DOI: 10.1109/IROS.2018.8594458
URL: https://ieeexplore.ieee.org/document/8594458

Please cite as:
A. S. Bauer, P. Schmaus, A. Albu-Schäffer and D. Leidner, "Inferring Semantic State Transitions During Telerobotic Manipulation," *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Madrid, 2018, pp. 1-9.

# Inferring Semantic State Transitions During Telerobotic Manipulation

Adrian S. Bauer[1], Peter Schmaus[1], Alin Albu-Schäffer[1,2], and Daniel Leidner[1]

***Abstract*— Human teleoperation of robots and autonomous operations go hand in hand in today's service robots. While robot teleoperation is typically performed on low to medium levels of abstraction, automated planning has to take place on a higher abstraction level, i.e. by means of semantic reasoning. Accordingly, an abstract state of the world has to be maintained in order to enable an operator to switch seamlessly between both operational modes. We propose a novel approach that combines simulation based geometric tracking and semantic state inference by means of so called State Inference Entities to overcome this issue. We also demonstrate how Evolutionary Strategies can be employed to refine simulation parameters. All experiments are demonstrated in real-world experiments conducted with the humanoid robot Rollin' Justin.**

## I. Introduction

Autonomous robots are turning into useful tools that can be deployed in areas that are hazardous for humans. One major application domain is thereby the space exploration sector. Even though space assistant robots such as the humanoid robot Rollin' Justin [1] are mechanically capable of manipulating their environment, they are currently unable to work fully autonomously. Accordingly, they are typically remotely operated by humans. Our research on telerobotic manipulation with space assistant robots is therefore mainly concerned with two aspects. First, direct teleoperation by means of haptic input devices, as for example conducted in the Kontur-2 experiment [2]. And second, human-robot co-operation by means of supervised autonomy, as it is performed in the METERON SUPVIS Justin experiment [3]. During those two missions, we have learned that both control modalities are necessary to operate a robot most efficiently under varying conditions (see Fig. 1).

*Traded control* is an approach that allows the operator to switch between autonomous task execution and teleoperation [4]. An open research question in traded control is the synchronization of world states while switching from teleoperation to autonomous mode. That is, planning in autonomous mode requires an accurately modeled semantic world state that corresponds to the real world environment. However, during teleoperation, robots are not yet able to keep track of the semantic state changes that are initiated by the operator. Thus, the semantic world state at the end of a teleoperation session is unknown, making it impossible for the robot to operate autonomously afterwards.

An intuitive analogy for this problem can be derived by comparing the robot with a sleepwalker. While sleepwalking at night, the motor capabilities of the sleepwalker are intact while at the same time he/she does not perceive the environment on a conscious level [5], analogous to the robot whose motors are active without "perceiving" the changes it exerts on the environment on a semantic level. After waking up, sleepwalkers tend to be disoriented and they are usually unable to remember their actions. Similarly, robots are "disoriented" after they are teleoperated due to the lack of status updates during teleoperation mode. While visual servoing may be used to track geometric state changes during teleoperation [6], it is not robust against occlusion, poor visibility, and constraint to objects in the field of view. There is currently no work known to the authors that deals with semantic state validation for telerobotic manipulation. Accordingly, this paper proposes an approach to derive semantic state transitions from robot telemetry retrieved during teleoperation by means of physics simulations and *State Inference Entities (SIEs)*.

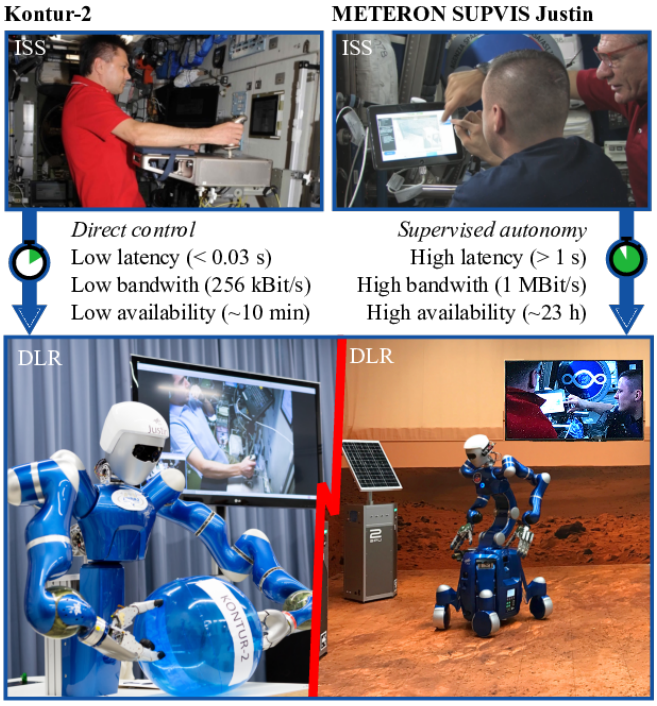


Fig. 1: The humanoid robots Space Justin and Rollin' Justin remotely operated from the International Space Station using direct control (Kontur-2, left) and supervised autonomy (METERON SUPVIS Justin, right). Given the varying communication conditions both control modalities are necessary, yet direct control prevents the use of supervised autonomy due to the loss of semantic information. This gap is visualized as red line in the bottom center.

[1]Institute of Robotics and Mechatronics, German Aerospace Center (DLR), Wessling, Germany, Contact: adrian.bauer@dlr.de

[2]Technische Universität München (TUM), Sensor-Based Robotic Systems and Intelligent Assistance Systems, Munich, Germany

The contributions of this paper are (i) a software architecture to infer semantic state transitions during robotic teleoperation, (ii) the concept of SIEs that allow to extract semantic knowledge from physics simulations, and (iii) a method to estimate simulation parameters by means of an evolutionary algorithm. The developed methods are validated based on real world robot telemetry recorded during teleoperation of the humanoid robot Rollin' Justin.

The remainder of this paper is structured as follows. Section II gives an overview over related work on the topic, especially traded control, semantic planning algorithms for robots and naive physics. Section III proposes a system concept to tackle the issue of state validation in teleoperation scenarios with traded control. Section IV presents the SIE used in the framework for state inference. Section V is dedicated to explaining how the parameters for the simulation were estimated, aiming at minimizing the reality gap. In Section VI the system and the parameter estimation method are evaluated and discussed before a conclusion is drawn and an outlook is provided in Section VII.

A preliminary version of this work has been presented in [7], where this paper extends on the parameter estimation and evaluation of the framework.

## II. Related Work

While robots are not yet able to operate fully autonomously, they already possess numerous autonomous features. Therefore, different approaches emerged that aim at blending human and robot intelligence in order to create powerful human-robot teams. Their recurrent idea is to ease task complexity for the human operator by integrating autonomous support-features on the robot side, allowing the user to command robots on different levels of autonomy. They come in multiple slightly different implementations (see [8, Tab. 5.2]).

*Supervised autonomy* is one implementation that allows the operator to initiate tasks on a high level of abstraction while the execution is carried out by the robot autonomously [8]. It allows the operator to command the robot without demanding full attention, thus, freeing resources of the operator that can be invested on other tasks. In *shared control* on the other hand, the robot is controlled via continuous input on a *Human-Robot Interface (HRI)* [8], augmenting the human input e.g. with safeguarding functions such as autonomous collision avoidance. Robot systems can also provide both, supervised autonomy and shared control as in [9]. This combination is called *traded control* and enables the operator to command the robot in supervised autonomy and switch to shared control whenever the robot encounters an unsolvable situation.

Achieving predefined goals in the supervised autonomy mode requires the robot to be able to generate and execute plans. Since goals are rather defined as symbolic states (e.g. "place the cup on the tray") than as geometric states (e.g. "cup at position $(x, y, z)$ with pose $(\alpha, \beta, \gamma)$") and since planning is much easier on the symbolic domain, planning is mostly performed on a symbolic level, even though it might be refined on the geometrical domain (e.g. *hybrid planning*). Hybrid planning algorithms have been described, for example, in [10], [11], [12].

On the robotic platform *Rollin' Justin* [1] hybrid planning is enabled by the use of *action templates* [13], [14]. Action templates are action representations that are separated into a symbolic header, providing the symbolic information about the action in the *planning domain definition language* (PDDL), and the body, grounding the action geometrically to the robot. Key to semantic planning is the description of the state transition in the symbolic header.

[15] learn the semantics of a task from raw robot sensor data. The drawback of this approach is the large amount of training data needed and the restrictive subgoal property.

Reasoning in intelligent systems still poses an unsolved problem. While humans employ common sense to deduce unknown parameters of a problem, robots are not yet able to do so and demand the specification of every detail of the problem. Considering the goal of predicting the symbolic world state after teleoperation (essentially a mapping from the initial world state and a time series of joint angles to the new world state), many parameters are included that are not specified explicitly, such as the physical rules by which state transitions take place. It is supposed that humans possess an inherent ability to predict and assess physical phenomena that is referred to as *naive physics* [16]. Naive physics gained impact on AI research with the work of Hayes [17], [18] who proposed to create a formalization of everyday physics knowledge that could be used by AIs trying to solve problems that include physics.

The idea of naive physics has recently been refreshed by [19] who propose the existence of an intuitive physics engine in the human mind. They also compare the simulation capabilities in the human brain with physical simulation engines as used for robot development or computer games.

It is thus not surprising that researchers tried to exploit the rich intrinsic physical knowledge of physic simulation engines in order to solve robotics tasks. Johnston and Williams [20] created a simulation environment to solve, for example, the *egg cracking problem*[1]. Mösenlechner and Beetz employed the simulation framework Gazebo[2] in order to evaluate task execution by simulation [21], to check possible put-down locations in terms of stability and visibility [22], and to find robot poses that allow object manipulation most easily [23]. The same approach has been used in [24] to solve the egg cracking problem, and in [25] to infer task parametrization of vaguely specified instructions.

## III. System Concept

The proposed framework to infer semantic state transitions is designed to be as general as possible. However, it was implemented in connection with the existing system on Rollin' Justin that has been described by Leidner et al. in [13], [14]. In this section we will first present the existing

[1] `http://commonsensereasoning.org/problem_page.html#eggcracking`, last retrieved on July 24, 2018

[2] `http://gazebosim.org`, last retrieved on July 24, 2018

system, then elaborate on the proposed framework in general, and finally present the modules of which it consists.

### A. Existing Planning System

The existing planning system is centered around the world representation module that provides the geometric and symbolic state of the world as well as an interface to query object-specific information about every object in the world.

Being able to employ the hybrid planning abilities requires permanent maintenance of the symbolic world state. The semantic planner, that is invoked first whenever planning is executed, searches for a series of actions, that enable a transition from the current semantic state of the world to the desired state. Once a symbolic plan is found, it is checked for feasibility with a geometric planner that tries different solutions and requests a new plan if all solutions fail. A deviation in the symbolic world state might thus result in an incorrect series of actions that cannot be executed by the physical robot.

Updating the symbolic world state during autonomous operation is straightforward once the initial world state is known. Whenever an action is executed on an object, the corresponding action template is queried from the *Object DataBase (ODB)*. Since the effects of actions on the symbolic world state are stated explicitly in the headers of the action templates (at least as long as no errors occur during execution), the new world state can be derived by applying the effects to the current world state. This process has been described by Leidner et al. in [13].

However, while being teleoperated, the robot is commanded on a low level and does not possess a symbolic representation of the action the user executes. Thus, the symbolic state of the world cannot be updated automatically but instead needs to be evaluated from whatever information the robot is able to acquire. Even a robot that was able to update its geometric world representation during teleoperation, e.g. by means of computer vision, would require a mapping of the geometric state to a symbolic state if it was to perform autonomous symbolic planning afterwards.

### B. Inference Framework

Generally the framework follows a circular pattern. It is, in the first step, designed to be active during teleoperation of the robot, thus, modules that cope with planning need not be considered in this context. An overview is provided in Fig. 2 showing the interaction between world representation, simulation engine, inference module, and the ODB. Since the framework forms a closed loop, the process described here is able to reach the configuration that it started from. In concrete terms this means that when starting from a known world state, by going through the process, the system finishes in a state where at least an estimate of the world state is known again. This corresponds to the situation of switching from autonomous control to teleoperation and back again as the framework infers the world state.

Starting from the described situation of a known world state, the world representation is able to initiate the simulation environment to reflect the current world state. Once

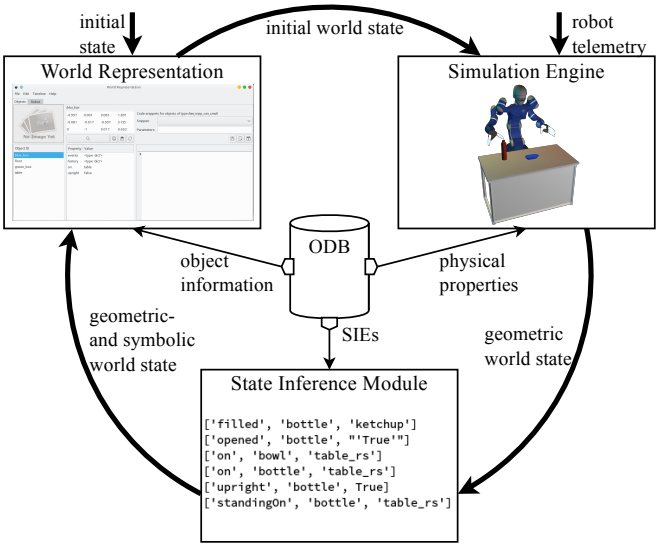


Fig. 2: Overview of the framework. The world representation holds the current state of the world. The simulation engine computes the new geometric world state from the initial world state and robot telemetry data. The state inference module computes the symbolic world state from the result of the simulation. Information about objects is provided by the central ODB.

the simulation has been initialized, it receives continuous input from the robot about its current telemetry and mirrors it to the simulated robot. Mimicking robot movements, the simulation computes interactions between the robot and the environment. Thus, the simulation maps the initial state of the world and a time series of robot telemetry to a resulting geometrical world state.

After the teleoperation finishes, the geometric world state needs to be transformed into a symbolic one. Therefore, the simulated geometric state of the world in combination with descriptions of collisions of objects is parsed to the inference module and inference is invoked. The results of the inference process are forwarded to the world representation where they are used to update the world state. This ensures that the world representation is updated accordingly whenever the robot switches back from teleoperation mode to autonomous mode. The knowledge necessary for inferring symbolic states from geometrical information is attached to the objects themselves alongside the predicates they provide.

Each of the aforementioned steps, relies on object-specific information. All object-specific information is stored in the ODB and queried from the respective modules, as can be seen in Fig. 2.

An advantage of the system that arises due to its loop-structure is that we can not only use it to perform the symbolic inference directly after the manipulation finished. In addition to that, the system can also be employed to generate on-line estimates of the world state during teleoperation. As described, the whole process can be viewed as a mapping from an initial world state and a time series of robot telemetry to a final world state. Such a mapping can be split up by

introducing an intermediate world state that occurs at time $t$. The time series of robot sensor values can be split up into the values that occurred before $t$, and those afterwards. The intermediate world state can be inferred from the initial world state and the sensor values up to time $t$ and the final world state can be inferred from the intermediate world state and the second part of the sensor values. Executing this process recursively, any temporal resolution can be achieved, limited only by the frequency of the sensor measurements and the real-time capability of the resulting system. Our experiences with the framework indicate that the simulation engine forms the main bottleneck for the overall speed.

### C. Simulation Environment

The physical simulation is based on the robot simulation environment *Gazebo* [26] in conjunction with the *Open Dynamics Engine* (ODE)[3]. The framework was augmented by an interface that allows to change objects in the simulation, control robot movements, and return simulation results. At the beginning of a teleoperation session, the simulation is initialized with the current state of the world that is queried from the world representation. Information about the objects present in the environment is requested from the ODB. This includes meshes as well as physical information like mass, center of mass, and inertia tensors.

Once the simulation is initialized, the robot in the simulation starts to mirror the movements of the real world robot. Accurate mirroring is achieved by streaming telemetry data of the robot (here: the joint angles of the robot) constantly to the simulation where the joint angles of the simulated robot are set accordingly.

As soon as the teleoperation finishes, results of the simulation, including object poses, a list of manipulated objects, and a list of object collisions, are collected and passed to the interference module.

Interfaces on the world- and robot level have been realized by creating plugins for Gazebo. As the overall system has been designed with focus on modularity, the simulation module can easily be replaced by any other simulation engine, only requiring a new adapter to allow for communication with the other modules.

### D. Inference Module

After receiving the data from the simulation module, the inference process starts with the goal to extract semantic predicates from the simulated geometric state of the world. So far we are interested in static attributes of the world, similar to the ones in [27], that are evaluated based on the poses of objects and forces acting between them. Nevertheless, the framework allows for expanding the input for the inference module later on. The predicates (unary or n-ary) that are to be evaluated are bound to the objects in the ODB and so is the information used for evaluation of predicates. Thus, we decided to implement the inference knowledge in terms of State Inference Entities that are stored alongside other object information in the ODB.

[3] `http://ode.org/`, last retrieved on July 24, 2018

The workflow for evaluating predicates is sketched in the following: in the first step, the inference module selects the objects on which the predicates are to be evaluated. This set of objects consists of (i) all objects that have moved during teleoperation and (ii) all objects that are in collision with an object from (i). Constraining the set of objects in this way reduces the computational load of the state inference process and represents the intuition of updating only the state of those objects that have been manipulated.

In the second step, the possible predicates and the corresponding SIEs for these objects are queried from the ODB. The SIEs are evaluated by the state inference module and return nothing (returning nothing means that the predicate is not fulfilled) or the parameters of the predicate (meaning that the predicate holds in a certain parametrization). The results are collected and used to update the world state.

## IV. State Inference Entities

The SIEs form a central aspect of the state inference process. They consist of executable Python code and implement a common interface method `executeSnippet` that is called from the state inference module. On calling this method, all information available to the State Inference Module is passed as argument to the function and used to infer the state of the inspected predicate. The function itself can be very simple, as seen in the pseudocode for the predicate `upright` in Algorithm 1 and used in this work, but it could potentially be any type of classifier. As the design of the SIEs allows to use the full spectrum of python code, there is virtually no limitation on the type of classifier. In the future it is planned to augment the whole system such that even probabilistic classifiers can be used.

The `executeSnippet` method returns either nothing or a list of length three where the first element represents the object name for which the predicate has been evaluated, the second element is the name of the predicate, and the third element is the value of the predicate. For example `['bottle', 'on', 'table']` means that the predicate `on` of the object `bottle` will be filled with the value `table`. In case the predicate is not fulfilled, the SIE returns an empty list.

Generally the SIEs are defined for each object and predicate, however, objects that inherit from other objects also inherit the SIEs. Inherited SIEs can be overwritten by creating a SIE of the same name in the inherited object.

**Algorithm 1** Example implementation of the method `executeSnippet` for predicate `upright`

```
function EXECUTESNIPPET(self)
    zAxis ← self.zAxis
    threshold ← 20
    angle ← angleBetween(zAxis, [0, 0, 1])
    result ← angle < threshold
    return ["upright", self.name, result]
end function
```

## V. Estimating Parameters

In order to test the implementation of the framework, an initial parameter set for the simulation had to be found. Two types of approaches had been chosen to estimate the parameters: computing physics parameters based on the meshes of objects and optimizing parameters via an *Evolutionary Strategy (ES)* [28].

If the geometry of an object and either the total mass or the density are known, physical parameters like mass, center of mass, and the inertia tensor can be computed assuming a uniform density over the object. We implemented an algorithm that generates prior estimates of these parameters for the objects in the ODB.

While some physical parameters can be derived from the geometry of an object, others such as e.g. friction parameters are not grounded in geometry. Uncertainties in these parameters result in the deviation of simulated behavior from the behavior observed in real world. This phenomenon is known as *reality gap*. Similar to Laue and Hebbel [29] we employed an *evolutionary strategy* (ES) to find a set of parameters that minimizes the reality gap. We focused on optimizing the friction parameters of two objects ($\mu_1, \mu_2$, see Section VI) and the simulation parameters *Constraint Force Mixture (CFM)* and *Error Reduction Parameter (ERP)*, ODE-internal parameters, critical for stability of the simulation.

Evolutionary strategies allow to optimize sets of real-valued parameters, inspired by evolution in nature. First, generations of individuals are generated, assessed and selected according to a fitness function. Then a new generation is created via mutation and recombination from the selected individuals. This type of optimization algorithms is found mostly in problems where no derivative of the error function with regards to the parameters can be computed [28].

Our optimization applied the $(\mu/\rho_D + \lambda)$ version of the ES. Given the dimensionality of the parameter vector $D = 4$, we selected $\lambda = 10 \cdot D = 40$ [30], $\mu = 10$ (empirically), and $\rho = 2$ (biologically inspired). Furthermore the mutation parameter $\sigma$ was tuned by $\sigma$-self adaptation [31] with $\tau = D^{-0.5} = 0.5$ [32].

Due to randomness in the simulation, each scene was simulated $K = 3$ times, yielding a predicted position $\hat{\mathbf{t}}_{i,k}$ and orientation $\hat{\mathbf{R}}_{i,k}$ for each object $i \in 1 \dots N$. We defined the translational error $e_t$ as the *Root-Mean-Square Error (RMSE)* of the Euclidean distance between predicted and measured object positions as

$$e_{t,i,k} = \sqrt{\left(\mathbf{t}_{i,k} - \hat{\mathbf{t}}_{i,k}\right)^T \left(\mathbf{t}_{i,k} - \hat{\mathbf{t}}_{i,k}\right)} \tag{1}$$

and the rotational error as the RMSE of the geodesic distance between the predicted and measured rotation on the unit sphere as [33]

$$e_{R,i,k} = \frac{1}{\sqrt{2}} \left\| \log\left(\mathbf{R}_{i,k} \hat{\mathbf{R}}_{i,k}^T\right) \right\|_F \tag{2}$$

with the log operator mapping the rotation matrices from SO3 to the Lie algebra so3 [34]. These errors are combined to the total error $e_{i,k}$ of object $i$ in run $k$ as

$$e_{i,k} = \sqrt{e_{t,i,k}^2 + c \cdot e_{R,i,k}^2}. \tag{3}$$

The parameter $c = 0.01$ (empirically) balances the rotational and translational error. The overall error $e$ of an individual over all objects and all scenes is computed as

$$e = \sqrt{\frac{1}{NK} \sum_{i=0}^{N} \sum_{k=0}^{K} e_{i,k}^2}. \tag{4}$$

## VI. Evaluation

The evaluation was performed in a twofold manner. In a first step the precision of the simulation and the parameters resulting from the ES were validated since a precise simulation forms the basis for the whole framework to be applicable. In a second step the inference capabilities of the framework were evaluated based on a small set of objects and predicates from the ODB. The accompanying video shows the evaluation procedure.

### A. System

The framework was run on a Linux-Desktop PC, running on a quad-core Intel® Xeon® CPU E5-1630 with 16GB memory. As simulation environment we used Gazebo 7.7.0 with ODE 0.15.1. Communication between the modules was enabled by the in-house developed middleware Links-and-Nodes. For the evaluation of the inference capabilities, the robot was commanded manually via a 3DCONNEXION SpaceNavigator®.

With this setup the simulation was running with a real time factor around 2.2, allowing us to run the simulation on-line while the robot was teleoperated.

### B. Experiment

In order to assess the precision of the simulation, an environment consisting of a table and two different sized boxes was set up. The experiment setup is shown in Fig. 3. The endeffector of the robot was moved manually in zero gravity mode such that it interacted with the objects and telemetry data was recorded. Ground truth was generated by using the localization capability of Rollin' Justin, based on

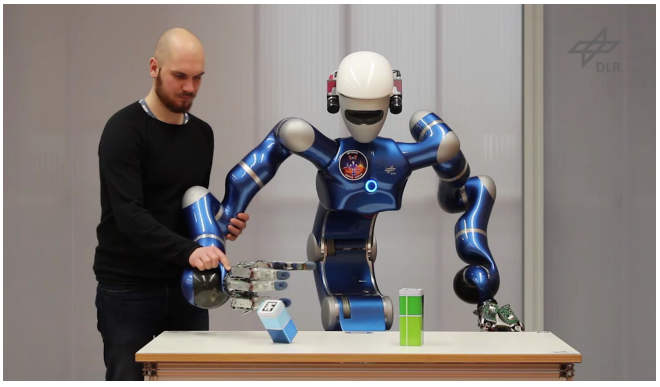

Fig. 3: Overview of the experiment setup.

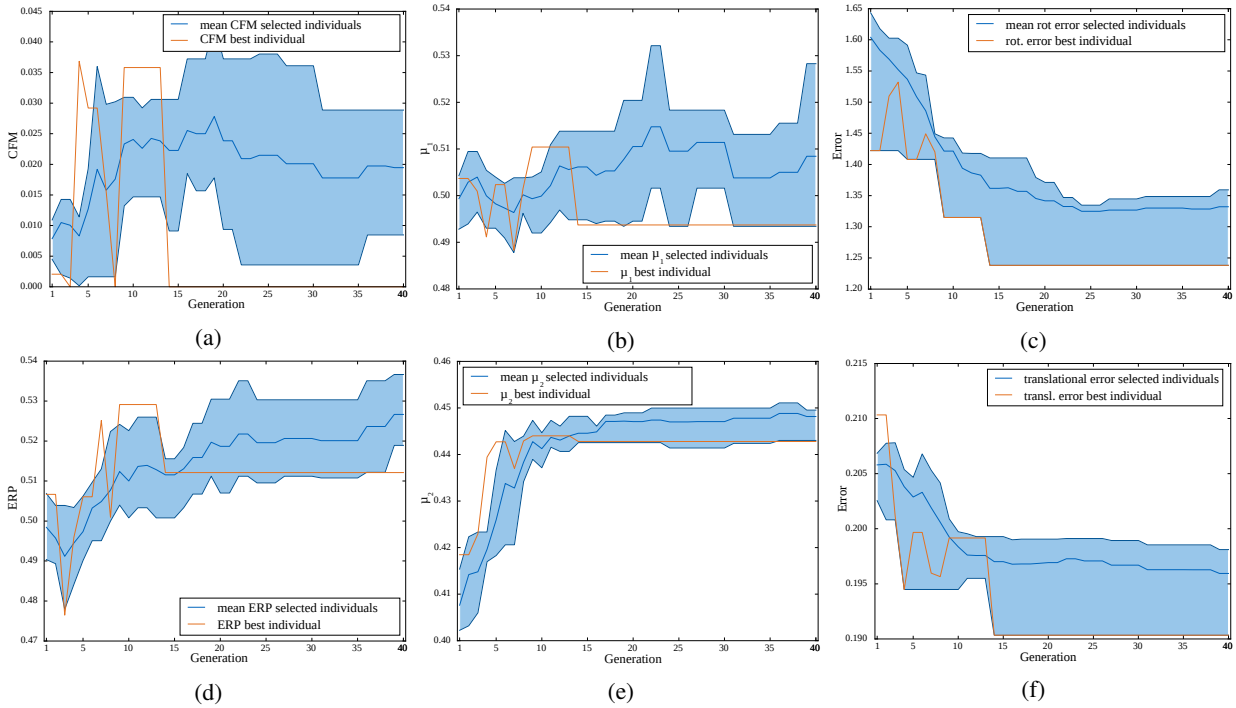


Fig. 4: (a), (b), (d), and (e) show the evolution of the simulation parameters. The orange line marks the parameter value of the best individual, the dark blue line the mean parameter value of all individuals selected as parents for the next generation and the light blue shaded area marks the 20th - 80th percentile of the parameter values in the selected individuals. (c) depicts the evolution of the rotational error over time and (f) the evolution of the translational error. The orange line marks the best *overall* individual, the dark blue line marks the mean and the blue shaded area the best individual to 80th percentile.

TABLE I: Results of the ES algorithm, comparing best individuals from initial generation and 30th generation.

| | CFM | ERP | $\mu_1$ | $\mu_2$ | error rotational $e_R$ | error translational $e_t$ | overall error $e$ |
|---|---|---|---|---|---|---|---|
| initial generation | 0.0021 | 0.5067 | 0.5037 | 0.4184 | 1.4223 | 0.2103 | 0.2539 |
| 30th generation | 0.0 | 0.5121 | 0.4937 | 0.4427 | 1.2381 | 0.1903 | 0.2271 |

AprilTag 2 [35], before and after each trial. Four trials were recorded: (i) the robot pushing over one box, (ii) the robot pushing one box sliding over the table, (iii) the robot pushing one box against another one, and (iv) the robot grasping a box and placing it on the table again. From this set of trials trial (iv) had to be excluded since grasping posed a problem in stability of the simulation at the current status.

The results of the optimization are shown graphically in Fig. 4 for selected parameters and numerically in Table I. The table shows the best individual from the first generation and after 30 iterations including the parameter set and the errors. Fig. 4 shows the evolution of the rotational and translational error as well as selected parameters of those individuals that "survived" each generation. The plots show that the mean error of the parents decreased quite steadily while the minimum error was already reached in generation 14. The ERP and $\mu_2$ show a rather clear trend in their evolution, whilst $\mu_1$ stays more or less constant with more variation.

The inference capabilities of the framework were tested by implementing two predicates, namely `on` and `upright` for the two `box` objects. Fig. 5 shows the simulation of the recorded telemetry of pushing over the green box and the resulting change in the world representation, both for a simulation with good and bad parametrization. In both cases the transition from the simulated geometrical world state to the extracted symbolic world state was performed correctly.

### C. Discussion

The proposed framework enables us to keep track of the symbolic world state during teleoperation. We decided to split up geometric and symbolic reasoning into two modules, allowing us to employ a physics simulation and its inherent physical common sense knowledge to infer state changes on the geometric level. Using a physic simulation engine brings the advantage of being able to use highly advanced physical knowledge without the need to re-implement it. If, in the near future, physical simulations will make a big leap forward, the modular design of the framework allows for integrating a new simulation engine with minimum effort.

The general framework has proven to be able to extract the predicates `on` and `upright` successfully, based solely on the initial world state and a recorded robot telemetry, same as would also be available in teleoperation. This marks a remarkable advance from not being able to update the symbolic world state during teleoperation at all.

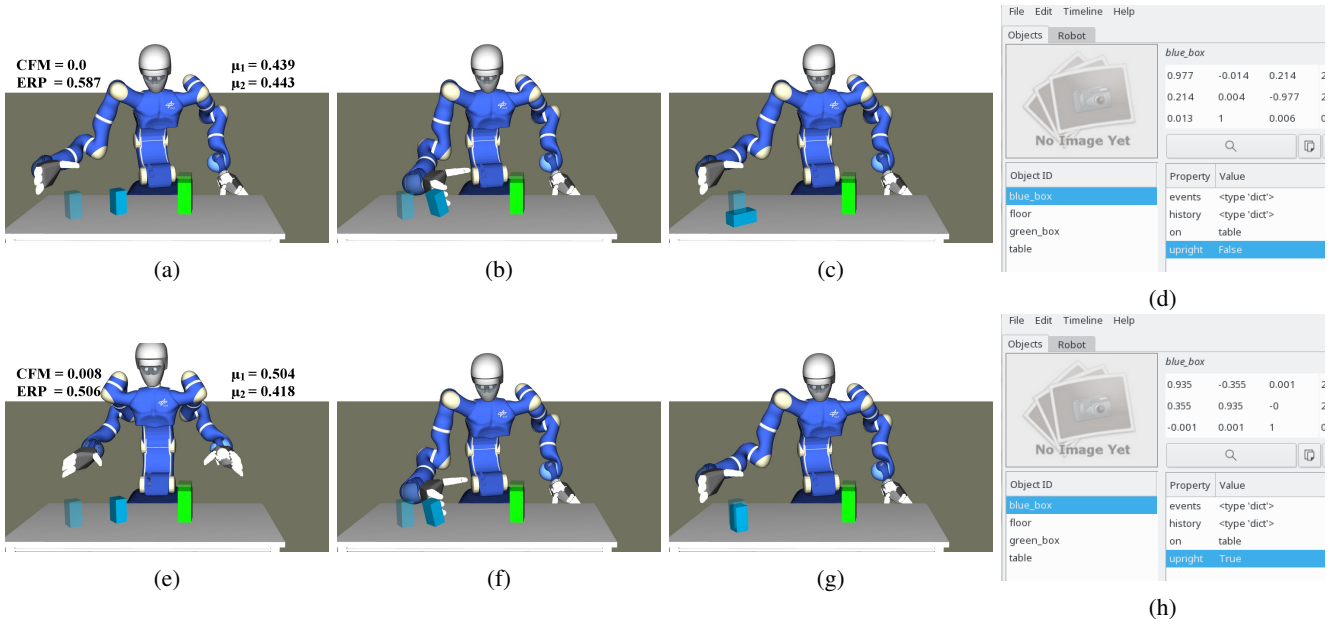


Fig. 5: Results from two runs with different parametrization. First line shows results from a run in bad parametrization: (a) before the recorded telemetry is replayed, (b) during replay Rollin' Justin is in contact with the blue box, (c) after the replay has finished, and (d) the extracted symbolic world state. Second line shows results from a run in better parametrization: (e) before replay, (f) during replay, (g) after replay, and (h) the extracted symbolic world state. Generally the transparent blue box marks the ground truth position of the blue box after the experiment.

The framework also showed to be able to learn from observation through the use of evolutionary strategies. While the prediction error decreased slowly over the optimization process, some simulation- and object properties were adapted to better fit the observations. Table I shows a decrease in rotational and translational error by each $\approx 10\%$, while Fig. 4 reveals that some parameters as the ERP and $\mu_2$ evolve with a stable trend. However, with a high number of objects involved in a scene and a high number of parameters that can be tuned for each object, the size of each generation in the ES grows rapidly. If the number of individuals per generation is too small, the ES might get stuck in some early local minimum. On the opposite, having a high number of individuals per generation comes with increased computational cost. Thus, the evolutionary strategy can right now only be used for small scenes with a few objects.

The results from Fig. 5 give an impression of how the simulation and the inference module interact and why the evolutionary strategy is important for estimating good parameters. If the estimated set of parameters is bad, the simulation predicts an incorrect world state and, thus, the inference module infers an incorrect world state as in Fig. 5d compared to Fig. 5h.

The concept of inferring the symbolic world state is based on State Inference Entities. Being implemented as mere Python code, SIEs offer a flexible interface for the inference process since the whole range of possibilities offered by the high-level programming language can be used. Furthermore, SIEs are stored attached to objects, resulting in user-friendly modularity and maintainability.

## VII. Conclusion and Outlook

With the proposed framework we come a step closer to awake the sleepwalking robot and enable it to keep track of the changes it induces in its environment, thus, providing the ground for smooth transmissions between teleoperation and (supervised) autonomous behavior. The framework can be used for teleoperation scenarios where the objects in the scene are well known and where the physics of the overall scene are rather stable.

The proposed framework does not relay on constantly monitoring object positions with cameras but instead aims at predicting their behavior based on a physical simulation. This approach is motivated by the facts that especially visual sensors are constrained on a field of view and that sensors are likely to fail because of different reasons while our approach keeps a model even of objects that are not visible and is robust to sensor failures. It could thus complement vision based scene understanding. On the other hand, using a physics engine comes with the disadvantage of requiring physically accurate models of the objects.

Following this work we see many opportunities and open questions that can be investigated further. Namely they are:

The input data for the simulation so far consists of the joint angles of the robot. A robot typically has a multitude of sensors, thus, our goal is to use more input modalities to refine the simulation. Especially force measurements and visual input could be integrated into the propriosimulation framework.

In the inference module, predicates and states in the world are assumed absolute or discrete. As all inference processes

are subject to noise and uncertainty, the system could be extended to represent this uncertainty by returning probabilities for states. The simulation could be invoked multiple times with initial conditions sampled from the probabilistic distributions of each parameter, resulting in a distribution over possible states after teleoperation.

Furthermore, the system could be used in the autonomous operation mode to support evaluation of tasks execution. Similar as in the teleoperation mode, the simulation could be fed with the recorded data of the robot in real time in order to infer the outcome of an action.

## ACKNOWLEDGMENT

This work was partially supported by the Bavarian Ministry of Economic Affairs and Media, Energy and Technology, and the project SMiLE.

## REFERENCES

[1] C. Borst, T. Wimbock, F. Schmidt, M. Fuchs, B. Brunner, F. Zacharias, P. R. Giordano, R. Konietschke, W. Sepp, S. Fuchs, C. Rink, A. Albu-Schaffer, and G. Hirzinger, "Rollin'justin-mobile platform with variable base," in *Proc. 2009 IEEE Int. Conf. Robotics and Automation (ICRA)*, Kobe, Japan, May 2009, pp. 1597–1598. 1, 2

[2] J. Artigas, R. Balachandran, C. Riecke, M. Stelzer, B. Weber, J. H. Ryu, and A. Albu-Schaeffer, "Kontur-2: Force-feedback teleoperation from the international space station," in *Proc. 2016 IEEE Int. Conf. Robotics and Automation (ICRA)*, Stockholm, Sweden, May 2016, pp. 1166–1173. 1

[3] N. Y. Lii, D. Leidner, P. Birkenkampf, B. Pleintinger, R. Bayer, and T. Krueger, "Toward scalable intuitive telecommand of robots for space deployment with meteron supvis justin," in *Proc. 14th Symp. Advanced Space Technologies in Robotics and Automation (ASTRA)*. Leiden, The Netherlands: European Space Agency (ESA), Jun 2017. 1

[4] D. Kortenkamp, R. P. Bonasso, D. Ryan, and D. Schreckenghost, "Traded control with autonomous robots as mixed initiative interaction," in *AAAI Spring Symp. Mixed Initiative Interaction*, Stanford, CA, USA, 1997, pp. 89–94. 1

[5] M. Jouvet, *The Paradox of Sleep: The Story of Dreaming*, ser. Bradford book. MIT Press, 2000. 1

[6] T. Schmidt, K. Hertkorn, R. Newcombe, Z. Marton, M. Suppa, and D. Fox, "Depth-based tracking with physical constraints for robot manipulation," in *Proc. 2015 IEEE Int. Conf. Robotics and Automation (ICRA)*, Seattle, USA, May 2015, pp. 119–126. 1

[7] A. S. Bauer, P. Birkenkampf, A. Albu-Schäffer, and D. Leidner, "Bridging the gap between supervised autonomy and teleoperation," in *Proc. FAIM/ISCA Workshop Artificial Intelligence for Multimodal Human Robot Interaction*, 2018, pp. 44–47. 2

[8] M. A. Goodrich, J. W. Crandall, and E. Barakova, "Teleoperation and beyond for assistive humanoid robots," *Reviews of Human factors and ergonomics*, vol. 9, no. 1, pp. 175–226, 2013. 2

[9] S. Hayati and S. T. Venkataraman, "Design and implementation of a robot control system with traded and shared control capability," in *Proc. 1989 Int. Conf. Robotics and Automation*, May 1989, pp. 1310–1315 vol.3. 2

[10] J. A. Wolfe, B. Marthi, and S. J. Russell, "Combined task and motion planning for mobile manipulation." in *Proc. 20th Int. Conf. Automated Planning and Scheduling (ICAPS)*, 2010, pp. 254–258. 2

[11] L. P. Kaelbling and T. Lozano-Pérez, "Integrated task and motion planning in belief space," *Int. J. Robotics Res.*, vol. 32, no. 9-10, pp. 1194–1227, 2013. 2

[12] D. Leidner, A. Dietrich, F. Schmidt, C. Borst, and A. Albu-Schäffer, "Object-centered hybrid reasoning for whole-body mobile manipulation," in *Proc. 2014 IEEE Int. Conf. Robotics and Automation (ICRA)*, May 2014, pp. 1828–1835. 2

[13] D. Leidner, C. Borst, and G. Hirzinger, "Things are made for what they are: Solving manipulation tasks by using functional object classes," in *Proc. 2012 IEEE-RAS Int. Conf. Humanoid Robots*, Nov 2012, pp. 429–435. 2, 3

[14] D. Leidner, "Cognitive reasoning for compliant robot manipulation," Ph.D. dissertation, Diss., Univ. Bremen, Bremen, 2017. 2

[15] G. Konidaris, L. P. Kaelbling, and T. Lozano-Perez, "From skills to symbols: Learning symbolic representations for abstract high-level planning," *J. Artificial Intell. Res.*, vol. 61, pp. 215–289, 2018. 2

[16] O. Lipmann and H. Bogen, *Naive Physik: Arbeiten aus dem Institut für Angewandte Psychologie in Berlin; theoretische und experimentelle Untersuchungen über die Fähigkeit zu intelligentem Handeln.* JA Barth, 1923. 2

[17] P. J. Hayes, "The naive physics manifesto," in *Expert Systems in the Microelectronic Age*, D. Michie, Ed. Edinburgh University Press Edinburgh, 1979, vol. 220, pp. 242–270. 2

[18] ——, "The Second Naive Physics Manifesto," in *Formal Theories of the Commonsense World*, J. R. Hobbs and R. C. Moore, Eds. Norwood, New Jersey: Ablex, 1985, pp. 1–36. 2

[19] P. W. Battaglia, J. B. Hamrick, and J. B. Tenenbaum, "Simulation as an engine of physical scene understanding," *Proc. Nat. Academy of Sciences*, vol. 110, no. 45, pp. 18 327–18 332, 2013. 2

[20] B. Johnston and M.-A. Williams, "A generic framework for approximate simulation in commonsense reasoning systems." in *Proc. 2007 AAAI Spring Symp.: Logical Formalizations of Commonsense Reasoning*, Stanford, USA, Mar 2007, pp. 71–76. 2

[21] L. Mösenlechner and M. Beetz, "Using physics- and sensor-based simulation for high-fidelity temporal projection of realistic robot behavior," in *Proc 19th Int. Conf. Automated Planning and Scheduling*, Thessaloniki, Greece, Sep 2009, pp. 249–256. 2

[22] ——, "Parameterizing Actions to have the Appropriate Effects," in *Proc. 2011 IEEE/RSJ Int. Conf. Intelligent Robots and Systems*, San Francisco, CA, USA, Sep 2011. 2

[23] ——, "Fast temporal projection using accurate physics-based geometric reasoning," in *Proc. 2013 IEEE Int. Conf. Robotics and Automation*, Karlsruhe, Germany, May 2013, pp. 1821–1827. 2

[24] L. Kunze, M. E. Dolha, E. Guzman, and M. Beetz, "Simulation-based temporal projection of everyday robot object manipulation," in *Proc Int. Conf. Autonomous Agents and Multiagent Systems*, vol. 1, Taipei, Taiwan, May 2011, pp. 107–114. 2

[25] M. Pomarlan, D. Nyga, M. Picklum, S. Koralewski, and M. Beetz, "Deeper Understanding of Vague Instructions through Simulated Execution (Extended Abstract)," in *Proc. 2017 Int. Conf. Autonomous Agents & Multiagent Systems*, Sao Paulo, Brazil, May 2017, pp. 1694–1696. 2

[26] N. Koenig and A. Howard, "Design and use paradigms for gazebo, an open-source multi-robot simulator," in *Proc. 2004 IEEE/RSJ Int. Conf. Intelligent Robots and Systems*, Sendai, Japan, Sep 2004, pp. 2149–2154. 4

[27] S. Lemaignan, R. Ros, E. A. Sisbot, R. Alami, and M. Beetz, "Grounding the interaction: Anchoring situated discourse in everyday human-robot interaction," *Int. J. Social Robotics*, vol. 4, no. 2, pp. 181–199, 2012. 4

[28] H.-G. Beyer and H.-P. Schwefel, "Evolution strategies – a comprehensive introduction," *Natural Computing*, vol. 1, no. 1, pp. 3–52, Mar 2002. 5

[29] T. Laue and M. Hebbel, "Automatic parameter optimization for a dynamic robot simulation." in *RoboCup*. Springer-Verlag Berlin Heidelberg, 2008, pp. 121–132. 5

[30] S. Chen, J. Montgomery, and A. Bolufé-Röhler, "Measuring the curse of dimensionality and its effects on particle swarm optimization and differential evolution," *Applied Intelligence*, vol. 42, 04 2015. 5

[31] H.-P. Schwefel, "Adaptive Mechanismen in der biologischen Evolution und ihr Einfluss auf die Evolutionsgeschwindigkeit," *Interner Bericht der Arbeitsgruppe Bionik und Evolutionstechnik am Institut für Mess- und Regelungstechnik Re*, vol. 215, no. 3, 1974. 5

[32] H.-G. Beyer, "Evolution strategies," *Scholarpedia*, vol. 2, no. 8, p. 1965, 2007, revision #130731. 5

[33] D. Q. Huynh, "Metrics for 3d rotations: Comparison and analysis," *Journal of Mathematical Imaging and Vision*, vol. 35, no. 2, pp. 155–164, 2009. 5

[34] J. M. Selig, *Geometrical Methods in Robotics*. New York, NY: Springer Verlag, 1996. 5

[35] J. Wang and E. Olson, "AprilTag 2: Efficient and robust fiducial detection," in *Proc. IEEE/RSJ Int. Conf. Intelligent Robots and Systems (IROS)*, October 2016. 6