Simulation-based physics reasoning for consistent scene estimation in an HRI context

Yoan Sallami[1], Séverin Lemaignan[2], Aurélie Clodic[1], Rachid Alami[1]

Abstract— **Reasoning about spatial and geometric relations between objects in a tabletop human-robot interaction is a challenge due to the perception not being always consistent: objects placed on a table seem to be slightly in the air; they overlap; they disappear due to occlusions. Yet, interpreting and anchoring perceptual data in a physically consistent estimation of the scene is a crucial ability for humans, and thus robots in HRI context. In this paper we present a simulation-based physics reasoner integrated in a lightweight situation-assessment framework called Underworlds, that allows the robot to stabilize objects and build at run-time a consistent estimation of the scene, even for entirely hidden objects, while inferring the actions performed by its human partner.**

I. INTRODUCTION

This work is part of an initiative to incrementally build and refine a software architecture for a cognitive and interactive robot which achieves collaborative activities with human partners [9].

In this context we present a situation-assessment component which builds a consistent estimation of the scene observed by the robot with the help of a physics engine. During the interaction, the physics engine is used in real-time to reason about objects occlusions, gravity, collisions and other associated geometric features, such as the surfaces on which objects are laid or the contents of boxes or containers.

Besides, the system needs to deal with perception issues specifically arising when humans manipulate objects in the robot's vicinity. Indeed, actions on objects like $Pick$,$Place$ or $Release$ can lead to perceived violations of spatial constraints (like an object left 'floating' in the air because it is not perceived anymore by the robot once inside of a container) that are naturally resolved with the help of physics reasoning. Along the same lines, our approach makes it possible to compute allocentric (ie viewpoint-independent) spatial relations like $isOnTop$ or $isIn$ that play an important role in scene understanding..

As such, our contribution is an open-source, modular situation-assessment component called *physics_reasoner*[1] as a plugin for the lightweight situation-assessment framework Underworlds[2] [8]. It is fully integrated with ROS and only requires URDF files to work. This component use an efficient deterministic physics engine, Bullet RT Simulation [2] to stabilize and infer poses for out-of-sight objects in human-robot tabletop interaction settings.

In such a context, objects can be held by a human or the robot or can be placed on a surface (a table, a box, the floor) or on another object, or can be put in a container (also perceived as an object). Let us assume that the robot is able to identify and localize the objects using an RGBD perception system. The physics-based reasoner presented here allows not only to track and maintain a symbolic state of the environment in a manner that is more robust to perception inaccuracies, but also to estimate it even in case of object occlusions, as commonly arising in dynamic environments.

In this this paper, we present a high level reasoning pipeline which analyzes geometric violations and corrects the objects' poses, even when out of sight. It has been designed as an extension of a pre-existing perception pipeline to provide corrected object poses, interpretation of the scene spatial configuration, and recognition of human actions by analyzing the transition of objects from physically plausible to not physically plausible states. Examples of such capabilities include: inferring that a hidden object has likely fallen onto the floor (Fig. 6a); inferring that an object is being transferred from one container to another (Fig. 6c). Based on the output of the reasoner, we can also compute estimations of other key information such as visibility or reachability [11] for non-observable objects: the physics-based reasoner makes it possible for the robot to continue to estimate the visibility or reachability of an object by its human partner even if the object is no more visible to the robot.

In the next section, we discuss the rationale for equipping the robot with such reasoning ability and we briefly review related work on similar simulation-based reasoners.

In section III we will present the design choices and the complete architecture with an example of implementation. In section IV we will present the implementation of the algorithm. We then present a study demonstrating the effectiveness of the system with results on challenging situations (Section V). Section VI discusses the main findings of this research and outlines future work directions.

II. RELATED WORK

Interpreting visual information relying on physical reasoning is a foundational cognitive ability that allows humans to anchor perception information into a consistent model of the world [16]. Recent research [17] explores the hypothesis that humans have their own 'physics engine' which works in a

*This work was supported by the european H2020 project MuMMER (grant 688147)

[1]Authors are with LAAS-CNRS, CNRS, Toulouse, France `firstname.surname@laas.fr`

[2]Author is with Bristol Robotics Lab, University of the West of England, Bristol, United Kingdom `severin.lemaignan@brl.ac.uk`

[1]https://github.com/underworlds-robot/uwds_physics_clients

[2]https://github.com/underworlds-robot/uwds

similar way as a video game engine by simulating Newtonian principles in real-time. The authors emphasize the similarity between the human brain and a real-time physics engine which would use rules and heuristics.

In robotics, rigid body simulation is often sufficient to cover a large number of use cases and several mature and open-source physics engines (ODE, Bullet, PhysX) are readily available. Simulation-based techniques for on-line spatial inference are today viable options, and appear as a natural choice to implement a simple yet effective *physical intuition* in a HRI context.

Generating the symbolic facts and events in real time is indeed a key requirement for a HRI system, as it serves multiple purposes for a range of on-line tasks, like grounding verbal expressions [7] or symbolic task planning [6].

In the literature, simulation-based physics reasoning is mostly used for planning purposes. [13] integrates collision, friction, center of gravity and forces to planning. In [14], Mösenlechner and Beetz use physics simulation to sample hypothetical states of the world based on a symbolic plan in order to parametrize high level actions. In a related domain, physics reasoning has also been used to predict the effects of an action [5]. In [12], [3], the authors suggest however that simulation-based physics reasoning is suited to real use case, due to the non-deterministic behavior of its predictions. As the authors of [3] explain, pure simulation-based reasoning is often impractical, and knowledge-based rules are needed in order to make the simulation usable. We claim that even if it is not accurate, a simulation with minimal symbolic reasoning is sufficiently correct to allow for a consistent estimation of spatial high-level symbolic facts and events needed to supervise a task. We show it is also sufficient to prime the location of out-of-sight objects to the robot, enabling more efficient search procedures when the robot needs to find a specific object (e.g. 'the robot knows for sure that the object is on the floor'). In [4], physical reasoning is used to enhance knowledge about manipulation tasks, by acquiring simulated real world data in a game engine, but not to enhance execution in the presence of real humans. In contrast, our reasoner is able to manage occlusions and complex object interactions while the human is collaborating with a robot, also inferring the potential geometric inconsistency caused by the human's actions.

III. DESIGN AND ARCHITECTURE

In this section we first present briefly the specifics of Underworlds, followed by the design choices we have made regarding the physics engine. We then introduce our entire reasoning pipeline.

A. *Cascading situation-assessment*

Underworlds is a novel framework which focuses on maintaining and distributing multiple (and possibly alternative) spatial (based on 3D bounding boxes and/or 3D meshes) and symbolic models of the physical world (based on events). It works as a distributed system where a set of loosely coupled *clients* provide ad-hoc reasoning capabilities (See Fig. 1).

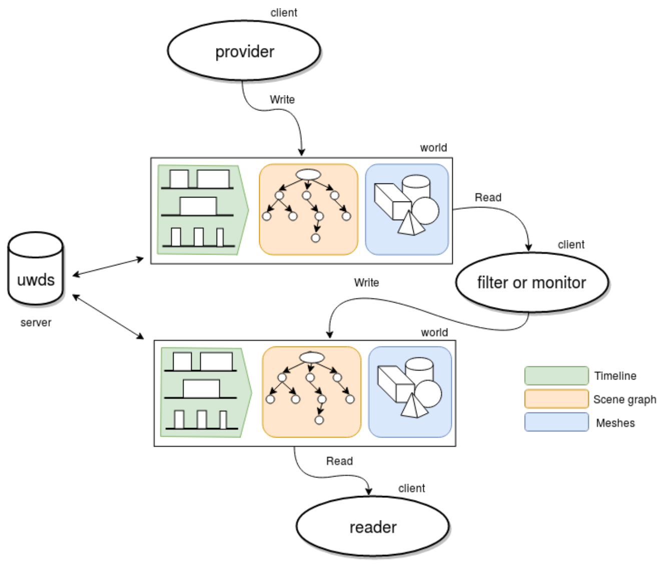


Fig. 1: Underworlds is a client/server architecture which aims to maintain and distribute several world states. Since world states share the same structure, it is possible to dynamically reconfigure the reasoning pipeline and to adapt it to a new task. Meshes are centrally stored by the server and accessible on-demand, avoiding data bottlenecks.

This approach allows to dynamically combine situation-assessment components depending on the task requirements; and to implement quickly new reasoners, as presented in [8]. In many ways, Underworlds can be viewed as a set of world states where geometric and symbolic models are tightly coupled. In order to represent the geometry of the scene, Underworlds uses a scene graph with each node referring to its parent with a position, velocity and acceleration (plus their respective covariance) and a timeline of temporal events, to represent symbolic temporal information about the world which can change over time.

In its current version, Underworlds is fully integrated with ROS, has a Python and C++ client API, and benefits from zero-copy pointer passing for C++ clients (to make communication time to distribute the data negligible with respect to the reasoning time in the clients). This allows to cascade reasoners without worrying about intra-process communication. This type of architecture can be viewed as an extension of the traditional low-level perception pipeline of an intelligent robot, allowing to design in the same modular way high level reasoning by cascading modular components.

Since Underworlds maintains in parallel multiple models of the world, we can have access at any time to raw perception and to any stage of the reasoning pipeline either to reason about multiple world states or for introspection purposes.

B. *Physics engine*

We decided to follow [18] and we have chosen the Bullet RT physics engine, because (1) game-oriented physics engine are optimized towards large scale simulations (hundreds of bodies), and (2) contrary to usual simulation in robotics, speed and stability are preferable to accuracy in our context. Besides, Bullet is already integrated into ROS (the TF library uses Bullet datatypes, for instance), which facilitates future reuse of this work. When it comes to real-time physics simulation, Bullet RT Physics is itself decentralized. This is an advantage in our case as different specialized client reasoners can use it without having to rely on a central server – every client can instantiate a physics server as needed, and distribute its output to other clients, relying on Underworlds to this effect.

Note that choosing the right simulation engine depends on the exact type of reasoning and type of objects (ie rigid, deformable, liquids) required by the environment. As such, others physics engine can be used like PhysX, ODE or MuJoCo. Since Underworlds provides an abstraction layer, different simulation engines can be used in different clients depending on the needs.

C. Input data

The input of our *physics_reasoner* component is a ROS URDF file and a 6D pose tracker for objects of interest. To generalize to any perception algorithm, Underworlds uses a special kind of clients called *providers* which bring the scene data into the system (e.g. convert the 6D object pose into a node of the scene graph, convert the object model to a 3D bounding box or a 3D mesh, or bind an event from an external reasoner into the timeline).

In this work, we rely on simple perception algorithms to put the focus on the underlying concepts. As such, perception is simplified by using either objects with AR tags or objects whose unique color can be used to cluster and segment a RGBD point cloud (in that later case, the detected objects have a fixed orientation).

D. Reasoning pipeline

In Underworlds, the clients are defined by their roles. The *providers* bring data into the system in the form of 'worlds' (nodes in a scene graph and an attached timeline of events), the *filters* are reasoners that alter/enhance existing worlds, the *monitors* generate symbolic knowledge from the worlds and the *readers* use the data to provide external functions like introspection or to bind the output to an external system. However, that classification is flexible and filters can also generate symbolic knowledge from the scenes if it is relevant to their reasoning (see Fig 1).

As previously mentioned, Underworlds can be seen as a high level reasoning extension of the traditional low-level perception pipeline. During the first stage, *providers* rely on the existing perception pipeline to detect objects and extract relevant spatial features, typically reading from ROS topics. *providers* can also collate additional static descriptions of the environment from a range of 3D file formats, including URDF. We choose to have one provider per modality (allowing for asynchoronous updates between modalities) which outputs a simple world (containing only a few objects). The *world_merger* node asynchronously fuses these worlds into a single *merged* world.

Finally the *physics_reasoner* is triggered after each update of the *merged* world, to correct object poses, infer out-of-sight objects poses and humans' actions. This results in a final, stabilized, world called *merged_stable* (Fig. 2).

IV. IMPLEMENTATION

In this section we present the details of the implementation and the algorithms. First we introduce the predicates used in the systel and how they are computed. Then, we describe the algorithm used to compute the output scene (*merged_stable*) and to infer the actions that explain physical inconsistency (Section IV-D).

A. Predicates

In order to build a physically plausible estimation of the scene and infer actions[3], this component computes at each reasoning step two predicates for each object:

- $isPerceived(object)$ true when the object was recently seen by the robot.
- $isPhysicallyPlausible(object)$ true when the object is in a stable configuration with respect to the scene.

Based on these predicates, the system is able to infer the following actions (to explain physical inconsistencies) :

- $Pick(object)$ when an object is picked up
- $Place(object)$ when an object is placed on a surface
- $Release(object)$ when an object is not held anymore

Then it computes allocentric spatial relations, based on a physically plausible estimation of the scene (Section IV-C):

- $isIn(object, object)$ true when an object contains another object
- $isOnTop(object, object)$ true when an object lies on a surface

B. Computation of stability

In order to know if an object is at a physically plausible state, we assume that if the object is at a stable state in the simulation, then the configuration is physically plausible.

To estimate whether an object is in a stable configuration, we execute as fast as possible a number of simulation steps in the future. The number of steps to execute is given by:

$$n_{steps} = P_{horizon}/S_{step} \qquad (1)$$

Where $P_{horizon}$ is the prediction horizon and S_{step} the duration that one step simulates. The computation time of the simulation relies heavily on a trade-off between the simulation step (which needs to be small to prevent missed collisions) and the prediction time (which needs to be long enough to e.g. give time to objects to fall).

In order to know if an object is in a *physically plausible state* (the value of the predicate $isPhysicallyPlausible$), we

[3]The predicates used are necessary for this reasoner but not sufficient in a HRI context. Other Underworlds clients presented in [8] are in charge of computing $isVisible$ or $lookAt$ that are essential when reasoning with humans

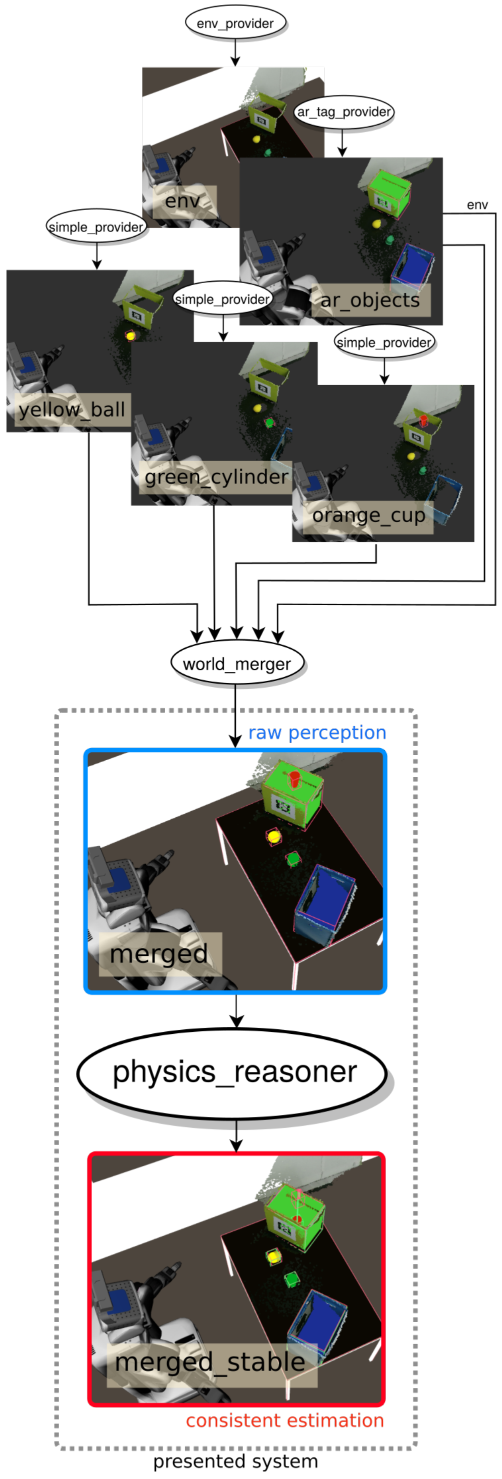


Fig. 2: The reasoning pipeline used in this experiment. During the first stage, the providers convert raw perception data into Underworlds's scene graph nodes; the nodes are merged into one (complete) world model; this 'raw' world model is fed to the physics reasoner, which generates an enhanced world model (e.g. spatial inconsistencies are corrected).

monitor the divergence in position between the position of the object at the end of the simulated steps and the perceived position ($\overrightarrow{d_{sim/perc}}$) (Alg. 1). This approach is similar to the one used by [12]. However, in our case, we evaluate it at each step of the simulation because we need to avoid as much as possible disturbances in the simulation scene caused by the objects falling during the first steps of the process (before being considered as not stable). To do so we override the position and velocity of the objects considered as unstable with perception data as soon as considered unstable.

Algorithm 1 $isPhysicallyPlausible$ computation

```
for $n_{steps}$ (see Eq. 1) do
  step simulation for $S_{step}$ seconds
  for all $object$ in the input scene graph do
    if $\left\|\overrightarrow{d_{sim/perc}(object)}\right\| > D^{max}_{sim/perc}$ then
      $isPhysicallyPlausible(object) = false$
    end if
    if not $isPhysicallyPlausible(object)$ then
      override $object$ simulation with perceived data
      for all $object_{contained}$ in $object$ contents do
        move $object_{contained}$
        simulation of $\overrightarrow{d_{sim/perc}}(object)$
      end for
    end if
  end for
end for
```

C. Support and contents computation

Since the object bounding boxes are corrected by the simulation engine (the meshes that overlap are popped up and the floating objects are placed on their support) we can compute the contents and placement support relations with an efficient classic approach based on 3D world bounding boxes tests as used in [15] (Fig. 3).

If an inconsistency is generated by the perception (one mesh perceived inside another) the simulation engine will correct it, thanks to the penetration and collision tests which are performed by Bullet.

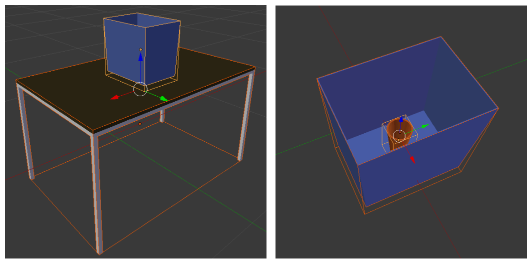


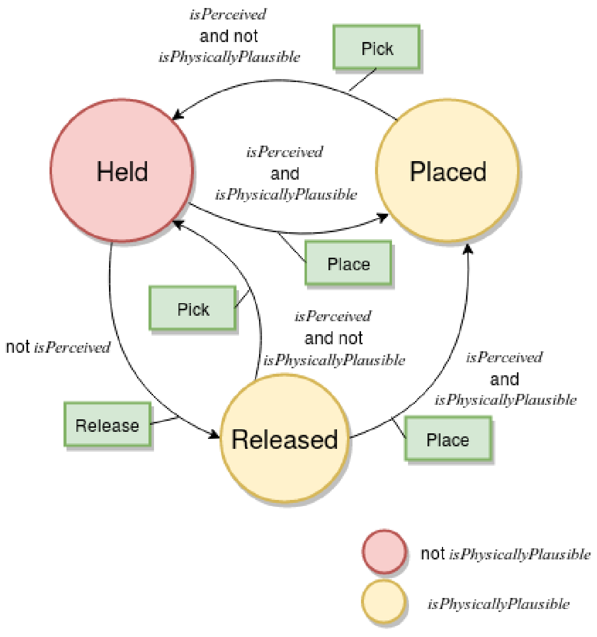
Fig. 3: The allocentric relations $isOnTop$ and $isIn$ are computed based on simple and efficient bounding boxes test.

D. Output scene computation and action inference

To generate the output scene (Alg. 2), we apply the following reasoning: if the object is in a physically plausible configuration at the end of the simulation steps, we use the resulting computed pose; otherwise use the perceived pose. In that case, and if the object is left in a physically implausible state, we seek to explain the inconsistency by looking for a human action that would explain the state.

In addition, when an object's position jumps out (i.e. its simulated displacement $\overrightarrow{d_{perc/prev}}$ is greater than a threshold $D^{max}_{perc/prev}$), we also move *contained* objects as well, if any.

The inference of the human actions builds on the assumption that physical inconsistencies are caused by a human manipulating objects. Specifically, when an object is perceived as being in a non-plausible state, we apply the heuristics described in Fig. 4.



isPerceived and not isPhysicallyPlausible
Pick
Held
isPerceived and isPhysicallyPlausible
Placed
Place
Pick
isPerceived and not isPhysicallyPlausible
not isPerceived
isPerceived and isPhysicallyPlausible
Release
Released
Place
not isPhysicallyPlausible
isPhysicallyPlausible

Fig. 4: The state-machine used to infer human actions based of the object state. The consistency is checked by monitoring the distance between what is actually perceived and the internal simulation of the robot (See Alg. 1).

E. Parameters

Table I lists the reasoner parameters used in our experiment. In order to have a correct behavior, the prediction horizon needs to be long enough to make the objects fall while been short enough to speed up the reasoning process. These values depend on the CPU/GPU combination used, and, combined with the *de facto* non-deterministic behaviour of Bullet's collision detection, our results might not be precisely reproducible.

V. EXPERIMENTATION AND RESULTS

In this section, we present the study setup (Section V-A) and several challenging physical situations that our reasoner can process. Note that the specific objects used in this study are simple (plain, colorful boxes and cups). The reasoner is however entirely decoupled from the perception (through Underworlds), and as such, our findings would apply to any objects, as long as the robot perception stack is able to acquire and track their 3D meshes.

Algorithm 2 Output scene computation

```
for all $object$ in the input scene graph do
    if object last observation < $T^{max}_{perceived}$ then
        $isPerceived(object) = true$
    else
        $isPerceived(object) = false$
    end if
    if $isPerceived(object) = true$ then
        Place object where perceived
        for all $object_{contained}$ in $object$ contents do
            if $\left\|\overrightarrow{d_{perc/prev}}(object)\right\| > D^{max}_{perc/prev}$ then
                move $object_{contained}$ by $\overrightarrow{d_{perc/prev}}(object)$
            end if
        end for
    end if
end for
Update $isPhysicallyPlausible$ (Alg. 1)
for all $objects$ in input scene graph do
    if $isPerceived(object)$ and not $isPhysicallyPlausible(object)$ then
        Set $object$ to perceived pose
    else
        Set $object$ to simulated pose
    end if
end for
Compute $isOnTop$ and $isIn$ on a physically plausible world (See Section IV-C)
```

TABLE I: Reasoner parameters used in the experiment

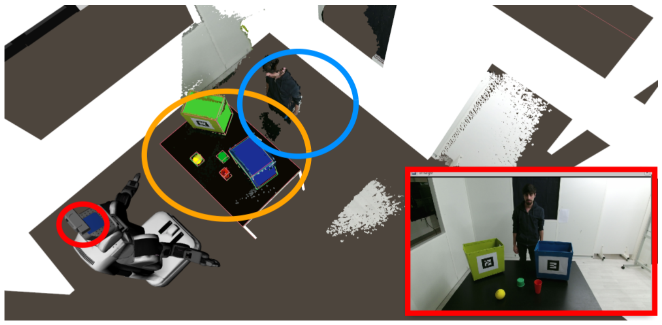
Parameter	Value	Description
$T^{max}_{perceived}$	$0.7[s]$	Perceived max duration
$D^{max}_{sim/perc}$	$0.045[m]$	Simulation tolerance
$D^{max}_{perc/prev}$	$0.032[m]$	Perception tolerance
$P_{horizon}$	$0.08[s]$	Prediction horizon
S_{step}	$0.00416[s]$	Simulation step

A. Experimental setup

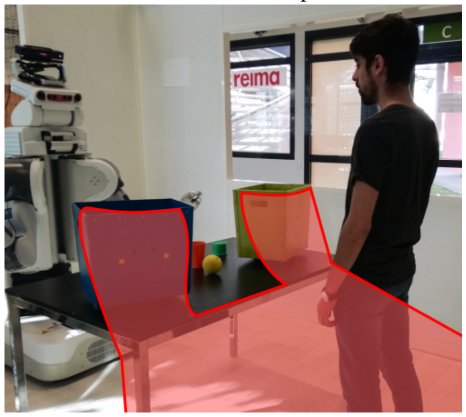
For this study, we use a PR2 robot, with a Kinect2 placed on its head as the only camera input. The head is static and directed towards the table, where the interaction takes place. From the point of view of the robot, a part of the scene is occluded by the boxes and the table (see Fig. 5). We have used this configuration as it is a classical setup for tabletop human-robot collaboration. The perception algorithms run on a laptop computer to which the RGBD sensor was directly plugged and the reasoning pipeline runs on a desktop computer. Here the aim is to have the perception pipeline and the reasoning pipeline running on their own nodelet manager.

B. Objects model

For this experience, object models are based on handcrafted URDF files, that include collision/inertial information and a CAD model like the one used to describe the robots.



(a) The red circle indicates where the Kinect2 RGBD sensor is positioned, and the red rectangle indicates the camera view of the sensor. The orange circle indicates where interaction takes place, and the blue circle where the human is placed.



(b) From the point of view of the robot, a part of the scene is occluded (colored in red in the picture).

Fig. 5: The experimental setup

They have only one main link, but could be more complex with moving parts described as a set of joints and links. For containers, the collision model is approximated by a set of boxes (one per wall), as Bullet would otherwise approximate the mesh to its convex hull, 'closing' the container, and preventing other objects to be effectively contained in it. In future work, meshes can be estimated on-line, from the sensor point cloud [10], and fed directly to Underworlds, without having to rely on handcrafted CAD models.

C. *Selected challenging situations*

Fig. 6 presents different qualitative results for challenging interactions. In these use cases, reasoning about physics and gravity is crucial in order to correctly infer over time positions and velocity of the objects. These situations have been chosen because they could occur in a classic tabletop human-robot interaction setting.

In such dynamic situations, the physics-based reasoner demonstrates that it can not only correct the scene geometry and maintain a symbolic state of the environment in a robust manner but also estimate the object poses even if completely hidden.

VI. Summary and Future Work

We have presented a preliminary work on a simulation-based physics reasoner integrated in a situation-assessment framework called Underworlds and illustrated how it can be already used to provide a more physically plausible world state in human-robot interaction context, as it is able to deal with sensory data inaccuracies and potential inconsistencies between different sensor sources by correcting objects poses. It is also able to estimate the effects of the perceived object motions on completely hidden objects while inferring the actions performed by the human partner.

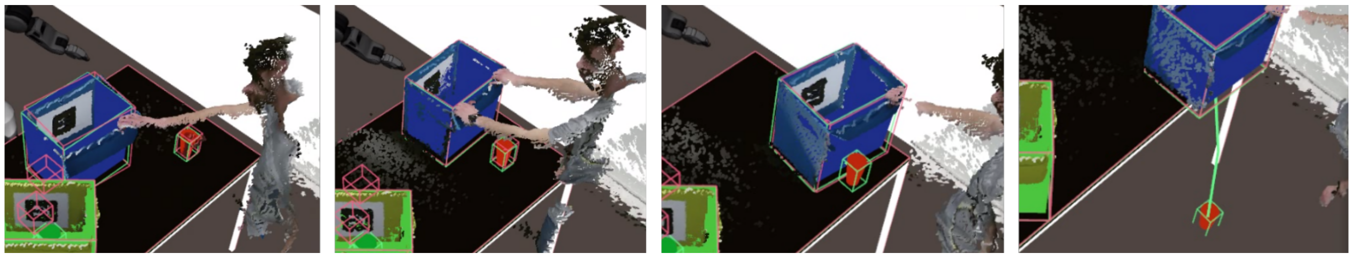
We discuss below some limitations of the system in its current preliminary state as well as future work. First we introduce how we could enhance the simulation engine with stochastic reasoning and secondly we discuss the future steps of the reasoning pipeline.

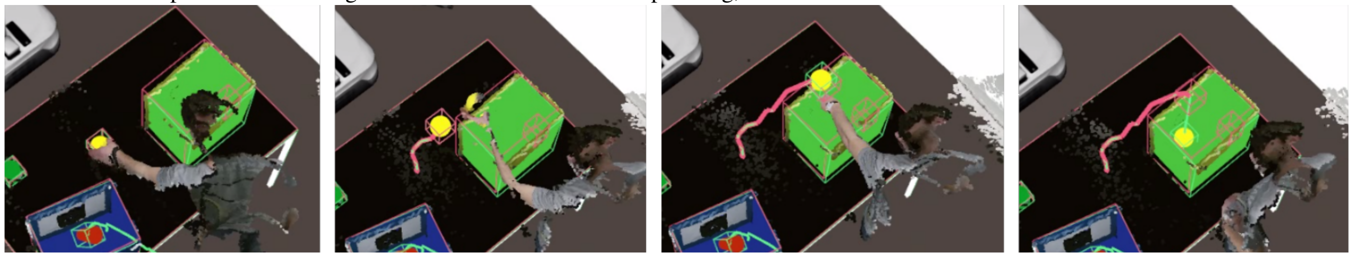
a) Uncertainty and simulation engine limits: Bullet does not handle uncertainty and consequently we do not use the covariances of the scene graph. However, since Underworlds handles uncertainty in its data-structure we could, in the future, benefit from stochastic physics simulation like the one presented in [1] to benefit from a more accurate friction model. However, even with stochastic models, reasoning about rolling objects which are chaotic and unpredictable in their behavior, will still be an issue. Due to this constraint we have chosen to have a high rolling friction in the simulation which almost disables rolling behavior.

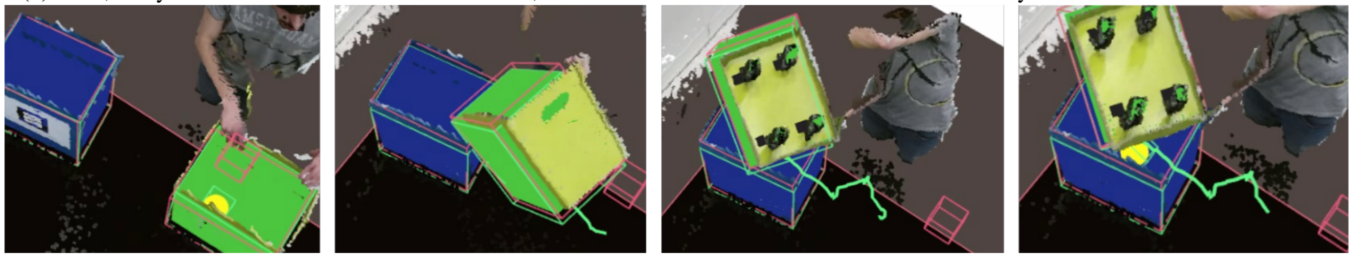
b) Enhancing tracking algorithms: The reasoning pipeline presented above takes as input properly identified and localised objects: we assume that the low-level perception pipeline takes care of performing the appropriate tracking and filtering stages, and the physics reasoner does not concern itself with dealing with e.g. noisy perception, as this would be the role of the trackers' filters.

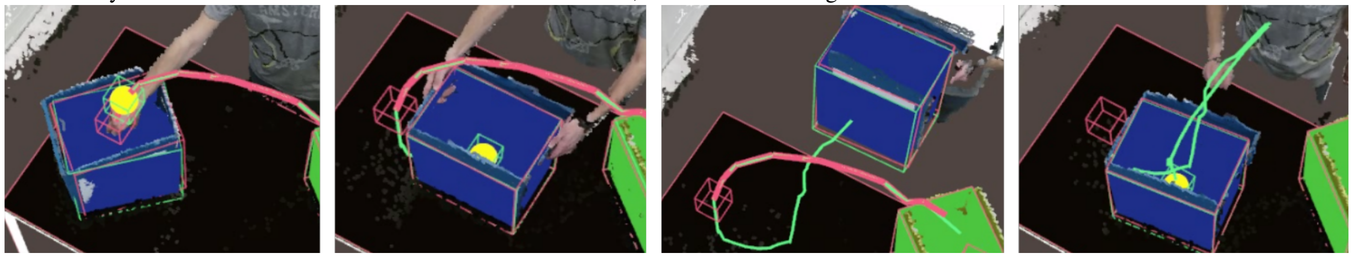
Combined tracking and filtering is typically performed with a Kalman filter, that takes into account the motion of the objects and noisy observations to predict a more accurate object position and velocity. The filter has however no information about the physics of the scene, and more importantly, whenever the object is occluded, the filter cannot update its motion model. As our pipeline can infer the pose of out-of-sight objects, it would seem that a natural extension of the existing low-level tracking algorithm could benefit from a simulation 'feedback' from the physics reasoner, to update the motion models of all the objects, even the occluded ones. Such a physics-aware Kalman filter would lead to enhanced object tracking while smoothing the object motion for the physical reasoner. This approach would rely on a close interaction loop between the physics reasoning and the low-level perception. We plan to investigate this idea in a near future.

c) Exploiting human model for action detection: In classical action detection tasks, the aim is not only to know, for example, that an object has been picked, but also who picked it, and to generate a triplet $< subject, action, object >$. Classically, these two tasks are handled together, by jointly classifying temporal human motion with respect to objects. We believe that it is however easier when we already known that an object is in an inconsistent state, as the generated event ($Pick$ in our case)


(a) In this example, the robot first perceives the blue box and the orange cup. The human moves the box that hides the cup, and pulls it back until the cup falls. Even though the robot did not see the cup falling, the reasoner infers that it is on the floor.


(b) Here, the yellow ball is first seen above the box, then the human release it. The reasoner correctly infers the location of the ball.


(c) In this example, the yellow ball is known to be in the green box. The human empties the green box into the blue one. The reasoner successfully infers that the ball is now inside the blue container, without ever seeing the ball.


(d) In this example, the reasoner successfully estimates the movement of the yellow ball, while inside the blue box.

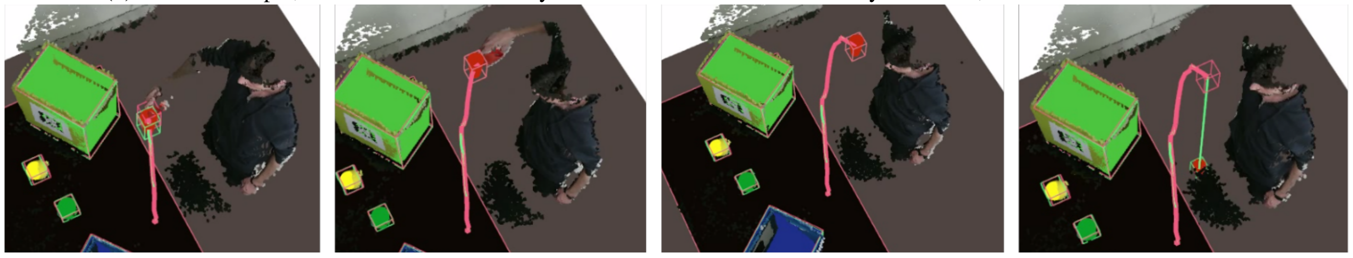

(e) This last example illustrates one of the limitations of the system: the human intentionally hides the cup behind himself; the reasoner fails to infer that it is still held, and instead computes that the cup must have fallen to the floor.

Fig. 6: Examples of challenging inferences. Only the RGBD sensor mounted on the PR2 head is used. The pink trajectories represent the observed trajectories, with the corresponding bounding boxes in pink; the green trajectories are those computed by the physics reasoner (and accordingly, the green bounding boxes).

can trigger a 'human identification' task (identifying who picked up the object, by matching the hands closest to the object to their owner – or reporting an inconsistency if no human is in the vicinity). To know if an object is actually released, we could monitor the distance between the hands and the object, if at least one hand is near we keep the attachment, otherwise we release it and output the object simulated pose. We will investigate this solution in the near future. This would solve the limitations of the actual algorithm when the object is occluded after being hold (See Fig. 6e) by adding a new filter that manage the attachment of objects to humans/robot hands.

ACKNOWLEDGMENT

This work is funded by the European Horizon 2020 project MuMMER (grant 688147). Many thanks to Matthieu Herrb and Jules Waldhart for their support during the implementation of the system and the setup of the study.

REFERENCES

[1] Anurag Ajay, Jiajun Wu, Nima Fazeli, Maria Bauza, Leslie P Kaelbling, Joshua B Tenenbaum, and Alberto Rodriguez. Augmenting physical simulators with stochastic neural networks: Case study of planar pushing and bouncing. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3066–3073. IEEE, 2018.

[2] Erwin Coumans and Yunfei Bai. Pybullet, a python module for physics simulation for games, robotics and machine learning, 2016–2018.

[3] Ernest Davis and Gary Marcus. The scope and limits of simulation in cognitive models. *CoRR*, abs/1506.04956, 2015.

[4] Andrei Haidu, Daniel Beßler, Asil Kaan Bozcuoğlu, and Michael Beetz. Knowrobsim—game engine-enabled knowledge processing towards cognition-enabled robot control. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4491–4498. IEEE, 2018.

[5] Lars Kunze and Michael Beetz. Envisioning the qualitative effects of robot manipulation actions using simulation-based projections. *Artificial Intelligence*, 247:352–380, 2017.

[6] Raphaël Lallement, Lavindra De Silva, and Rachid Alami. HATP: An HTN Planner for Robotics. In *2nd ICAPS Workshop on Planning and Robotics*, Portsmouth, United States, June 2014.

[7] Séverin Lemaignan, Raquel Ros, Emrah Akin Sisbot, Rachid Alami, and Michael Beetz. Grounding the Interaction: Anchoring Situated Discourse in Everyday Human-Robot Interaction. *2012 International Journal of Social Robotics*, 4(2):181–199, April 2012.

[8] Séverin Lemaignan, Yoan Sallami, Christopher Wallbridge, Aurélie Clodic, Tony Belpaeme, and Rachid Alami. UNDERWORLDS: Cascading Situation Assessment for Robots. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Madrid, Spain, October 2018.

[9] Séverin Lemaignan, Mathieu Warnier, E. Akin Sisbot, Aurélie Clodic, and Rachid Alami. Artificial cognition for social human–robot interaction: An implementation. *Artificial Intelligence*, 247:45–69, 2017.

[10] Zoltan-Csaba Marton, Dejan Pangercic, Nico Blodow, Jonathan Kleinehellefort, and Michael Beetz. General 3d modelling of novel objects from a single view. In *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3700–3705. IEEE, 2010.

[11] Grégoire Milliez, Matthieu Warnier, Aurélie Clodic, and Rachid Alami. A framework for endowing an interactive robot with reasoning capabilities about perspective-taking and belief management. In *The 23rd IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN)*, pages 1103–1109. IEEE, 2014.

[12] Lorenz Mösenlechner and Michael Beetz. Fast temporal projection using accurate physics-based geometric reasoning. In *2013 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1821–1827. IEEE, 2013.

[13] Lorenz Mösenlechner and Michael Beetz. Using physics- and sensor-based simulation for high-fidelity temporal projection of realistic robot behavior. In *19th International Conference on Automated Planning and Scheduling (ICAPS'09)*, 2009.

[14] Lorenz Mösenlechner and Michael Beetz. Parameterizing actions to have the appropriate effects. In *2011 IEEE International Conference on Intelligent Robots and Systems (ICRA)*, pages 4141–4147, 2011.

[15] E Akin Sisbot, Raquel Ros, and Rachid Alami. Situation assessment for human-robot interactive object manipulation. In *2011 IEEE International Conference on Robot and Human Interactive Communication (RO-MAN)*, pages 15–20. IEEE, 2011.

[16] Elizabeth S Spelke and Katherine D Kinzler. Core knowledge. *Developmental science*, 10(1):89–96, 2007.

[17] Tomer Ullman, Elizabeth Spelke, Peter Battaglia, and Joshua B. Tenenbaum. Mind games: Game engines as an architecture for intuitive physics. *Trends in Cognitive Sciences*, 21, 06 2017.

[18] Erik Weitnauer, Robert Haschke, and Helge Ritter. Evaluating a physics engine as an ingredient for physical reasoning. In *International Conference on Simulation, Modeling, and Programming for Autonomous Robots*, pages 144–155. Springer, 2010.