paper.

Next we introduce the evaluation scenario and discuss related work; we then describe the integrated system and the experimental results before ending with a summary.

A. Scenario

In our demonstration scenario, the robot fetches an object from a counter and carries it on a tray to the destination. Task planning and symbolic reasoning consider only an abstract description of the world state and the expected changes resulting from the execution of a task such as 'carry object'. Task execution is atomic – it may succeed or fail. In contrast, the parametrization of physical aspects of the transition between abstract symbolic states is typically the domain of a physical simulation. By considering the trajectory, velocity and acceleration, a simulation can accurately model the carrying of an object and can determine the parameters required to prevent toppling.

To evaluate our approach, we used the following scenario: the robot must carry a pepper mill from a restaurant counter to a table (cf. Fig. 1). To prevent the pepper mill from toppling, a working parametrization (velocity and acceleration settings) for the `move_base` robot action must be 'imagined' in simulation. This action parametrization, in combination with the time the action takes in simulation, is

*This work was supported by the EC Seventh Framework Program theme FP7-ICT-2011-7, grant agreement no. 287752 and no. 288899.

[1]University of Hamburg, Germany, {rockel,zhang}@informatik.uni-hamburg.de

[2]Örebro University, Sweden, {sky,fpa}@aass.oru.se

[3]Osnabrück Univ., Germany, {sestock,jhertzbe}@uos.de

[4]DFKI Robotics Innovation Center, Osnabrück Branch, Germany

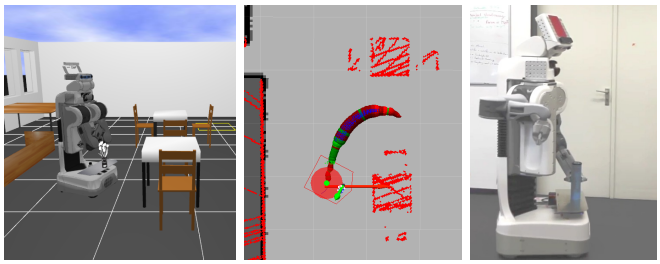

Fig. 1: Left: PR2 approaching pre-manipulation area left of table1 in the simulation. Center: Trace marker thickness indicates speed, color indicates acceleration (green: positive; red: negative); the red circle indicates the actual rotation center and radius of the tray object., red squares represent obstacles. Right: PR2 placing the pepper mill on its tray.

used by the reasoning modules to take appropriate measures. If a simulated action parameterization is more likely to succeed than that chosen by the planner, the SEM will adapt the plan. Thus functional imagination is used to predict action outcomes and allow adaption prior to execution.

II. Related Work

Prior work has attempted to integrate physics-based simulation into plan based robot control and to integrate dedicated arm motion planning into task planning. For example, [3] integrates a physics-based simulation for object classification. [4] employs a physics engine to efficiently re-plan in changing environments. [5], [6] use physics-based simulations for manipulation in cluttered environments.

[7] evaluates qualitatively a number of freely available physics engines for simulation systems and game development. Among the ten physics engines evaluated, our work mainly uses the Open Dynamics Engine (ODE), on which the Gazebo simulator used here is based. The accuracy and performance of physics engines when calculating friction, collisions and constraints are important for predicting robot action results in a typical domestic domain. To handle potential variations in system behavior, the proposed prediction model has to take even minor variations and deviations from 'true' into account.

[8] used the term *robot imagination* for a system generating models of objects prior to their perception. Robot imagination is defined here as the robot's capability to generate feature parameter values of unknown objects by generalizing characteristics from previously presented objects. [9] describes *functional imagination* as the purposeful manipulation of information that is not directly available to the senses (of the robot) and states that imagination always relates to something that in reality is not there. In order to be useful, imagination also needs to provide information about the likely consequences of actions. This is then to be used by a mechanism for translating imagined motor actions into sensory-based representations of their consequences. In [1], the authors propose a framework for modeling functional imagination: an embodied agent that simulates its own behaviors, predicts their sensory-based consequences and extracts behavioral benefit from doing so. The system is based on the physics-based humanoid simulator SIMNOS [10].

Recently much work has been done to extend robots' understanding of their environments. Feeding this information into the planning process is beneficial for many reasons. Symbolic planners such as HTN planners [11] were combined with geometric or motion planners to obtain more detailed and executable plans. [12] interleave geometric and HTN-based task planning. A hybrid planning approach that jointly reasons about temporal, spatial and resource knowledge is presented in [13]. There the goal is to model spatio-temporal relations between objects in a table-top setting and reason about this information to execute actions to produce the desired table layout.

Another approach is to learn robot action parametrization a-priori and offline [14], [15]. However, our approach has an advantage in that rather than attempting to learn a model for action parametrization in general, imagination replicates the specific context of the considered action.

III. Approach

The following assumptions are made for the approach presented below: A toppling threshold is empirically defined, together with a discrete motion-parameter-set. The standard simulation friction coefficient is kept. Furthermore, discrete topple thresholds are chosen for computational efficiency.

A. Architecture

Before detailing the modules involved, we will sketch how functional imagination is integrated into the robot control architecture developed in the RACE[1] project (for details, cf. [16]). Fig. 2 shows an excerpt, including the integration of our additional modules. All our modules query a common exchange medium, employed in this architecture, called the Blackboard.

Without imagination, the nominal system workflow is as follows. When a new task is given to the task planner, the planner generates a plan based on the current state retrieved from the Blackboard. This plan, which consists of actions representing basic capabilities of the robot, is sent to the SEM (Fig. 2 combines the planner and execution monitor). The SEM monitors preconditions and effects of plan actions and dispatches actions when appropriate. It also maintains a temporal network, which represents temporal expectations about execution [17], e.g., an action's earliest possible start or finish time. This network is updated with observations and information resulting from functional imagination. This allows for the seamless integration of imagination with reasoning about temporal constraints and with rescheduling.

Actions are dispatched by sending them to the Execution Middlelayer, which calls the appropriate Robot Operating System (ROS) actions implementing the basic robot capabilities.

To allow the use of functional imagination, this workflow needed several extensions. Some planned actions should be

[1] http://project-race.eu

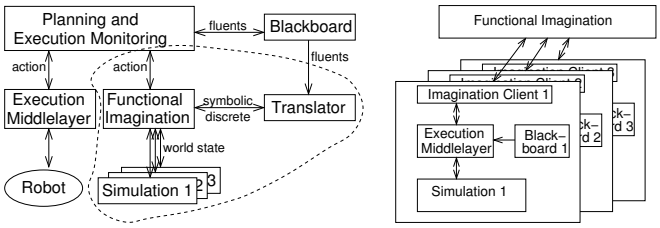

Fig. 2: Left: integration of functional imagination (highlighted) into the RACE architecture (excerpt). Right: functional imagination: detailed view of the modules running on external computers for improved performance.

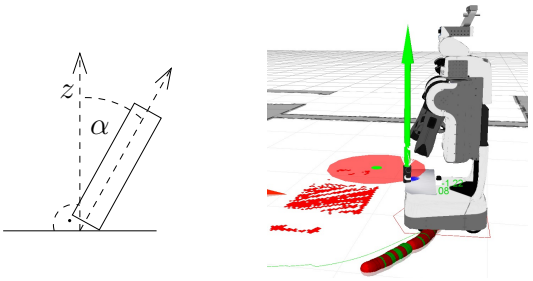

Fig. 3: (left) Object deviation angle α. (right) Visualization of the tray object, its current orientation and the robot (from simulation).

	θ_{fast}	θ_{slow}
v_{linear} $[m{\cdot}s^{-1}]$	0.55	0.30
$v_{angular}$ $[rad{\cdot}s^{-1}]$	1.00	1.00
a_{linear} $[m{\cdot}s^{-2}]$	2.50	1.40
$a_{angular}$ $[rad{\cdot}s^{-2}]$	3.20	2.00

TABLE I: Experimental results: motion parameter sets (θ).

simulated with a variety of parameters before being executed on the real robot. The plan therefore needs to specify the actions to be simulated, at which step this should be done and the parameters that should be tested. To this end, we introduce a dedicated *imagine* action, to be discussed in Sec. III-D.

The SEM has to analyze the plan and send each action to be tested to the Imagination Client instead of the Execution Middlelayer. As we wish to simulate action execution at a later plan stage, the *current* robot and environment states are not appropriate; the SEM must send the Imagination Client the states projected to exist at that later plan stage.

Furthermore, the Imagination Client needs to know the poses of all objects and of the robot, together with the robot's configuration, in order to initialize the simulation accordingly. This is achieved by the Translator module, which gets the projected symbolic world state and returns it in a format suitable for input to the simulation. We let the SEM generate the expected states and call the Translator to decouple the Imagination Client from the symbolic representation; in this way, imagination can be integrated more easily into other systems.

The Imagination Client is provided with possible parameters that should be tested. For each parameter combination, the Imagination Client starts a separate subset of the robot control system on a different machine, i.e., each with an independent ROS system, as presented in Fig. 2.

B. Functional Imagination

The prediction system used in this work is based on our previously developed HIRES framework [2]. It was integrated into the RACE framework and extended to predict common-sense physics events and to handle uncertainty.

Model and scenario related definitions will now be introduced.

α is the smallest angle between the perpendicular up-axis (z) of the global reference frame and the principal axis of the tray object, as shown in Fig. 3. A topple state is a tuple of topple events, $s_t : \{notopple, topple, shaking\}$, which are defined in Eq. 1. The thresholds are defined as follows: $\alpha_n = 0.01\,rad$ ($\approx 0.5°$), $\alpha_t = 0.5\,rad$ ($\approx 29°$). The threshold angles were defined empirically and are automatically and continuously measured during experiments. Furthermore, any toppling is automatically detected, and the tray object is returned to its start position to detect more potential topple events. For the experimental scenario, a cylindrical model (the pepper mill) of height $27.4cm$ and radius $3.7cm$ with a point mass of $140g$ was modeled to match the real object. The object inertia was defined appropriately.

$$x = \begin{cases} notopple, & if\ \alpha \leq \alpha_n \\ topple, & if\ \alpha \geq \alpha_t \\ shaking, & if\ \alpha_n < \alpha < \alpha_t \end{cases} \quad (1)$$

To run the scenario, two sets of motion parameters θ_{fast} and θ_{slow} were defined, as shown in Table I. The parameter set elements are needed in the local motion planner; the individual elements are defined by the standard values for $fast$ and by empirical minimum values for $slow$. The latter were chosen to allow a maximal difference from the $fast$ set, while still allowing the local path planning algorithm and motion controller to execute a trajectory smoothly. Whereas lower and upper limits are constrained by physics, it would be possible to use continuous motion parameters provided by an optimization scheme to determine the most appropriate parameter values for a given task.

The Dynamic Window Approach (DWA, [18]) is used for local path planning. It considers current and projected velocity and acceleration settings for linear and angular directions. One important property of the ROS implementation is its support for dynamic changes to parameters such as velocity and acceleration.

Arbitration between the competing simulations is based on several factors. If the action itself returns a failure, the simulation is eliminated. If it is successful, the confidence measure c is evaluated (cf. confidence definition in Eq. 2); higher confidence c wins. Finally, the duration measure T_{action} is taken into account to favor faster execution.

$$c = \left(\frac{a_t \cdot c_t + a_s \cdot c_s + a_n \cdot c_n}{a_t + a_s + a_n} \right)^{-1} \quad (2a)$$

$$c = (0, 1] := \{c \in \mathbb{R}^+ | 0 < c \leq 1\}, a_t + a_s + a_n > 0 \quad (2b)$$

	θ_{fast}	θ_{slow}
all events	179054	171717
notopple	165906	165596
shaking	12735	6031
topple	413	90
duration [s]	18090 (∼5h)	17350 (∼4.9h)

TABLE II: Experimental results: simulated toppling scenario measuring topple events for $fast$ and $slow$ parameter sets.

X	$\mathcal{L}(\theta_{fast}\|x)$	$\mathcal{L}(\theta_{slow}\|x)$	ratio (f/s)
$notopple$	0.9266	0.9644	≈1
$shaking$	0.0711	0.0351	≈2
$topple$	0.0023	0.0005	≈5

TABLE III: Experimental results: simulated toppling likelihood $\mathcal{L}$ for $fast$ and $slow$ parameter sets and the relation between them: ratio (f/s).

The coefficients are: c_t (topple), c_s (shaking), c_n (no event); the variables are: a_t-topple, a_s-shaking, a_n-no event. To keep the confidence between 0 and 1 and to scale well between the three possible events, the following coefficient values were defined empirically from the data of Table II and Table III: $c_t = 100$, $c_s = 10$, $c_n = 1$. The duration of an action is defined simply as $T_{action} = t_{end} - t_{start}$ with $T_{action\_max} := 300s$.

The Simulation Client monitors toppling events by querying the object's deviation angle at a rate of approximately 20 Hz. Depending on the measured angle, the counter of one of the three possible events is increased. If an action fails, its corresponding confidence value is set to 0. If no action (parametrization) was successful or if imagination failed to complete in time (either because of a predefined deadline or because the SEM can wait no longer and must execute the action), a standard value, i.e., $fast$, is inserted to instantiate the complete plan.

C. Simulation Validation

Based on the previous definitions, the following validation experiments were executed in simulation: the robot had to drive between fixed positions, repeatedly accelerating and stopping. Fig. 5 visualizes these positions, which correspond to pre-manipulation areas (PMAs) at pieces of furniture in the environment: tables (T1, T2) and a counter (C1). Multiple PMAs belonging to the same piece of furniture are distinguished by respective cardinal directions, e.g., ST1 for south and NT1 for north. The nearStartArea1 (NSA1) robot starting area was not included in the testing.

The robot's velocity and acceleration and the tray object's deviation were continuously tracked. As expected, moving $slow$ly resulted in significantly fewer topple events, as Table II shows. In this experiment, the robot approached every PMA (Fig. 5) in the simulation using both fast and slow parameter sets while tracking the topple-state.

Based on these results, the likelihood of the three distinct events is given by $\mathcal{L}(\theta|x) = P(X = x; \theta)$. Here, x is defined by the three distinct values and θ is the (motion) parameter-set. Table III shows the likelihoods, which are calculated as $\mathcal{L}(\theta|x) = x \cdot (\sum_{i=1}^{N} \theta_i)^{-1}$. Although the $topple$ ratio between fast and slow results is ≈ 5, toppling is not guaranteed to happen in any time period. Its probability by time of occurrence is once every $\approx 44s$, whereas a $shaking$ occurs every $\approx 1.4s$ with θ_{fast}.

D. Planning

For plan generation, we used the off-the-shelf HTN planner SHOP2 [19]. The planner was integrated into the RACE system as described in [20], which also describes how the domain is modeled. HTN planning [11] is a hierarchical planning approach that works by decomposing compound tasks into sub-tasks. Decomposition continues until only so-called 'primitive' tasks are left - those that the robot can execute directly. These primitive robot capabilities are given to the planner as *operators* representing action preconditions and effects, as in classical planning. HTN planning also employs *methods* to decompose compound tasks into sub-tasks. The operators and methods are given to the planner in a *planning domain*.

The planner is given a goal task and generates a plan as a sequence of ground instances of operators (actions) according to the planning domain and the current state of the Blackboard. Listing 1 shows an example of a ground operator for `!pick_up_object peppermill1 leftarm1` with preconditions and effects. It can be executed if `peppermill1` is on counter `C1` and the robot is at an area near this counter (line 2). If the action is successful, `peppermill1` is removed from `C1` (line4) and ends up in the robot's arm (line 4).

```
1  !pick_up_object peppermill1 leftarm1
2  prec: robotAt(PMA C1), on(peppermill1, C1)
3  add: holding(peppermill1, leftarm1)
4  del: on(peppermill1, C1)
```

Listing 1: Ground instance of an operator for picking up a pepper mill with the left arm.

In our approach the planner itself does not need to be modified. Only the planning domain is extended to integrate functional imagination. We add a new operator `!imagine ?task ?arg1 ?arg2` to the domain; its arguments represent a task to be tested in simulation. It has no preconditions or effects. In addition, methods are required that make use of this operator. For our demonstration scenario, we modified an existing method for planning the task of moving an object by adding the task `!imagine !move_base_param ?area slow/fast` to the beginning of its decomposition. With the action `!move_base_param` the robot drives to a specified area with a given speed, e.g., `slow` or `fast`. For the `!imagine` operator, however, we can define multiple parameter values that should be simulated. In the given example `slow` and `fast` will be simulated by functional imagination.

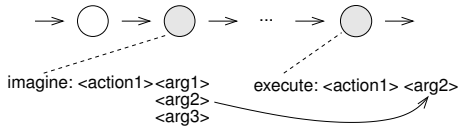

Fig. 4: Parameter instantiation in principle: during action execution (circles), the result of an imagine action may alter the parameter of another action consecutively.

E. Execution

The SEM generates the expected world state for each action `a` given as an argument of an `!imagine` operator (in our case `move_base`). To this end, it has to consider the current state of the Blackboard and the sequence of actions `seq` executed before `a`. We essentially collect all effects added $A = \cup_{a_i \in seq} add(a_i)$ and deleted $D = \cup_{a_i \in seq} del(a_i)$. Then $A \setminus D$ and $D \setminus A$ represent how the world state is expected to be changed by the executiion of `seq`. In our scenario, the robot moves ($robotAt(PMAEC1) \in A \setminus D$, $robotAt(NSA1) \in D \setminus A$) and the location of `peppermill1` changes as well ($on(peppermill1, tray) \in A \setminus D$, $on(peppermill1, EC1) \in D \setminus A$). Therefore, functional imagination is initiated with `robotAt(PMA EC1)` and `on(peppermill1, tray)`.

The altered world state, the action and its argument candidates are then sent to the Imagination Client. Meanwhile, all operators other than `!imagine` are physically executed as usual, including operators to be imagined for which the plan prescribes a set of default parameters. These defaults are used if the action has to be dispatched before the Imagination Client returns the outcome of the action.

The Imagination Client produces an ordered list of action parameterizations together with the expected durations of actions. If the most desirable action parameterization differs from the default specified in the current plan, the SEM modifies the plan by adapting the action parameters, as shown in Fig. 4. In addition, the SEM adds the expected duration of the actions to the temporal network. This improves the estimate of start and finish times of all actions planned after `move_base`.

IV. Experimental Results

All experiments were performed in simulation (Gazebo) or on the real robot (PR2). Fig. 5 shows the demo restaurant environment in the ROS visualization.

We will now present the results of validating simulated actions against reality, before giving the results of executing the proposed system on a PR2.

A. Simulation Validation

A physics-based simulation usually has no inherent noise, except for the minor noise introduced by the underlying hardware and by operating system scheduling. An obvious exception is explicitly modeled noise, such as Gaussian noise, as used in the Gazebo laser simulation. Action durations measured in the simulated restaurant environment are consistent with small variance. Table IV shows the mean

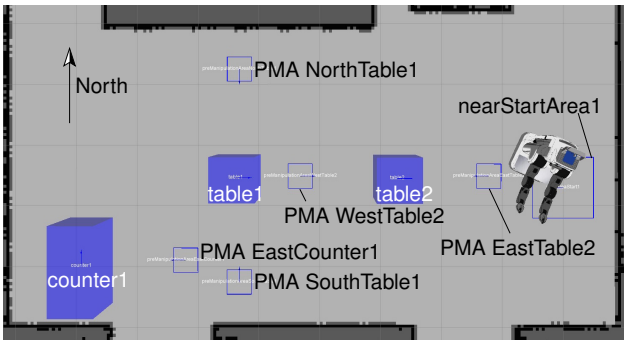

Fig. 5: Restaurant floor map with markers for the counter and the two tables (filled blue boxes) and for all pre-manipulation areas: two for each table (on opposite sides), one for the counter and one for the start area (near the right hand door). *North* is defined here upwards.

Action	Parameter	to/from	μ_{sim} $[s]$	μ_{pr2} $[s]$
`!MOVE_BASE_PARAM`	fast	EC1/ET2	14.04	16.40
`!MOVE_BASE_PARAM`	slow	EC1/ET2	20.22	22.59
`!MOVE_BASE_PARAM`	fast	EC1/NT1	15.99	16.85
`!MOVE_BASE_PARAM`	slow	EC1/NT1	18.38	19.41
`!MOVE_BASE_PARAM`	fast	ET2/EC1	24.08	23.07
`!MOVE_BASE_PARAM`	slow	ET2/EC1	27.00	29.86
`!MOVE_BASE_PARAM`	fast	NT1/EC1	15.33	16.59
`!MOVE_BASE_PARAM`	slow	NT1/EC1	19.06	20.89
`!MOVE_TORSO`		U/D	20.99	23.39
`!MOVE_TORSO`		D/U	21.69	22.19
`!TUCK_ARMS`		U/T	9.25	8.58
`!TUCK_ARMS`		T/U	5.70	5.29
Total mean			17.65	18.76

TABLE IV: Action duration mean μ in $[s]$, of 20 runs per action, executed in the simulation and on the real PR2.

duration of navigation, torso movement, and arm-tucking actions both in simulation (μ_{sim}) and on the PR2 (μ_{pr2}). In the table, μ is the mean value of an action duration, while 'to/from' lists the acronyms for the starting pose or posture of the robot. As discussed earlier, for `move_base` ET2 corresponds to EastTable2, EC1 is EastCounter1, etc. For `move_torso` actions, 'U' means up and 'D' means down. The `tuck_arms` action uses 'U' for untucked arms and 'T' for tucked arms.

The validation data was generated by running the robot capabilities (perception, actuation) and the robot control system. Generally the system load on each computer performing the simulation is close to its limit. Therefore, in addition to the following conclusions, system load will influence the measurements slightly. The data shows that μ is slightly smaller in simulation than on the real robot.

Differences in sensor data may result in μ being smaller in simulation. Simulated sensor data is almost perfect, whereas real sensor data contains significant noise. Odometry and laser sensors, which are used for localization and path planning, are noisy due to wheel slippage and reflection or absorption of the laser beams. Furthermore, the laser has some inherent noise. Friction and laser noise are modeled in the simulation (as a Gaussian distribution) but perhaps imperfectly. Arm and torso movements depend on real and

	θ_{fast}	θ_{slow}
Duration simulation [s]	32.01	45.27
Duration real [s]	76.24	109.85
Real-time factor	0.42	0.41
Duration scenario [s]	384.0	
Relative imagination time	0.08	0.12

TABLE V: Experimental results: time resources used by simulation during the toppling scenario.

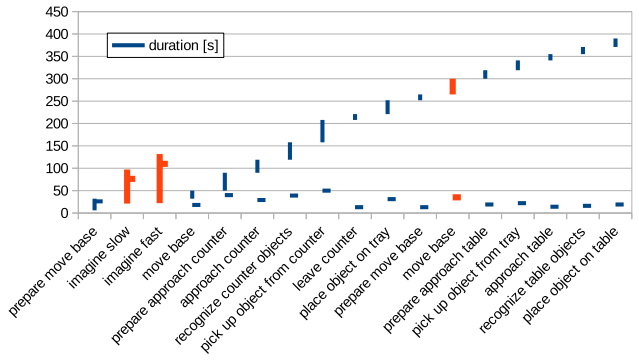

Fig. 6: Topple scenario: robot actions over time in seconds. The two imagination actions executed in parallel and the affected move base action are highlighted in red.

modeled joints, springs, and friction. Deviations could result from slightly inaccurate modeling parameters.

B. System Evaluation

In our experiment the robot was instructed to fetch a pepper mill from `counter1` and carry it safely, i.e., without toppling, to `table1` (cf. Fig. 5). Fig. 6 lists the plan steps produced by the planner. Note that, except for the imagination actions, all actions are executed in sequence. Thus, running imagination actions in parallel does not extend the overall duration here as the action to be fully instantiated is executed much later. The figure also shows that both imagination actions return before the robot picks up the object from the counter. Therefore, information gained from imagination can be used to schedule and reschedule any action after picking up the object. In this scenario, we had the following results from the imagination actions: $c = 0.87$ ('fast', 12 shaking events) and $c = 1.00$ ('slow', no shaking) supporting a 'slow' motion, preventing possible object toppling. Table V lists the (temporal) resources used by imagination in this scenario compared with the total scenario duration.

The same scenario evaluated with a standard system, i.e., without (functional) imagination, instantiates the move-base action with 'fast' motion settings and causes the pepper mill to topple and fail in 1 out of 4 experiments.

V. Conclusion and Future Work

We have presented a flexible integrated system combining high-level symbolic and concrete physics-based reasoning running in parallel on a PR2 robot and in the Gazebo simulation. The scenario presented shows how physics-based common sense knowledge can improve robust plan execution via action parametrization. Because of the common simulation character, functional imagination reasoning can be extended to temporal information about actions as well as to results from other actions. The computational cost of simulations limits the sample size and duration of actions in principle. Nevertheless, our practical scenario shows that interleaving simulation with execution does not have to come with extra costs.

In future work we will investigate dynamic plan adaptations during execution based on imagination results. We also plan to enable parallel simulations independent of world state.

References

[1] H. G. Marques, O. Holland, and R. Newcombe, "A modelling framework for functional imagination." in *"Proc. AISB Convention*, 2008, pp. 51–58.

[2] S. Rockel, D. Klimentjew, L. Zhang, and J. Zhang, "An hyperreality imagination based reasoning and evaluation system (hires)," in *Proc. ICRA*, 2014.

[3] L. Hinkle and E. Olson, "Predicting object functionality using physical simulations," in *Proc. IROS*, November 2013.

[4] S. Zickler and M. Veloso, "Variable level-of-detail motion planning in environments with poorly predictable bodies," in *Proc. ECAI*, 2010.

[5] N. Akhtar, A. Küstenmacher, P. Plöger, and G. Lakemeyer, "Simulation-based approach for avoiding external faults," in *Proc. ICAR*, 2013.

[6] M. Dogar, K. Hsiao, M. Ciocarlie, and S. Srinivasa, "Physics-based grasp planning through clutter," in *Proc. RSS*, 2012.

[7] A. Boeing and T. Bräunl, "Evaluation of real-time physics simulation systems," in *Proc. of the 5th International Conference on Computer Graphics and Interactive Techniques in Australia and Southeast Asia*, ser. GRAPHITE '07. USA: ACM, 2007, pp. 281–288.

[8] J. Victores, S. Morante, A. Jardon, and C. Balaguer, "Towards robot imagination through object feature inference," in *Proc. IROS*, 2013, pp. 5694–5699.

[9] H. G. Marques, R. Knight, R. Newcombe, and O. Holland, "An anthropomimetic robot with imagination: one step closer to machine consciousness?" in *Nokia Workshop on Machine Consciousness*, 2008.

[10] D. Gamez, R. Newcombe, O. Holland, and R. Knight, "Two simulation tools for biologically inspired virtual robotics," *Proc. of the IEEE 5th chapter conference on advances in cybernetic systems, Sheffield*, pp. 85–90, 2006.

[11] K. Erol, J. Hendler, and D. Nau, "HTN Planning: Complexity and expressivity," in *Proc. AAAI*. AAAI Press, 1994, pp. 1123–1128.

[12] L. de Silva, A. K. Pandey, and R. Alami, "An interface for interleaved symbolic-geometric planning and backtracking." in *Proc. IROS*, 2013.

[13] M. Mansouri and F. Pecora, "More knowledge on the table: Planning with space, time and resources for robots," in *Proc. ICRA*, 2014.

[14] L. Mösenlechner and M. Beetz, "Using physics- and sensor-based simulation for high-fidelity temporal projection of realistic robot behavior," in *Proc. ICAPS*, 2009.

[15] L. Kunze, M. E. Dolha, and M. Beetz, "Logic programming with simulation-based temporal projection for everyday robot object manipulation," in *Proc. IROS*, 2011.

[16] S. Rockel et al., "An ontology-based multi-level robot architecture for learning from experiences," in *Proc. of Designing Intelligent Robots: Reintegrating AI II, AAAI Spring Symposium*, 2013.

[17] Š. Konečný, S. Stock, F. Pecora, and A. Saffiotti, "Planning domain + execution semantics: a way towards robust execution?" in *Proc. of Qualitative Representations for Robots, AAAI Spring Symposium*, 2014.

[18] D. Fox, W. Burgard, and S. Thrun, "The dynamic window approach to collision avoidance," *Robotics Automation Magazine, IEEE*, vol. 4, no. 1, pp. 23–33, Mar 1997.

[19] D. S. Nau, T.-C. Au, O. Ilghami, U. Kuter, J. W. Murdock, D. Wu, and F. Yaman, "SHOP2: An HTN planning system," *J. Artificial Intell. Research*, vol. 20, pp. 379–404, 2003.

[20] S. Stock, M. Günther, and J. Hertzberg, "Generating and executing hierarchical mobile manipulation plans," in *Proc. of ISR/Robotik*, 2014.