Impact-Aware Manipulation by Dexterous Robot Control and Learning in Dynamic Semi-Structured Logistic Environments

I.AM. Software Integration Policy

Dissemination level	Public (PU)
Work package	WP5:Management
Deliverable number	D5.1
Version	F-v1.0
Submission date	30/06/2020
Due date	30/06/2020

www.i-am-project.eu

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 871899

Authors

Authors in alphabetical order		
Name	**Organisation**	**Email**
Jos DEN OUDEN	TU/e	j.h.v.d.ouden@tue.nl
Pierre GERGONDET	CNRS	pierre.gergondet@gmail.com
Abderrahmane KHEDDAR	CNRS	kheddar@lirmm.fr
Claude LACOURSIÈRE	Algoryx	claude.lacoursiere@algoryx.se
Alessandro SACCON	TU/e	a.saccon@tue.nl

Control sheet

Version history			
Version	**Date**	**Modified by**	**Summary of changes**
0.1	13/05/2020	Jos DEN OUDEN	TOC & first contents
0.11	24/06/2020	Claude LACOURSIÈRE	document structure created
0.12	25/06/2020	Alessandro SACCON	introduction and modeling parts written
0.13	27-28/06/2020	Abderrahmane KHEDDAR	CNRS part (control)
0.14	29/06/2020	Pierre GERGONDET	control review and CNRS/AGX integration
0.15	29/06/2020	Abderrahmane KHEDDAR	All CNRS parts integrated
0.16	29/06/2020	Aude BILLARD	All EPFL parts integrated
0.17	29/06/2020	Saeed ABDOLSHAH	All TUM parts integrated
0.5	29/06/2020	Alessandro SACCON	Pre-final version ready for peer-review
1.0	30/06/2020	Claude LACOURSIÈRE	Revised version ready for submission, quality check

Peer reviewers		
	Reviewer name	**Date**
Reviewer 1	Teun BOSCH	
Reviewer 2	Aude BILLARD	

Legal disclaimer

The information and views set out in this deliverable are those of the author(s) and do not necessarily reflect the official opinion of the European Union. The information in this document is provided "as is", and no guarantee or warranty is given that the information is fit for any specific purpose. Neither the European Union institutions and bodies nor any person acting on their behalf may be held responsible for the use which may be made of the information contained therein. The I.AM. Consortium members shall have no liability for damages of any kind including without limitation direct, special, indirect, or consequential damages that may result from the use of these materials subject to any liability which is mandatory due to applicable law. Copyright © I.AM. Consortium, 2020.

TABLE OF CONTENTS

ABBREVIATIONS

Abbreviation	Definition
EC	European Commission
PU	Public
WP	Work Package

EXECUTIVE SUMMARY

The performance of the three I.AM. demonstrators and the complementarity of the scientific knowledge created by the I.AM. consortium depends on an successful and seamless integration of the software components that will be created all along the project. This is the reason why integration is discussed among the I.AM. partners at the very start of the project. This deliverable reports on how integration plans are going to be handled, and on the consensual agreements.

Several bilateral meetings among the academic partners and industries allowed identifying what are the software bricks that are currently available or that are in the course of development, how are they going to evolve, and how are they going to interact and feed each other in order to be smoothly integrated. The first and primary concern was the simulation framework that will allow conducting high-fidelity experiments without compromising the real robots' hardware integrity. For this, Algoryx could produced an application including the simulated dynamics of the robots used in I.AM. along with the virtual environments relevant to the scenarios investigated. This is distributed freely among the consortium for at least the duration of the project. Algoryx on its side justifies its choice to integrate within Unity 3D and Unreal Engine as these have become the reference for simulations involving 3D graphics, of course, but also deep learning, machine learning, cloud execution, and much more. This is in contrast to integrated simulation tools popular in robotics research recently.

We also discuss experimental results with parameter identification, as well as the first integration of `mc_rtc` with AGX Dynamics. This is already simulating a robot (Panda from partner Franka Emika) that is controlled using the `mc_rtc` task-space quadratic program framework developed by CNRS and to be enriched by all the partners in the course of the project. CNRS also committed to port its controller in all the I.AM. robots. Controllers for robots subject to impacts do not exist at the moment, yet, increasing the speed of robot manipulators requires the introduction of contact at non-zero relative velocity. Developing such controllers requires a considerable amount of software components which then need to be integrated. Software components themselves must be chosen carefully to meet specifications, and be validated.

Meanwhile, discussions on API development to host observers and estimators developed by TUM to serve the purpose of closed-loop impact-aware control and to be fully integrated to `mc_rtc`. Moreover integrating the `mc_rtc` to the planning and control techniques based on dynamical systems and machine learning is also discussed and investigated between partners EPFL and CNRS that have long standing collaborations and mutual understanding. Therefore, we are very confident that the I.AM. components plan, learn, sensing and control will be integrated at an early stage of the project. In parallel, TU/e is developing several software to support preliminary experiments for collecting data in the TOSS scenario (using built-in robot controller) and create knowledge that spreads and feed all the other I.AM. components. That is models to improve (i) impact-aware sensing and observation (ii) impact-aware planning and promote the interlink between learning and modeling, (iii) improve model-based control in all aspects (performance and prediction) and finally (iv) promote the interlink between robot experiments and modeling.

1 Introduction

1.1 I.AM. project background

Europe is leading the market of torque-controlled robots. These robots can withstand physical interaction with the environment, including impacts, while providing accurate sensing and actuation capabilities. I.AM. leverages this technology and strengthens European leadership by endowing robots to exploit intentional impacts for manipulation. I.AM. focuses on impact aware manipulation in logistics, a new area of application for robotics which will grow exponentially in the coming years, due to socio-economical drivers such as booming of e-commerce and scarcity of labour. I.AM. relies on four scientific and technological research lines that will lead to breakthroughs in modeling, sensing, learning and control of fast impacts:

1. I.Model offers experimentally validated accurate impact models, embedded in a high fidelity simulator to predict post-impact robot states based on pre-impact conditions;
2. I.Learn provides advances in planning and learning for generating desired control parameters based on models of uncertainties inherent to impacts;
3. I.Sense develops an impact-aware sensing technology to robustly assess velocity, force, and robot contact state in close proximity of impact times, allowing to distinguish between expected and unexpected events;
4. I.Control generates a framework which, in conjunction with the high fidelity models, advanced planning, and sensing components, allows for robust execution of dynamic manipulation tasks.

This integrated paradigm, I.AM. , brings robots to an unprecedented level of manipulation abilities. By incorporating this new technology in existing robots, I.AM. enables shorter cycle time – estimated at 10% – for applications requiring dynamic manipulation in logistics. I.AM. will speed up the take-up and deployment in this domain by validating its progress in three realistic scenarios: a bin-to-belt application demonstrating object tossing (TOSS), a bin-to-bin application object fast boxing (BOX), and a depalletizing scenario demonstrating object grabbing (GRAB).

1.2 Purpose of the deliverable

Deliverable D5.1 is a document summarizing (i) the software background available in the consortium at the start of the project, (ii) an initial plan for the I.AM. software architecture (comprising, in particular, modeling, simulation, learning, planning, sensing, control, communication, hardware aspects), and (iii) the consortium achievements up to M6.

This deliverable paves the way to ensuring the reaching of milestones

- MS1 “Scenario Specification and interfacing architecture agreed” (Smart Robotics)
- MS2 “Software Policy agreed and shared repository put in place” (Algoryx) both at M12 (Dec 2020),

as well as

- M7 “Software integration and numerical testing” (Smart Robotics) at M18 (Jun 2021).

A final update of the I.AM. software integration effort and final I.AM. software architecture will be provided in deliverable D5.8 “I.AM. Software Integration Policy”, at M36 (Dec 2022).

1.3 Intended audience

The dissemination level of D5.1 is “public” (PU) – meant for members of the Consortium, including Commission Services, and the general public.

2 Software background

2.1 Modeling

Regarding dynamic modeling, the coordinator TU/e brought into the consortium its experience in non-smooth mechanics using complementary system formulation and time stepping integrators. Prior the start of the I.AM. project, a 3D rigid body model of a falling rigid box impacting a planar surface comprising normal and tangential (spatial isotropic Coulomb-type) impact laws was already available. This model was developed as a building block for visual object tracking with impact, and it is currently implemented in MATLAB producing results as seen in Fig. 1. This model has direct relevance for the I.AM. project's validation scenario TOSS.

The partner Algoryx is also heavily involved in dynamic modeling and simulation. Since 2007, Algoryx develops practical high-performance numerical methods and delivers commercial software implementation for nonsmooth models The latter based on the same scientific literature and modeling approach, such as [Glo13]), as the coordinator TU/e.

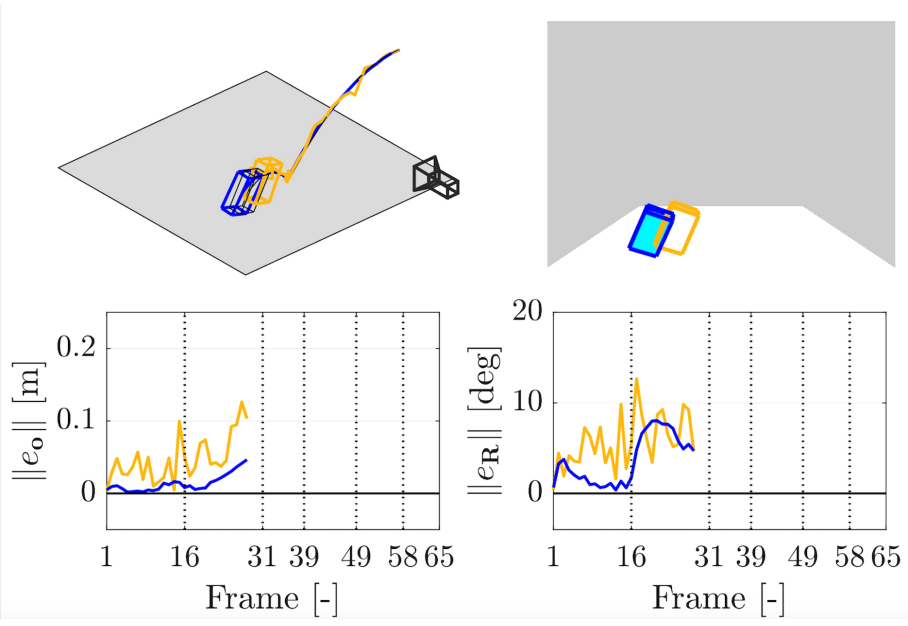

Figure 1: This is a snapshot of the existing TU/e software for visual box tracking with impacts. Comparison of state-of-the-art visual tracking in yellow with no impact model, against developed geometric unscented particle filter based on impact model in blue. The plots show 3D scene with box trajectory trace and impact surface (top left), camera view (top right), reconstruction position error (bottom left) reconstruction orientation error (bottom right) [Jon20]

2.2 Learning

Prior to the I.AM project, EPFL has developed in [AKB19] a Dynamical systems based approach for motion and force generation applicable for single as well as multi-arm robotic systems. The framework exploits the instant re-planning ability of Dynamical Systems (DS) to achieve robustness to perturbation such as sudden displacement of the object before and once contact established. While this this framework was originally designed to make smooth contact with the object prior applying forces, as shown in Fig. 2, it successfully achieves non-smooth contact in a fast object grabbing and lifting task with two KUKA LWR 4+ robotic arms. Thus, while leveraging robustness of DS, the developed approach can induce impact at grabbing time. The system is coupled with learning procedures to learn the nominal dynamics to approach the objects [MFB18] and the contact forces [Ama+20] at run time to compensate for inaccuracies. A ROS packages containing the software implementation (written in `C++`) of the framework has been developed [Ama20], [Mir18] and it is relevant for I.AM. GRAB Scenario.

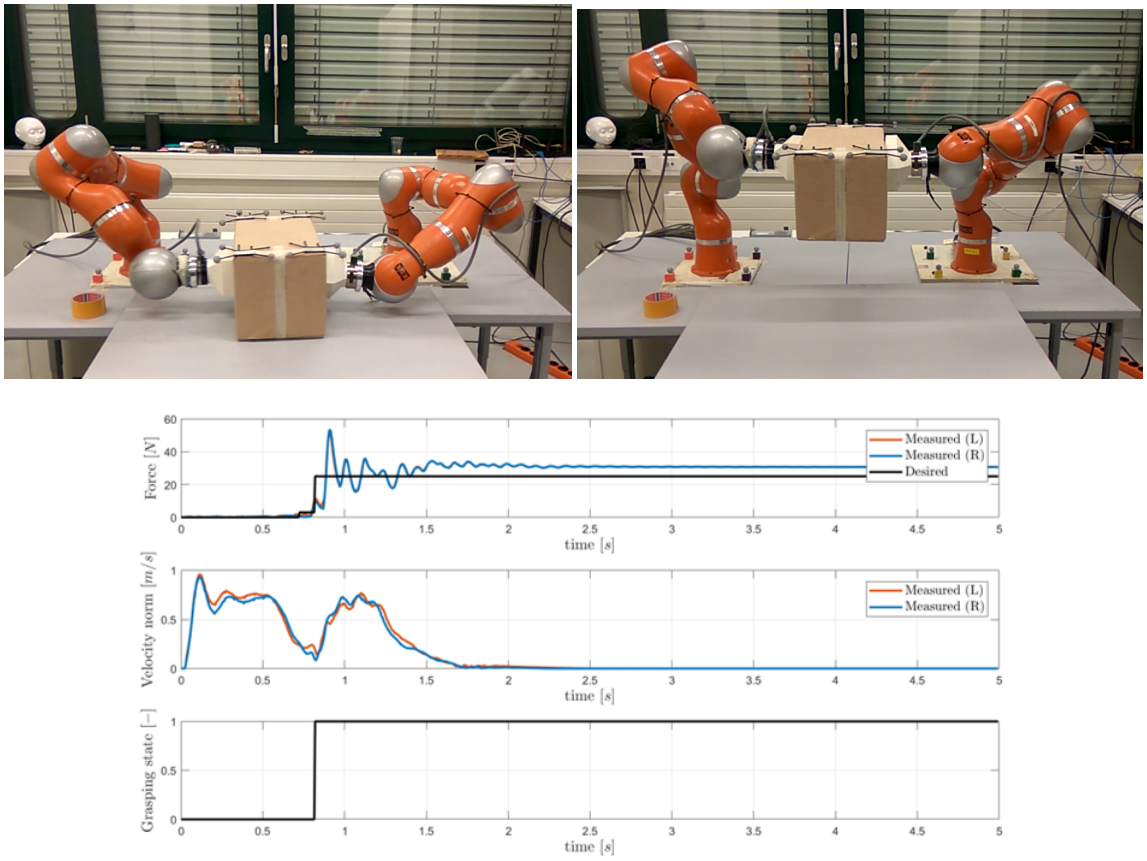

Figure 2: DS-based motion and forces generation in contact for fast object grabbing and lifting. (a): grabbing with impact (left) and object lifting (right). (b) illustration of end-effectors force and velocity profiles during the task. The end-effectors move with high velocity toward the box, decelerate and contact the box with non-zero velocity and quickly accelerate to lift the box.

In the same vein, inspired by [SFB16], we developed another DS-based motion and forces coordination approach, where two robotic arms coordinate their motion by synchronizing with a virtual object that converges towards the real object. This allows coordinated reach-to-grasp motion of static as well as

moving object. Moreover, the generation of grasping and manipulation forces exploits Quadratic Programming (QP) to enforce explicitly the contact constraints, namely, unilateral normal forces, tangent forces within friction cone, stability through moments, and complementary condition between normal force and distance to contact. While this approach was validated on a humanoid robot, it can be easily ported on robotic manipulators adopted within the consortium. Additionally, the emphasis put on the robust coordination between the robotic arms is particularly important in the context of I.AM. project as it prevents premature contact – an impact – of one robot with the object, which might result in undesirable change of the object's pose just before grabbing.

Prior to and since the start the project, EPFL has been developing an anticipatory postural controller for humanoid robots. The objective of this scheme is to allow a robot to take anticipatory actions in the form of postural adjustments such as shifting its center of mass, taking a step in order to mitigate the effects of an upcoming balance perturbation when it occurs. The proposed approach complements classical feedback or reactive-based balance control approaches with a feed-forward controller to allow proactive actions. Thus, given an encoding or previously learned model of the task to perform, in the form of motion and/or forces, the controller predicts the induced balance perturbation and its postural consequences on the robot. And accordingly, it can generate postural adjustments to maintain the robot balance. This concept is being validated in simulation, but it is yet to be tested on a real robot. However, the preliminary results suggest that the proactive behavior introduced by the anticipatory controller improves the robustness of the task execution. It allows the robot to remain stable in situations where classical balance control could not maintain the robot stability without compromising the task. Beyond robot posture, EPFL now seeks to extend anticipatory behavior to the impedance of the robot in preparation of an impact or any other interaction with the environment. In that respect, EPFL will develop strategies to modulate or shape the impedance of the robot according to the task at hand.

Regarding the software implementation of dynamical systems in general, EPFL has developed over the years several packages and toolboxes to learn and test dynamical systems. For instance, DS-OPT [Fig20] is a Matlab toolbox that includes several techniques for the estimation of Globally Asymptotically Stable Dynamical systems from demonstrations. The DS motion generation package [LAS20], which is a ROS-nodified version of DS motion generators. This package supports (i) analytically parametrized DS for simple motions such as linear motions, point-to-point oscillatory motions and swiping in shown in [KB19], (ii) Non-linear DS learned from demonstrations using either the SEDS parameterization approach [KB11] or the LPV-DS parametrization approach [FB18]. The package provides for testing a number of pre-learned model such as curvilinear motion with different targets, non-linear, non-monotic motions used in a variety of tasks e.g., sink motion, via-point motion, CShape motion. A DS-based control architecture using such algorithms is shown in Fig. 3.

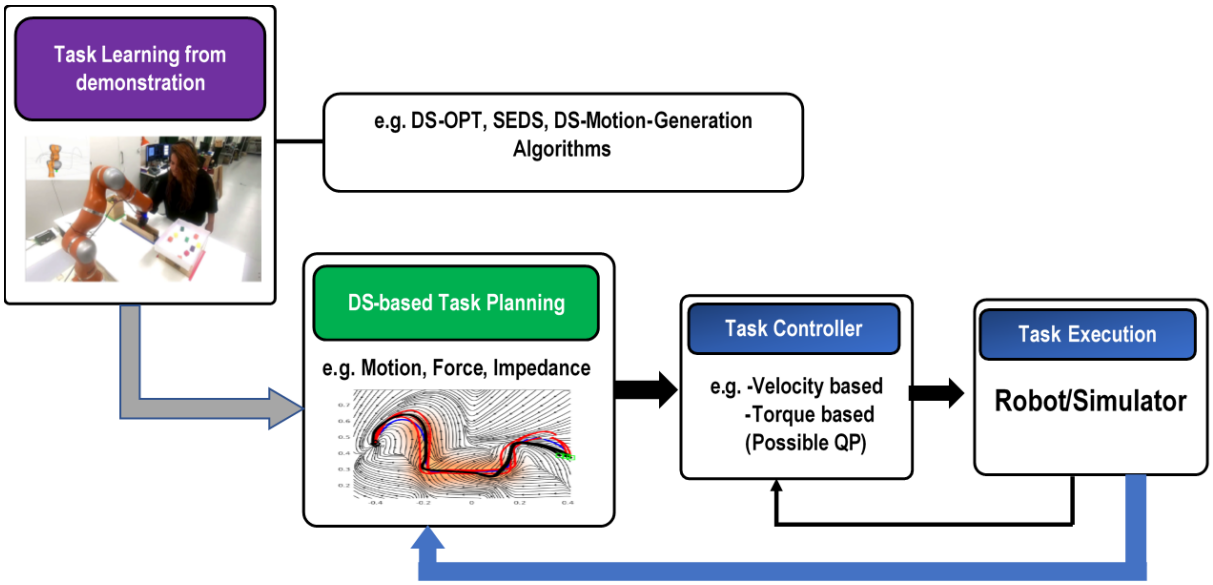

Figure 3: The DS-based task control architecture.

2.3 Sensing

The work flow of the sensing part WP3 is described by the Collision Event Pipeline [HLA17] illustrated in Fig. 4. For the first three steps, detection, isolation and identification, commonly a momentum-based observer [LM05] is used. This is already builtin in several lightweight robot controllers as proprietary functionality. Another more recent approach for these steps is the observer-extended direct method [BKH20]. For this method additional external hardware, like IMU sensors, and software for gathering the measurements from these sensors is needed. Both parts are available as early stage prototypes, whereas the software part is written in standalone Matlab functions, which can be used by project participants. For the other steps, mathematical methods and concepts are available, e.g., for classification [GOH15], and for reaction [Had+08]. However, these parts are not available within I.AM.

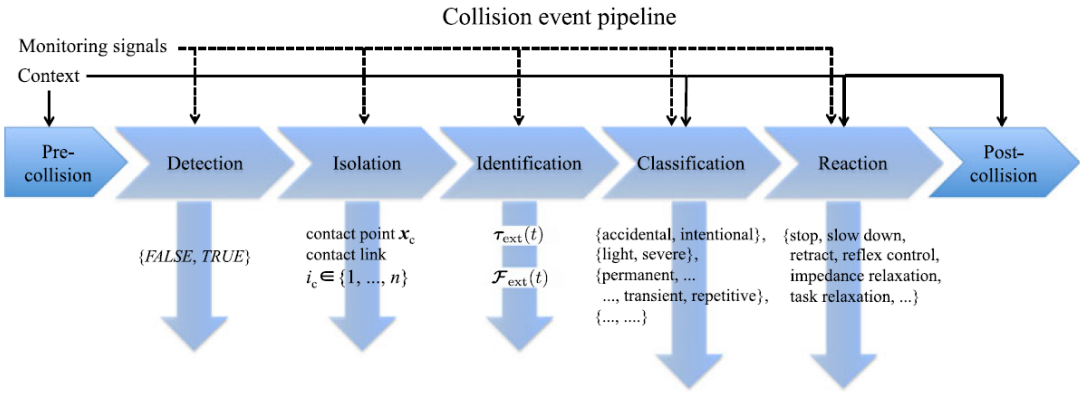

Figure 4: The depicted Collision Event Pipeline is used in this project as a framework for the collision handling and given in [HLA17].

2.4 Control

The control approach that will be promoted in I.AM. consists in formulating robotic objectives in terms of goals to achieve optimally in the task-space as an quadratic program (QP), an approach which is brought in into the I.AM. consortium by the partner CNRS [Bou+19]. Each task is defined as a sensory error function to be reduced as best as possible under constraints such as joint limits and undesired foreseeable collisions. If it happens that not all tasks can be realized perfectly; they are said to be conflicting. In this case, a compromise between tasks is needed. It is therefore necessary to define a hierarchy of priorities. There are three main approaches: (i) soft hierarchy is achieved by weighting tasks giving them relative importance; (ii) strict hierarchy will resolve for the highest priority task first. Then, the next, lower priority task is resolved within what remains feasible without compromising any performance of the higher priority task, etc. Finally, (iii) combines both approaches, and is the most common in practice.

The idea behind this whole-body controller is that a task is a function f of sensory inputs e, any sensor can be considered, $f(e)$ is to be minimized under a set of state constraints, equalities and inequality. The framework aims at building complex controllers from simple task templates that form instructions, and a hierarchical finite-state machine for branching. According to CNRS partner's view, each task is a "motion primitive".

Figure 5 illustrates the main components of the task specification and control architecture used by CNRS for the control of several robots as complex as humanoids. This framework architecture, called `mc_rtc`, is already implemented on several other robots such as those of the SoftBank Robotics family such as Pepper and Nao, the HRP humanoid family including HRP-2, HRP-2Kai, HRP-4, and more recently the HRP-5P, in addition to Sawyer, KuKa arms. Recently, in the context of I.AM. , `mc_rtc` was successfully ported to control the Panda robot.

The Partner CNRS committed to make all the robots of I.AM. – Panda, UR and others if any – powered by `mc_rtc`. The overall architecture is structured into three main components:

1. Low-level and high-performance C++ libraries meant for experts who are familiar with both robotics and the intrinsics of these libraries;

2. A unified controller interface `mc_rtc`: the control framework, meant to facilitate the development of controllers and the integration of new robots, simulation software and robot hardware interfaces;

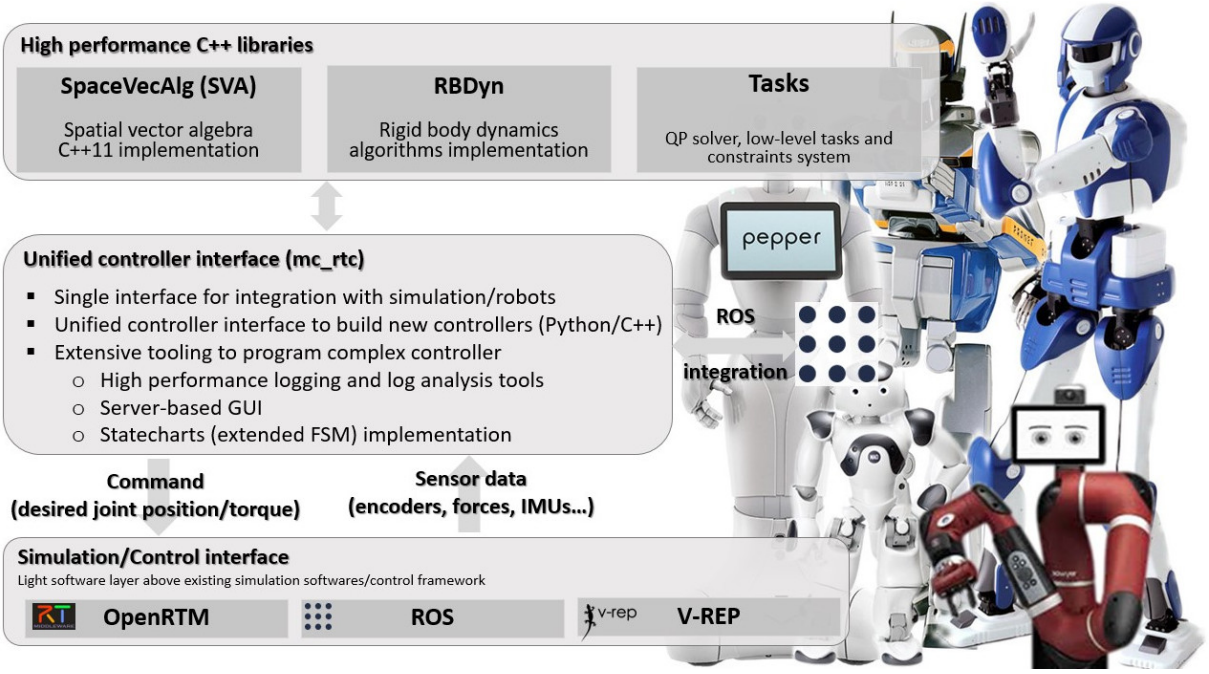

Figure 5: The `mc_rtc` control architecture framework.

3. Simulation/control interfaces which are simple "glue-layers" between the control framework and a simulation software, such as vRep or Choreonoid, or robot-hardware interface. Later, in section 4 we discuss its extension to AGX Dynamics.

The `mc_rtc` framework is written in C++, but allows writing any robot controller in either C++ or `Python`. The low-level libraries are mainly concerned with the mathematical and numerical aspects of the control, i.e., computing all the required matrix and vector quantities correctly in a timely manner, and setting up and solving optimization problems. The `mc_rtc` framework brings simpler interfaces, simpler semantics –which are task-centric instead of model-centric– and a range of tools to support the development of new controllers, such as new mission instances. This framework allows experts and non-experts to build complex and maintainable robotic applications.

Recently `mc_rtc` was successfully used in achieving the EU H2020 COMANOID project demonstrator. In this project a humanoid robot was demonstrated in-situ at the Saint-Nazaire aircraft site, where it achieved complex operations inside a scale 1:1 airplane mockup [Khe+19].

Within I.AM. so far, recent results [SWK18; WK19] were used to extend the task-space control approach to impact-awareness. Indeed, instead of explicitly designing a dedicated controller to handle task-aware impacts, we integrate impact and continuous dynamic models as well as constraints consistently as part of our `mc_rtc` controller. The core idea is to perform a one-step-ahead prediction in every control cycle based on the impact model. That is to say, given imminent, intended or expected impact, we assume that it will happen in the next iteration. As a result, the controller becomes *aware* of impact-induced velocity jumps and restricts the robot motion to meet impact with (possibly maximum) velocities that can handle the upcoming impact safely, i.e., within the pre-defined hardware resilience and task-dictated acceptable bounds. By this approach, the resulting robot motion is robust to uncertainties in impact time and location.

2.5 Simulation

Simulation requires numerous components to be orchestrated. There are computational kernels for controllers, physical motion, graphics rendering, communication and virtual sensors. And these are synchronized by a simulation master. Others components include configuration files, authoring tools, and data storage.

There are available solutions for all these components at different level of maturity, some free, some not. There is however nothing yet for impact-aware control, the raison d'être of the I.AM. project. This also applies computational dynamics libraries – the physics engine as some say – with regards to impact laws.

There are also choices for physics simulation but as each library takes a different path from models, numerical methods, heuristics and approximations, as well as software implementation towards a satisfied end-users who get to focus on their own problems. There is a broad spectrum in terms of speed, fidelity, accuracy, and stability. AGX Dynamics is not unique in with regards to the list of features needed in the present project, namely multibody dynamics and frictional contacts the very minimum. However, nearly all libraries which handle contacts and friction at high speed use iterative solvers which leave very large residual error. AGX is based on a direct factorization and pivoting solver.

However, considering what was written in Sections 2.1 and 2.4, it is closer to the spirit of this project, and is expected to deliver in speed, fidelity (good models), and accuracy. This is because it is a synthesis [Lac07] of discrete time variational methods [MW01], nonsmooth mechanics including impacts [Glo13; LAG09], and specially tailored high performance linear algebra [LLS12]. Being designed for fixed step, and with special attention to all nonsmooth aspects, it couples perfectly with robot controllers which also operate at fixed step.

It is robust as it has been used in thousands of training simulators with scenarios that can last for weeks at a time in the maritime sector for instance. It is now being adopted by engineers moving away from Finite Element Method (FEM) packages as these do not deliver the speed needed, and the error difference between the results are small enough.

As for assembling of all the components into a system simulation, there are several integrated solutions which have been used extensively and successfully in the past by various members of the consortium, and have received good reviews in the robotics community. These include

- Gazebo [Fou20], which has an end-of-life set to 2025,
- Ignition [Rob20b] (its successor),
- CoppeliaSim [Rob20a], and
- Choreonoid [Nak20],

among many more. These solutions support connections to control algorithms via a variety of communication protocols, and have a large number of features. They also provide multibody dynamics in the end-user application via several different libraries through a plugin API, and basic rendering and helper tools for building simple virtual environments. Configuration management is handled via the Scene Description Format (SDF)[1] which is fairly limited in scope.

But there are practical issues with all of them for the present project. The plugin mechanisms involve a "meta" API for each components, viz, sensors, vision, and physics, as this connects existing library to GUI and simulation elements. At least for physics simulation, these APIs are woefully inadequate as no effort

[1] http://sdformat.org/spec

was made to make an ontology which maps to all existing libraries. This resulted in the lowest common denominator. For AGX Dynamics, that would lead to eliminating the possibility of simulating flexible links, conveyor belts, drivelines, and probably suction cups. Extending this API leads to branching the original software. After that point, the possibility of merging back is slim, which is not good for dissemination of the results of this project. In addition, the labor cost involved in integrating AGX Dynamics is high, not relevant to this project, and presents little commercial reward. AGX has a programmatic interface accessible from python, and declarative one in the works. This should be sufficient to support simple applications for the time being.

Outside of the aforementioned integrated tools, "game engine" companies such as Epic Games and Unity 3D have been so successful that they are releasing their simulation environment for free, with full source code in the case of Epic, only charging royalties for published end products. The momentum here is far bigger than what the Open Robotics Foundation can deliver on the simulation front and rendering front. There are big investments[2], direct support from Nvidia for robotics applications[3][4], and with connections to Alphabet for deep learning algorithms, entry into cloud computing. From January 2019 until June 2020 alone, a search on the combined databases of Ex Libris discovery yielded more than 700 articles combined for robotics research using either Unity 3D or Unreal. Around 500 for the first, around 200 for the second. The same keywords give 250 hits for "gazebo", and only one for CoppeliaSim.

And the consequence of all this investment and popularity is the emergence of a fast growing ecosystem in which one can mix and match components. And these "game engine" companies actively encourage this. As concrete and direct example, Epic Games awarded two 100,000$ grants to Algoryx to write a plugin as part of their MegaGrant program [Gam20]. A great deal of what is needed for robotics simulation is already there with respect to virtual sensors, connection with ROS, interfaces with ML and AI, batch job management on GPGPU clusters, and of course, several libraries for physics simulation. And all this without defining meta APIs which can be very wrong.

In conclusion, the Unity 3D and Unreal platforms are rapidly gaining adoption by the robotics community. From Algoryx' perspective, these two have been the obvious target for some time already. From I.AM. 's perspective, this is a path to broad dissemination of the results.

The integration of AGX Dynamics with Unity 3D and Unreal Engine is funded by customers.

Taken together, what this means is that the majority of the effort from Algoryx within I.AM. is to improve on models – the suction cup for instance –, fidelity, speed, and connectivity.

2.6 Human-Robot interaction and safety

Relevant to task T5.5 and deliverable D5.6 "Human safety in impact-aware manipulation", the TUM partner has developed the Safe Motion Unit (SMU)[Had+12] concept, which is currently implemented as a ROS node, making it possible to use, e.g., Gazebo for simulations.

The SMU ROS node requires the kinematic and dynamic parameter of the used robot. Points of interest and their impact on safety are specified prior to execution. Furthermore, the position of a human in the vicinity of the robot needs to be monitored and published, so that the SMU node can subscribe to this ROS topic. As output, the SMU node provides as a ROS service a safe velocity limit for the most critical point of interest as well as the end-effector of the robot. This safe velocity limit is determined based on the inputs and a human biomechanics database included in the ROS node (at present, the database is currently not completed for all human body parts).

[2] https://www.unrealengine.com/en-US/blog/grishin-robotics-aims-to-accelerate-the-future-with-epic-games-unreal

[3] https://docs.nvidia.com/isaac/isaac_sim/plugins/robot_builder/index.html, jun 2019

[4] https://docs.nvidia.com/isaac/isaac/doc/simulation/unity3d.html May 29, 2020

As such, the SMU with ROS is compatible with any simulation environment which supports virtual sensors to publish the position of a simulated human. Simulations are used for testing but not directly for design.

3 Integration policy and plan

For a coherent software integration plan, the consortium started from the I.AM. work package and task interrelation and interdependencies diagram, shown in Fig. 6.

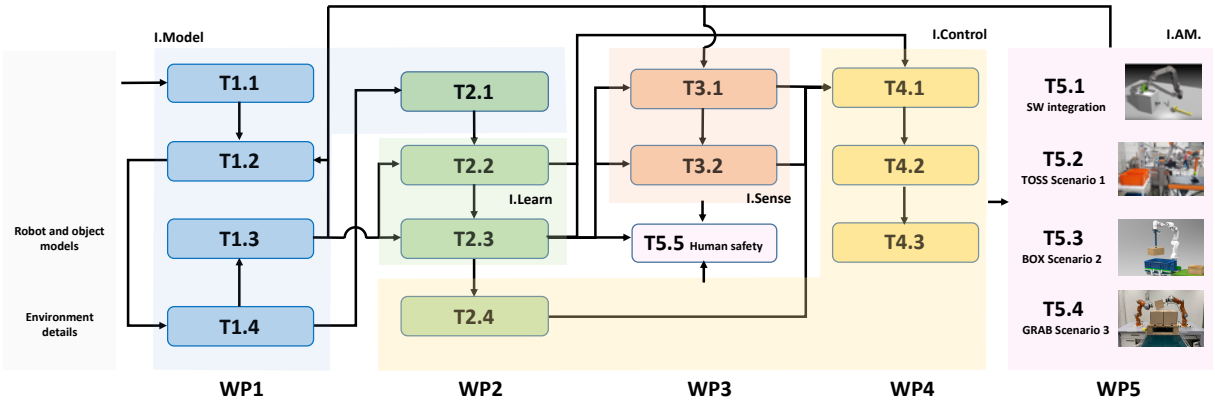

Figure 6: I.AM. work package and task interrelation and interdependencies diagram. There is almost a one-to-one correspondence between the five challenges and corresponding project objectives, namely, I.Model, I.Learn, I.Sense, I.Control, and I.AM.

From this diagram, we have identified and listed all the bricks of software that are planed in each task or a group of tasks, as defined in each WP. Consequently, we thought about gathering, when this is possible, these bricks into a software frameworks. The latter indicate the flow of information or knowledge, from which a global integration architecture is possible.

3.1 Integration architecture

The diagram in Fig. 6 is mapped into the integration architecture sketched in Fig. 7.

The modeling and knowledge generated from I.Model consists of mainly standalone software and code that will be integrated on-demand in almost all the existing software bricks/modules/frameworks of I.AM.For example, contact models can serve the purpose of `mc_rtc` in rewriting the impact-aware tasks and also in AGX to improve the simulation of contacts. Data knowledge can also be used in I.Learn and planning. This is the reasons why I.Model covers all modules. The understanding of how impacts propagates in the structure be used in the observers and the control.

The integration of `mc_rtc` with the results from WP2 is straightforward. The dynamical system planning and part of the control strategy will be integrated as a dedicated module that specify the task objective and even the task automatically. EPFL partner already started integrating task-space constrained QP control with the dynamical system (DS) control approach that will pave the way for impact-aware learning and control integration. From the control part `mc_rtc` can easily be enhanced using `Tasks` and the `mc_rtc::fsm` (branching) from the software developed by EPFL partner. This is work is currently starting.

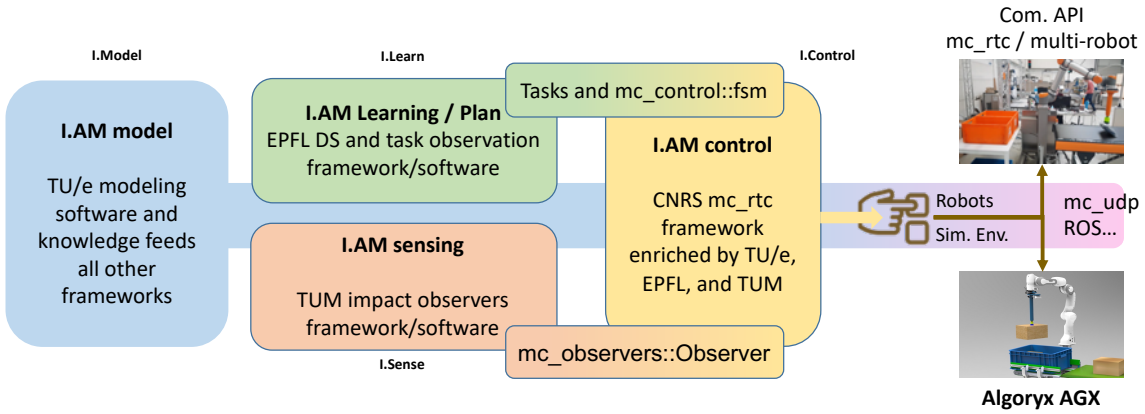

Figure 7: I.AM. integration architecture.

In order for `mc_rtc` to be enriched with observers developed by TUM partner, CNRS partner already prepared an observer pipeline as follows. An observer is an on-line estimation of given parameters by means of models and eventually some of the existing sensors' data and/or some of the existing observers or estimators. Hence, an observer runs a sequence of methods to update the observed parameters from available sensors and previous observer results. An update could be optional in few cases; e.g., log ground truth from a simulator that is not used in the controller. An observer is configured through the controller configuration (default at `mc_rtc` level) as follows:

```
# List of observers to run
RunObserver: [Encoder, BodySensor, KinematicInertial]
# List of observers used to update realRobots() from
UpdateObservers: [Encoder, KinematicInertial]
```

For implementing an observer the following steps are to be made:

1. Write an observer plugin for `mc_rtc`;
2. Inherits from `mc_observers::Observer`;
3. Implements the following methods[5]

```
void reset(MCController); //Reset the observer's state
bool run(MCController);  //Run the estimation but no update
void updateRobot(MCController, Robots); //Update robots
```

The integration of `mc_rtc` with the new simulation environment developed for the I.AM. project will follow the same philosophy as our current integration with v-Rep, Gazebo and Choreonoid simulation framework. The latter already embed the AGX Dynamics library for simulating the physics. In section 4, we highlight the choices made for integrating `mc_rtc` and AGX.

Coupling between `mc_rtc` and a simulation library or a robot can be done via a number of protocols, e.g., UDP, ROS [Rob20d] or OpenRTM [AIS20]. The UDP version is working with AGX, and the other protocols are in development for this library. This type of message passing architecture makes it very

[5]More details in: `https://jrl-umi3218.github.io/mc_rtc/tutorials/recipes/observers.html`

easy to integrate and requires little configuration. The whole-body controller `mc_rtc` already powered the control of a large number of robots and has a well defined policy. In fact, `mc_rtc` assumes the low-level control of each robot to be well tuned and optimized, therefore it acts mainly at task space level. However, shall the low-level control be open to parameters tuning (gains, change of the control law...) `mc_rtc` can benefit from such an opening. For instance, past work prior to I.AM. allowed to optimize gains of the low-level PD controller as part of the QP control framework by considering the PD gains as decision variables.

Models of robots used within the consortium are available as URDF [Rob20c] files, and the same can be done for conveyor belts, although these need additional parameterization.

Algoryx is developing a declarative format using YAML[6] which is both human readable and writable by contrast with XML. What this declarative format is good for is to collect what the simulation needs to start, without having to write a script, and without having to make a GUI. A proposal has been submitted to ITEA3 for further development and integration or interoperability with the Universal Scene Description (USD)[7] originally developed by Pixar. USD is rapidly evolving to support all aspects of simulations including deep learning and cloud execution. The momentum behind this effort is significant because of the coupling with Unity and Unreal, and goes far beyond the SDF. This should have significant outreach.

Indeed, the current configuration is not too pleasing (see code below).

```
python AGX_URDF_Environment.py \
    -urdf_file ./urdf_models/panda_suctioncup.urdf \
    -urdf_package ./urdf_models/franka_ros-kinetic-devel_suctioncup \
    -robot_name panda_tool -pos -0.25,3.45,0.55 -rpy 0,-0,2.35 \
    -urdf_file ./urdf_models/panda_suctioncup.urdf \
    -urdf_package ./urdf_models/franka_ros-kinetic-devel_suctioncup \
    -robot_name panda_2 -pos 0,-0.45,0.45 -rpy 0,-0,-0.75 \
    -urdf_file ./urdf_models/box.urdf -pos 0.3,1.85,0.860898 -rpy 0,-0,0 \
    -urdf_file ./urdf_models/box.urdf -pos -0.552,1.1,0.860898 -rpy 0,-0,0 \
    -urdf_file ./urdf_models/box.urdf -pos 0.3,0.35,0.860898 -rpy 0,-0,0 \
    -urdf_file ./urdf_models/box.urdf -pos -0.552,2.6,0.860898 -rpy 0,-0,0 \
    -conveyor demo -conveyor_speed 0.3 \
    -pickable_objects box -suction_cup true --timeStep 0.005 \

agx=$!
# make sure the server has time to start
sleep 4
MCUDPControl -s > mclog 2>&1 &
mc=$!
wait $agx
kill -15 $mc
```

Yet the results are promising: Fig. 10 shows the result of a tossing motion realized by using CNRS whole-body controller where physics was simulated employing AGX physics library. In Fig. 10, there are three impacts following the simulated TOSS of parcel on a conveyor belt, one with four simultaneous contacts. The time step was 1/200 seconds, here so these collisions happen in rapid succession.

[6] yaml.org

[7] https://graphics.pixar.com/usd/docs/index.html

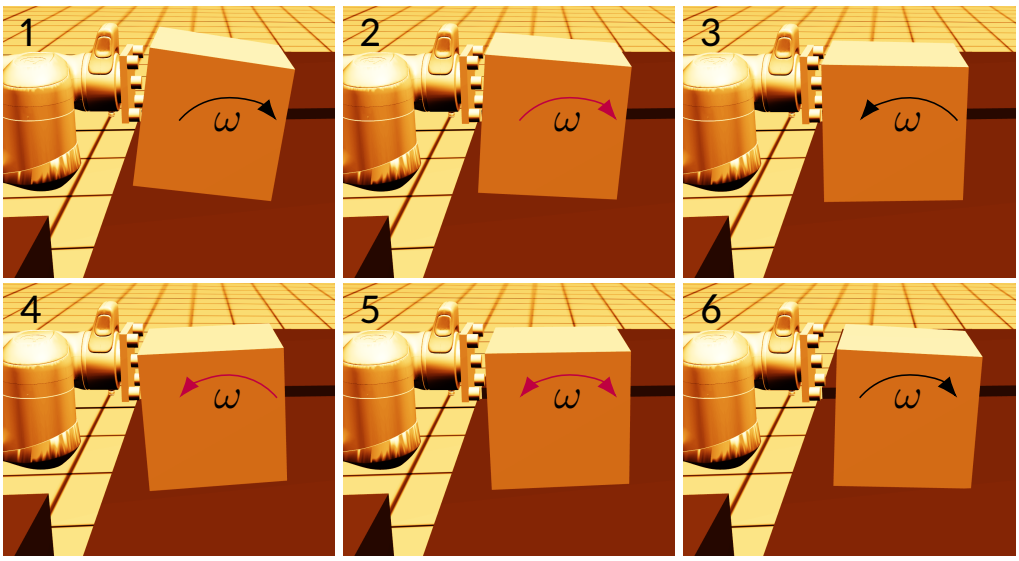

Figure 8: Snapshots of a simulation with AGX Dynamics and CNRS whole-body controller `mc_rtc`.

3.2 Software development policy

The `mc_rtc` control framework, a task-space optimization universal robot controller and all its bricks and documentation is Open Source licensed under the BSD 2-Clause terms[8]. The core C++ libraries with a heavy focus on performance that achieve the most fundamental operations (spatial vector algebra, rigid body dynamics and setting up and solving QP) have already been released[9]. However the QP solver implementations that are used (QLD and LSSOL) are released by 3rd-parties under different licenses (public domain and commercial respectively). This software is currently maintained with a dedicated support for all the I.AM. partners.

As for Algoryx, AGX Dynamics is proprietary but licenses have been issued to all members of the consortium and for the entire duration of the project, with extensions as deemed useful. The code written within the project that is not related to internal components – such as code for the friction model – will be licensed under the BSD 2-Clause as `mc_rtc`.

At present, the code for connecting `mc_rtc` and AGX Dynamics and the simple application are stored in a git archive host on Algoryx' gitlab. This archive is both readable and writable for all members of the consortium.

3.3 Communication with robots

Communication with robots is abstracted away from the programmer in the `mc_rtc` framework. Therefore, to switch from the simulation to an actual robot, the user simply needs to change the interface that is used to initiate and run the controller. The communication of the robot can use different interfaces depending on the robotic platform, e.g., using the same UDP protocol as with the simulation or a dedicated application using the vendor library[10].

[8] `https://opensource.org/licenses/BSD-2-Clause`
[9] `https://jrl-umi3218.github.io/mc_rtc/`
[10] Prototype interface for libfranka and `mc_rtc` `https://github.com/gergondet/mc_franka`

4 What has been achieved

4.1 Impact model parameter identification software

The TU/e has developed various software routines, currently in MATLAB, to automatically identify friction laws parameters directly from box-conveyor impact data. This includes normal and tangential coefficient of restitutions, as well as dry friction coefficients. The software makes use of the simulation code implementing the box-plane impact model described as background in the Section 2.1, which is able to simulate the impact and possible "tumbling" of a box-shaped parcel on a conveyor belt. This is relevant in particular for I.AM. 's validation scenario TOSS. Robot, parcel, and gripper motions are recorded via a mocap system at 360FPS. It was found that parcel-conveyor impact duration lasts between two or three samples, and TU/e has obtained a preliminary confirmation that nonsmooth mechanics formulation is a valid approximation for the parcel-conveyor forward dynamics. In the meantime, Algoryx has augmented the impact model of their simulation library to include tangential restitution which the TU/e found necessary to match the experimental results collected via the mocap system and this has been validated against existing theory [Glo13]. The data collected from Vanderlande and TU/e will give Algoryx (as well as other physics engine companies) the opportunity of validating everything contact related in the AGX Dynamics library and make sure it matches the experiments within tight margins for the applications of interest.

The impact dataset is stored in the Hierarchical Data Format 5 (HDF5) [Gro19]. This format is binary and hierarchical. It supports many different types of data, and since annotations can be stored with the data, the files are self-descriptive, as illustrated in Fig. 9. The HDF5 impact dataset be used in the coming months also for the validation of AGX Dynamics in the TOSS scenario. The impact database will be available starting from end of September 2020 as part of I.AM. project's deliverable D1.1, and it will be kept up-to-date with new impact/release measurements related to the TOSS, BOX and GRAB validation scenarios used within I.AM.

Algoryx and TU/e, with support of Vanderlande and Smart Robotics, are now turning their attention to the modeling, parameter identification, and simulation of the suction cups and conveyor belts and the development of related software routines for the TOSS and BOX scenarios. As for simulation of suction cups needed for these scenarios, there appears to be nothing for direct use in this context. There are articles [Ani+15; CWH19] on the topic aimed controls, but other than one article on analytic models [Liu+06] or the finite element method for aimed at design optimization simulations [NH09], there is no indication that this is yet available. At present time, suction is simulated in AGX Dynamics by making contact points adhesive *after* contacts are detected.

4.2 Interface between AGX dynamics and CNRS's whole-body control software

At the start of the project, Algoryx planned to use Choreonoid [Nak20] as a simulation framework because it nominally supports all the required components, namely, AGX Dynamics, `mc_rtc`, and URDF robot models. Since Chorenoid was developed by the National Institute of Advanced Industrial Science and Technology (AIST) Japan, with whom the CNRS has established an international collaborative laboratory, a joint research unit, the CNRS-AIST Joint Robotics Laboratory (JRL), also a partner of I.AM, this was the perfect candidate. It appears now that the main developer, Dr Nakaoka, created a start-up with only one employee and already several engagements with Japanese industries. After several discussions with Dr Nakaoka and the CNRS and AIST researchers, it was deemed too risky to engage the I.AM. consortium with Chorenoid as our simulation framework. The software is open source, but without support, there

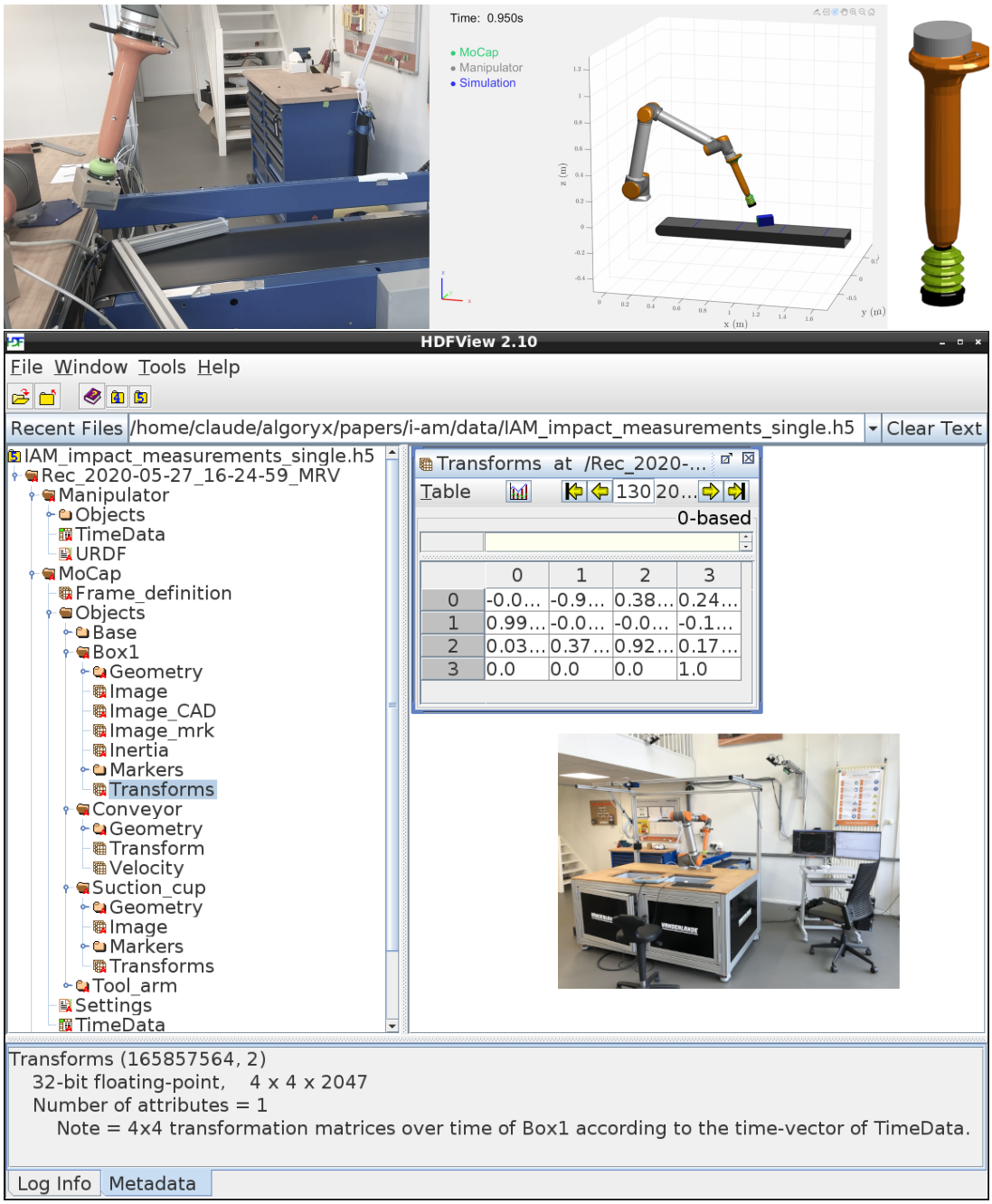

Figure 9: On the top row is the experimental setup available on TU/e campus, in Vanderlande's Innovation Lab, making use of a mocap system (Optitrack 17W cameras, 360FPS). The setup uses reflective markers on the box, conveyor, robot, and gripper (top) and the visualizer for comparing recorded data with nonsmooth model simulations center, and the bellows suction cup right. On the bottom row is a screenshot of an HDF5 viewer showing the structure of the file and the type of data it can contain.

is no future. Therefore, both Algoryx and CNRS agreed to move forward with a different plan leading to a simple prototype application with all the aforementioned capabilities, and a virtual environment

needed for the TOSS scenario. A URDF parser was implemented for Algoryx' own, humble visualization program, and as soon as the Panda robot could be simulated, simple conveyor belt and suction cup models were added. The performance at 500Hz was sufficient to keep synchronization with the CNRS `mc_rtc`. This is soon to be distributed to the consortium partners. Support for ROS communication as needed by Smart Robotics is in development and nearly finished.

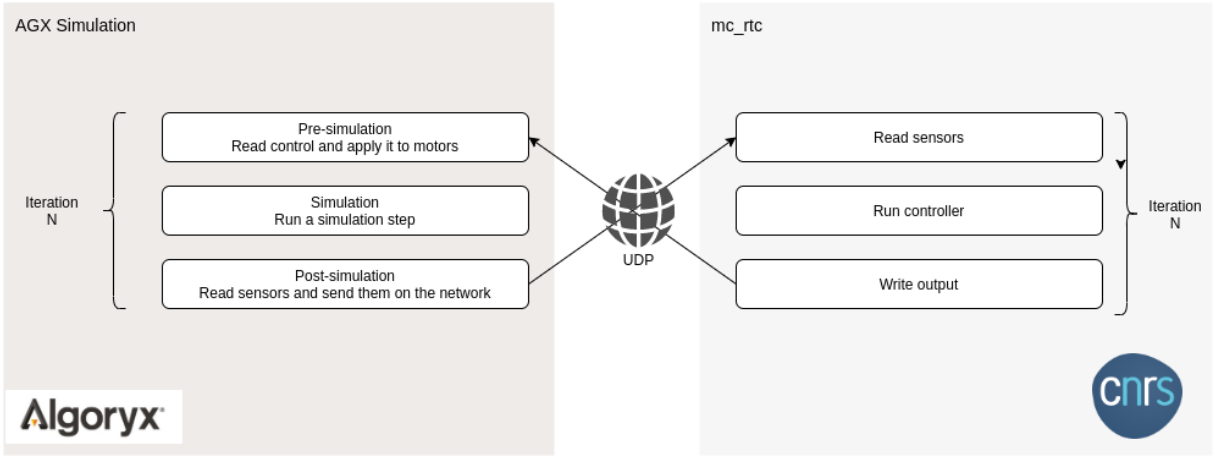

Figure 10: Integration scheme between `mc_rtc` and AGX Dynamics

Figure 10 shows an overview of the current integration between `mc_rtc` and AGX dynamics, or, more accurately, the `agxViewer` application which runs the `mc_rtc` UDP server which is configured to implement the commands defined in the `mc_rtc` messages. This links to the `mc_rtc` client which implements the tasks. This is simple indeed but can go quite far yet. The protocol was established by CNRS to communicate with robots that do not have sufficiently performant hardware to compute the control in real-time. This is provided in the `mc_udp` package[11]. This integration scheme shall evolve as AGX configuration format materializes. Currently, the simulation's and controller's initial state must be specified in a shell script as described above.

[11] `https://github.com/jrl-umi3218/mc_udp`

Conclusion

This deliverable D5.1 provides a preliminary overview of existing software and methods within the consortium that will be used to create the I.AM. software. It also provides insight and describes the software integration policy and plan, with a description of the achieved results within the first six months of the project in this context. We believe that within only six months, the objectives of the deliverables are much beyond what was expected (rough scheme).

There is still three years and six months this project duration until M48 (that is December 2023); this deliverable will be thoroughly and continuously updated to be wrapped in its finale version by M36 (that is December 2022) into the deliverable D5.8 "I.AM. software integration policy (update D5.1)" as well as in the internal reports of the milestones MS1 "Scenarios specification and interfacing architecture agreed" and MS2 "Software policy agreed and shared repository put in place" both planned at M12 (that is December 2020).

All in all, all the I.AM. partners are aware of the importance of an early effort toward integration and that integration is the matter of all.

REFERENCES

[AIS20] AIST. *OpenRTM*. `https://www.openrtm.org/openrtm`. June 2020.

[Ama20] Amanhoud. *ds-based-contact-tasks*. `https://github.com/epfl-lasa/ds_based_contact_tasks`. June 2020.

[AKB19] Walid Amanhoud, Mahdi Khoramshahi, and Aude Billard. "A Dynamical System Approach to Motion and Force Generation in Contact Tasks". In: *Proceedings of Robotics: Science and Systems*. June 2019.

[Ama+20] Walid Amanhoud et al. "Force Adaptation in Contact Tasks with Dynamical Systems". In: *International Conference on Robotics and Automation*. 2020.

[Ani+15] Yunafiátul Aniroh et al. "Dynamics and Control of a Suction-Type Wall-Climbing Robot". In: *IFAC-PapersOnLine* 48.1 (2015). 8th Vienna International Conferenceon Mathematical Modelling, pp. 902–903.

[BKH20] Seyed Ali Baradaran Birjandi, Johannes Kuehn, and Sami Haddadin. "Observer-Extended Direct Method for Collision Monitoring in Robot Manipulators Using Proprioception and IMU Sensing". In: *IEEE Robotics and Automation Letters* 5.2 (2020), pp. 954–961.

[Bou+19] Karim Bouyarmane et al. "Quadratic Programming for Multirobot and Task-Space Force Control". In: *IEEE Transactions on Robotics* 35.1 (Feb. 2019), pp. 64–77.

[CWH19] Hao Chen, Weiwei Wan, and Kensuke Harada. *Combined Task and Motion Planning for a Dual-arm Robot to Use a Suction Cup Tool*. 2019. arXiv: `1909.00192 [cs.RO]`.

[Fig20] Nadia Figueroa. *ds-opt*. `https://github.com/nbfigueroa/ds-opt`. June 2020.

[FB18] Nadia Figueroa and Aude Billard. "A Physically-Consistent Bayesian Non-Parametric Mixture Model for Dynamical System Learning." In: *CoRL*. 2018, pp. 927–946.

[Fou20] Open Source Robotics Foundation. *Gazebo*. `http://gazebosim.org`. June 2020.

[Gam20] Epic Games. *Epic MegaGrants*. `https://www.unrealengine.com/en-US/megagrants`. 2020.

[Glo13] Christoph Glocker. “Energetic consistency conditions for standard impacts, Part I. Newton-type inequality impact laws and Kane’s example”. English. In: *Multibody System Dynamics* 29.1 (2013), pp. 77–117.

[GOH15] Saskia Golz, Christian Osendorfer, and Sami Haddadin. “Using tactile sensation for learning contact knowledge: Discriminate collision from physical interaction”. In: *2015 IEEE International Conference on Robotics and Automation (ICRA 2015)*. IEEE, 2015, pp. 3788–3794.

[Gro19] The HDF5 Group. *HDF5 version1.10.15*. `https://www.hdfgroup.org`. May 2019.

[HLA17] Sami Haddadin, Alessandro de Luca, and Alin Albu-Schäffer. “Robot Collisions: A Survey on Detection, Isolation, and Identification”. In: *IEEE Transactions on Robotics* 33.6 (2017), pp. 1292–1312.

[Had+08] Sami Haddadin et al. “Collision Detection and Reaction: A Contribution to Safe Physical Human-Robot Interaction”. In: *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2008, pp. 3356–3363.

[Had+12] Sami Haddadin et al. “On making robots understand safety: Embedding injury knowledge into control”. In: *The International Journal of Robotics Research* 31.13 (2012), pp. 1578–1602.

[Jon20] Maarten Jongeneel. “Model-Based Visual Object Tracking with Collision Models”. report DC2020.024. MA thesis. Eindhoven University of Technology (TU/e), Dept. of Mechanical Engineering, 2020.

[KB11] S Mohammad Khansari-Zadeh and Aude Billard. “Learning stable nonlinear dynamical systems with gaussian mixture models”. In: *IEEE Transactions on Robotics* 27.5 (2011), pp. 943–957.

[Khe+19] A. Kheddar et al. “Humanoid Robots in Aircraft Manufacturing: The Airbus Use Cases”. In: *IEEE Robotics Automation Magazine* 26.4 (Dec. 2019), pp. 30–45.

[KB19] Mahdi Khoramshahi and Aude Billard. “A dynamical system approach to task-adaptation in physical human–robot interaction”. In: *Autonomous Robots* 43.4 (2019), pp. 927–946.

[Lac07] Claude Lacoursière. “Ghosts and Machines: Regularized Variational Methods for Interactive Simulations of Multibodies with Dry Frictional Contacts”. PhD thesis. Dept. of Computing Science, Umeå University, June 2007.

[LLS12] Claude Lacoursière, Mattias Linde, and Olof Sabelström. “Direct Sparse Factorization of Blocked Saddle Point Matrices”. In: *PARA 2010, Part II*. Vol. 7134. LNCS. Springer, 2012, pp. 324–335.

[LAS20] LASA. *ds-motion-generator*. `https://github.com/epfl-lasa/ds_motion_generator`. June 2020.

[LAG09] R. I. Leine, U. Aeberhard, and C. Glocker. “Hamilton’s Principle as Variational Inequality for Mechanical Systems with Impact”. In: *Journal of Nonlinear Science* 19.6 (Dec. 2009), pp. 633–664.

[Liu+06] Jihong Liu et al. “Analytical modelling of suction cups used for window-cleaning robots”. In: *Vacuum* 80.6 (2006), pp. 593–598.

[LM05] Alessandro de Luca and R. Mattone. “Sensorless Robot Collision Detection and Hybrid Force/-Motion Control”. In: *2005 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2005, pp. 999–1004.

[MW01] J. E. Marsden and M. West. “Discrete mechanics and variational integrators”. In: *Acta Numer.* 10 (2001), pp. 357–514.

[Mir18] Mirrazavi. *A Unified Framework for Coordinated Multi-Arm Motion Planning*. `https://nbfigueroa.github.io/multi-arm-coordination/`. June 2018.

[MFB18] Seyed Sina Mirrazavi Salehian, Nadia Figueroa, and Aude Billard. “A unified framework for coordinated multi-arm motion planning”. In: *The International Journal of Robotics Research* 37.10 (2018), pp. 1205–1232.

[Nak20] Shin’ichiro Nakaoka. *Choreonoid*. `https://choreonoid.org`. June 2020.

[NH09] Fr Novotny and M. Horak. “Computer Modelling of Suction Cups used for Window Cleaning Robot and Automatic Handling of Glass Sheets”. In: *MM Science Journal* 2009.2 (2009), pp. 113–118.

[Rob20a] Coppelia Robotics. *CoppeliaSim*. `https://www.coppeliarobotics.com`. June 2020.

[Rob20b] Open Robotics. *Ignition*. `https://ignitionrobotics.org`. June 2020.

[Rob20c] Open Robotics. *Unified Robot Description Format (URDF)*. `wiki.ros.org/urdf`. June 2020.

[Rob20d] Open Robtics. *Robot Operating System (ROS)*. `https://www.ros.org`. June 2020.

[SFB16] Seyed Sina Mirrazavi Salehian, Nadia Figueroa, and Aude Billard. “Coordinated multi-arm motion planning: Reaching for moving objects in the face of uncertainty”. In: *Proceedings of Robotics: Science and Systems*. June 2016.

[SWK18] Y. Su, Y. Wang, and A. Kheddar. “Sample-Efficient Learning of Soft Task Priorities Through Bayesian Optimization”. In: *2018 IEEE-RAS 18th International Conference on Humanoid Robots (Humanoids)*. Nov. 2018, pp. 1–6.

[WK19] Yuquan Wang and Abderrahmane Kheddar. “Impact-friendly robust control design with task-space quadratic optimization”. In: *Robotics: Science and Systems*. 2019.