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

Scenario 1 (TOSS) report

Dissemination level	Public (PU)
Work package	WP5
Deliverable number	D5.3
Version	v1.0
Submission date	30-12-2022
Due date	31-12-2022

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**
Saeed ABDOLSHAH	TUM	saeed.abdolshah@tum.de
Micheal BOMBILE	EPFL	michael.bombile@epfl.ch
Teun BOSCH	Smart Robotics	tbosch@smart-robotics.nl
Maarten JONGENEEL	TU/e	m.j.jongeneel@tue.nl
Stijn de LOOIJER	Vanderlande	stijn.de.looijer@vanderlande.com
Fredrik NORDFELDTH	Algoryx	fredrik.nordfeldth@algoryx.com
Alexander OLIVA	TU/e	a.a.oliva@tue.nl
Jari van STEEN	TU/e	j.j.v.steen@tue.nl
Ahmed ZERMANE	CNRS	ahmed.zermane@lirmm.fr
Sjouke de ZWART	Smart Robotics	sdezwart@smart-robotics.nl

Peer reviewers		
	Reviewer name	**Date**
Reviewer 1	Claude LACOURSIÈRE	26-11-2022
Reviewer 2	Alessandro MELONE	29-11-2022
Reviewer 3	Alessandro SACCON	8-12-2022
Reviewer 4	Harshit KHURANA	12-12-2022
Reviewer 5	Jos DEN OUDEN	30-12-2022
Reviewer 6	Abderrahmane KHEDDAR	30-12-2022

Legal disclaimer

Executive summary

Robotic automation is mainly applied in the loading, unloading, and infeed processes of a logistic system, where human workers still have a central role given their speed and reliability compared to current available robotic solutions. In fact, currently available robot solutions are often higher in cycle time and less reliable than a human, which is preventing logistic companies from further automating their processes. To fill this gap, a transition from pick-and-place towards pick-and-toss can help to speed up the process and thereby improve the productivity of the robot solution.

To make the robots pick-and-toss instead of pick-and-place, impact-aware tossing was developed. Various components are developed and combined to achieve impact-aware tossing. Several models have been developed to predict the behaviour of an item being tossed. These models include the behaviour of an item being held by a suction cup, the release dynamics when it is being tossed as well as the behaviour of the item when impact with the conveyor occurs. To make use of the models, it is essential to know the inertial parameters of the object that the robot is going to handle. Therefore, a payload identification method was explored to determine the object inertial parameters online. Although additional sensing units such as vision system, IMU, or even a prior knowledge about object may facilitate the payload identification process, we decided to stick to the application of proprioceptive sensors and avoid using exteroceptive sensors that may not be in the scope of the project use cases. A planner was developed that can find a release state when a desired rest pose of the item is given (see also deliverable D2.1). To plan a motion of the robot towards this release state, a dynamical system based controller was developed. This robot-independent second-order task-space dynamical system (DS) has been integrated into the `mc_rtc`[1] framework (a quadratic programming (QP) task-space based control framework). These components are developed, tested, and validated in simulation. This is done by doing real-world experiments and then parameterize the simulation models to be able to repeat the real-world behavior, and then be able to use the validated simulation environment with the same controller that is running with the real robot.

The item-conveyor models, the planner, and the controller have been integrated to do real-world testing on a setup that resembles those used in the current industry. Testing is done on both a Universal Robotics UR10 robot as well as a Franka Emika robot. The benchmark results show the potential of the I.AM. technology, especially for the traysorter scenario and a bin-to-belt system. The sensitivity analysis shows that item parameters do not have to be estimated perfectly in order to toss accurately within a desired window. The results show that by tossing items a 10% decrease in cycle time can be achieved when comparing on the same experimental setup. The tossing scenario is, however, not yet ready in its current state to be used in the industry. This is because in the current tossing algorithm, the item parameters are assumed to be known such that optimization for a toss can be done offline. In the industry, the item variety is large and it is most often not known what item is being handled next. Therefore, parameter identification and tossing optimization will have to be performed online. However, the report shows by means of real experiments that tossing can indeed decrease the cycle time of robotic systems in logistics by at least 10%, showing the potential gain it can generate on the current logistics industry.

[1] https://jrl-umi3218.github.io/mc_rtc/

TABLE OF CONTENTS

1 Introduction **7**
- 1.1 I.AM. project background 7
- 1.2 Purpose of the deliverable 8
- 1.3 Intended audience 8

2 Business value **9**

3 Developed components for tossing **11**
- 3.1 Tossing Robot Architecture 11
- 3.2 Dynamic Holding, Release, and Impact Models 11
  - 3.2.1 Box-suction cup holding 12
  - 3.2.2 Suction cup release dynamics 14
  - 3.2.3 Object-conveyor impact model parameters 14
- 3.3 Payload estimation 15
- 3.4 Tossing planner 16
- 3.5 Tossing controllers 17
  - 3.5.1 DS-based tossing motion generation 17
  - 3.5.2 Robot control through QP robot control (mc_rtc) 19
- 3.6 Simulation environment 19
  - 3.6.1 Simulated components 20
  - 3.6.2 Simulation software and models 20
  - 3.6.3 Simulation runtime 21
  - 3.6.4 Validation procedure 21
  - 3.6.5 Expected user experience 21
  - 3.6.6 Release of GLUE 21

4 Performance testing **22**
- 4.1 Performance evaluation (KPIs) 22
- 4.2 Test plan 25
  - 4.2.1 TOSS algorithm 25
  - 4.2.2 Place algorithm 25
  - 4.2.3 Metrics 25
- 4.3 Test setup 26
- 4.4 Tossing test results 28
- 4.5 Sensitivity analysis 32
- 4.6 Using the UR10 32
- 4.7 Evaluation 33

5 Conclusion **34**

6 REFERENCES **35**

A Infeed scenario tossing **36**

B Results of the sensitivity analysis for tossing **37**

ABBREVIATIONS

Abbreviation	Definition
CO	Confidential
EC	European Commission
SIR	Smart Item Picking Robot
WP	Work Package
pph	packages per hour

1 Introduction

1.1 I.AM. project background

Europe is leading the market of torque-controlled robots. These robots are compliant and 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 that will grow exponentially in the coming years, due to socio-economical drivers such as the booming of e-commerce and scarcity of labor. 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 proximity of impact times, allowing distinguishing between expected and unexpected events;
4. I.Control generates a framework that, in conjunction with the realistic 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 (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 scenario);
- a bin-to-bin application demonstrating object fast boxing (BOX scenario);
- a case de-palletizing scenario demonstrating object grabbing (GRAB scenario).

These scenarios are further abbreviated as the TOSS, BOX, and GRAB scenario.
In this report, we mainly evaluate the development and results of the TOSS scenario. In Section 2, we descibe first the value the TOSS scenario potentially has for various applications in the current industry and what development the current industry is missing. Once the gap in the current technology is outlined, we discuss the components that are developed within the I.AM. framework with respect to the TOSS scenario in Section 3. In Section 3.6, the setup of the simulations is described as well as the results of tossing in simulation. Experiments were done on real robots, which is shown in Section 4, where also performance testing and results are shown. Finally, in Section 5 conclusions are drawn for the results of the TOSS scenario.

1.2 Purpose of the deliverable

This report aims at providing insight on the demonstration of the TOSS scenario as well as an overview of the components developed by the I.AM. consortium over the course of the project. Also, it contains the results of the integration of these components occurred in the period 12-16 September 2022 during a project integration week. The I.AM. partners TU/e, CNRS, EPFL, Smart Robotics, Vanderlande attended physically at the Vanderlande Innovation Lab on TU/e campus in Eindhoven, while the partners TUM and Algoryx supported remotely. The report contains furthermore the corresponding performance testing related to the results of the TOSS scenario.

Figure 1: TU/e, CNRS, EPFL and Smart Robotics researchers working on the integration of components for TOSS scenario.

1.3 Intended audience

The dissemination level of this report is 'public' (PU) – meant for members of the Consortium (including Commission Services) and the general public.

2 Business value

The booming e-commerce market, the increased labor scarcity and the need for more future-proof working environments are driving the demand for robotic automation in logistics. In the past, robots have already proven their value in automating highly repetitive tasks (e.g. in automotive or manufacturing). Nowadays robot solutions are also entering the logistic world. Here they have to deal with less structured environments and a high variation in the products to handle.
Robotic automation is mainly applied in the loading, unloading and infeed processes of a logistic system, where human workers still have a central role given their speed and reliability compared to current available robotic solutions. In fact, currently available robot solutions are often slower and less reliable than a human, which is preventing logistic companies from further automating their processes. To fill this gap, a transition from pick-and-place towards pick-and-toss can help to speed up the process and thereby improve the productivity of the robot solution.
Two use cases are identified for which tossing technology can be utilized to speed up the robotic process; infeed on a crossorter infeed (see Figure 2) and direct infeed on a moving traysorter (see Figure 3).

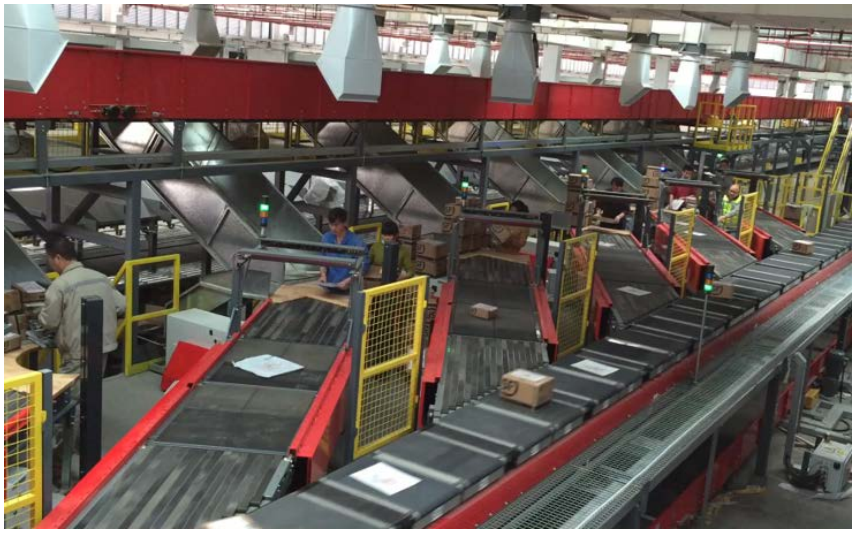


Figure 2: Manual infeed stations that merge onto a crossorter

The crossorter in Figure 2 is one of Vanderlande's parcel loop sorters. It is made of decks with a moving conveyor that is used to sort the parcel towards the right end destination. Parcels are loaded onto an infeed conveyor, from where they are merged onto the crossorter. The parcels should be oriented with their short edge leading and under a 30°angle.
The traysorter shown in Figure 3 is another loop sorter in the portfolio of Vanderlande which can be used for parcel sortation. In this case, Parcels are directly loaded onto a traysorter moving at 1.5 m/s. The parcels should be positioned in a window of the traysorter, but the orientation is not strict: only the barcode should not be facing down so it can be read by barcode scanners in a later stage.

The crossorter infeed case will be used to explain the potential value of pick-and-toss compared to pick-and-place solutions. A reference system is defined that represents a mid-size European parcel customer with the following characteristics:

- System capacity 10.000 pph,
- Capacity human operator 1.500 pph,
- 7 infeed workstations,
- Hourly rate €40/hr,
- Operation 8h/day, 6 days/week, 52 weeks/year.

We assume a robot induct station with the following costs:

- Sales to customer (STC) price €420.000,
- Operational cost €42.000/year.

The operational costs cover the maintenance of the robotic system and license fees.
As benchmark, we take a state-of-the-art industrial robotic induct system that can reach 1500 picks per hour. Important to note is that those systems make use of an industrial robot (e.g., ABB or Fanuc). Those robots can reach much higher accelerations than the cobots (Universal Robots' or Franka Emika's) that are used in the I.AM. project. The industrial robots need to be fenced off to secure human safety, so for research purposes a collaborative robot is easier to work with. It is assumed that the cycle time decrease that we can achieve on a collaborative robot can be transferred to an industrial robot.
So let's consider the benchmark system with 1500 picks per hour. We assume that this system can handle the complete parcel flow that passes the induct station. With the above mentioned parameters, this results in a return of investment of 8 years. In the I.AM project we aim to increase the throughput of pick and place operations by 10%. So applying I.AM technology to the benchmark system with 1500 picks per hour, will lead to a throughput of 1650 picks per hour. This decreases the return of investment significantly to 6.4 years.

Figure 3: Manual traysorter infeed stations

3 Developed components for tossing

In this section, one can find the various components that are developed in order to achieve impact-aware tossing. At first, the architecture is discussed, explaining what components are in the system and how they relate to each other. Secondly, the components are discussed. These components are (a) the dynamic models used for predicting tossing results, (b) sensing of the items (payload estimation), (c) the tossing planner, and (d) the controller tasked to execute the planned tossing motion. Finally, we discuss the simulation that was used to verify the tossing in a simulated environment.

3.1 Tossing Robot Architecture

In this section, the different components of the developed TOSS framework are described. In Figure 4, a graphical overview of this framework is described, showing the core components and their interconnections with other components. Starting on the top left, the *real-life setup* is shown, which describes the physical setup, consisting of one or multiple robots with a suction gripper, objects to be handled, and the tossing environment, including for example the conveyor belt on which the objects will be tossed.
This physical setup is modeled in a *virtual world model*. This includes models of the robot and the attached suction gripper with a flexible suction cup model, as well as models of the different objects being tossed and the environment where the objects land after tossing. To validate this virtual world model, a *model parameter validation* procedure is performed by comparing tosses on the real setup with identical tosses in the virtual world. More information on the models used in the virtual world model and the parameter validation procedure can be found in Section 3.2, while the software aspect using AGX dynamics is explained in Section 3.6.
Using this validated virtual world model, a *planner* is formulated, which currently uses a fully data-driven approach: for a given object, several numerical tossing simulations are performed within a compact set of initial conditions to determine the optimal tossing release state for a given desired rest pose of the object on the environment (e.g., a traysorter deck or a conveyor belt). Details of this planner are given in Section 3.4 and more thoroughly in the deliverable D2.1. Because in an industrial scenario the inertia properties of the object are generally unknown, a *payload estimation* procedure is designed to determine the object's center of mass, weight, and inertia tensor, so that a suitable tossing release pose obtained using the planner can be retrieved, even when the object is unknown. Next to payload estimation, the shape and geometry of the item must also be known for accurate tossing, however, item shape/geometry estimation (by means of a vision system) is already existing in the current industry, so no focus for development has been done within the I.AM. project.
When the item parameters (such as mass, dimension, and inertia) are known, it can be used together with the desired rest pose of the item as input for the planner to find a desired release state. A Quadratic Programming based controller, for short *QP robot controller*, is designed to steer the robot in such a way that the desired release state (pose and velocity) is reached. This controller can be used both for tossing on the real setup, as well as in the virtual world model, allowing for safe preliminary testing and controller optimization without requiring from the start the real setup. The details of the developed QP controller can be found in Section 3.5.

3.2 Dynamic Holding, Release, and Impact Models

This section explains the model used for the tossing scenario. An accurate model is needed to predict the behaviour of a toss. Firstly, the suction cup holding a box as well as the release dynamics of the said

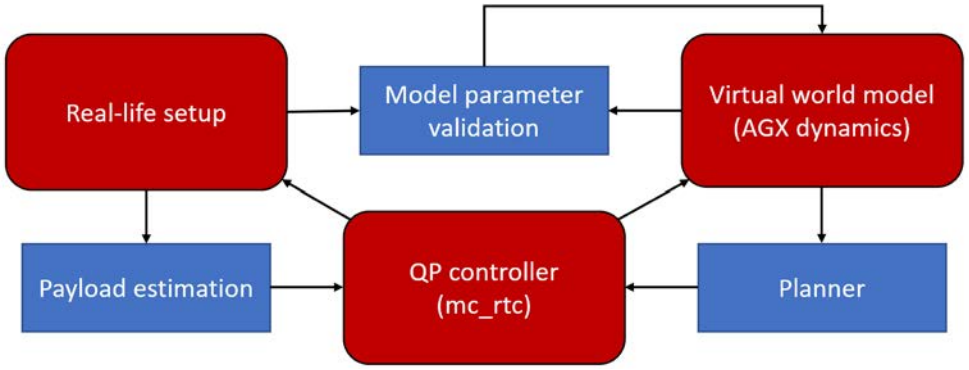


Figure 4: Diagram showing the correlation between different components of the tossing framework

suction cup are described. This is done in Section 3.2.1 and 3.2.2 respectively. In Section 3.2.3 the impact and friction dynamics of the box and the conveyor are detailed.

3.2.1 Box-suction cup holding

The bellows suction cup is a ubiquitous gripper for robotic package picking and placement in logistics. Its compliance allows a robotic manipulator to handle a wide range of packages, whether them being rigid, semi-rigid, or deformable. When dealing with tossing motions that will be generated autonomously by a motion planner, the compliance of the suction cup must be taken into account in order to properly compute the trajectories that will lead to have package landing in the desired location. To forecast the deformation of the suction cup during the holding phase, its dynamic behaviour needs to be modeled, and the parameters describing such a model identified.
In this study, the dynamics of the bellow suction cup gripper are modeled using a model-based approach. The suction cup is assumed to be a massless 6D force-torque coupled elastic element. Based on the surveyed literature, to model a spring wrench that has geometrical meaning, the elastic element has been derived from a spring potential energy function.

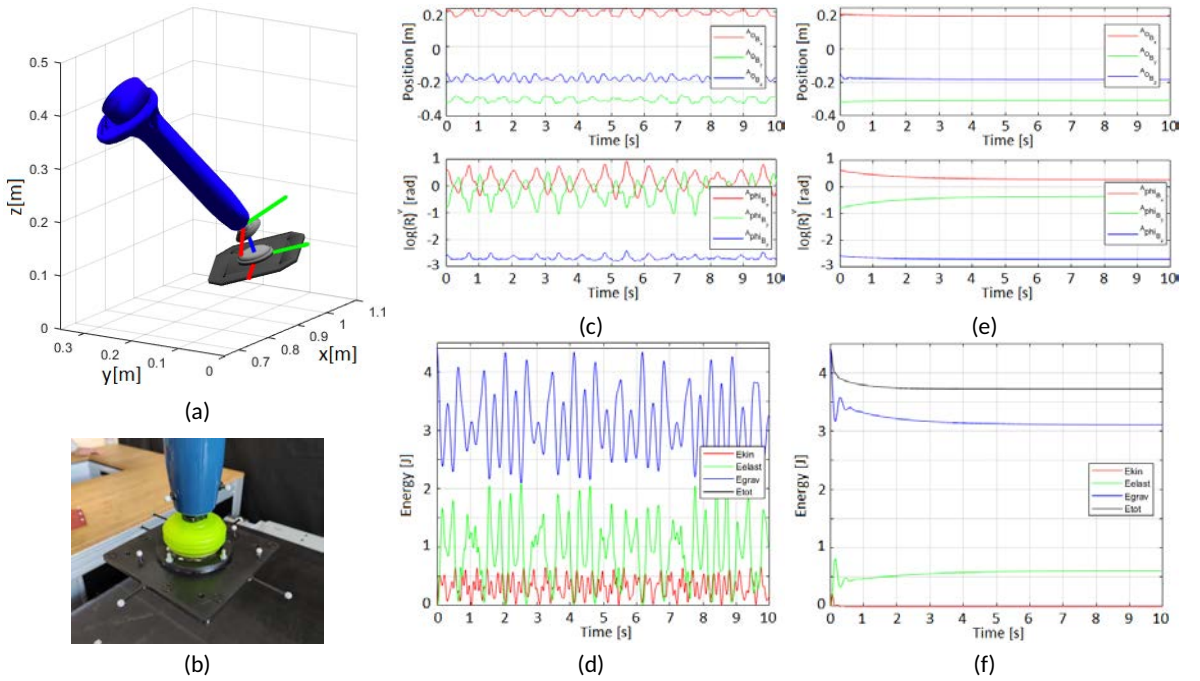
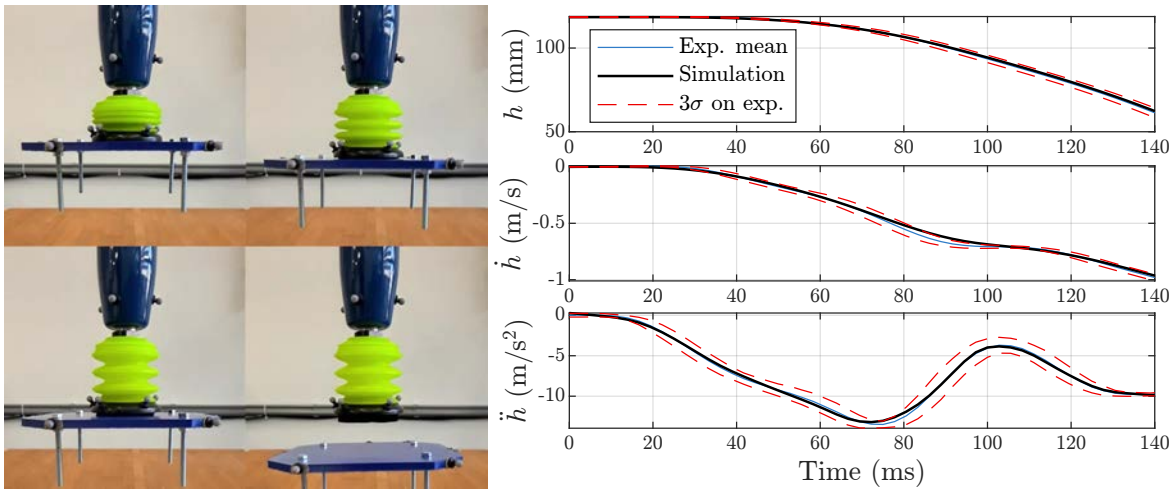


Figure 5: Numerical implementation of the bellow suction cup dynamics model: (a) tool-arm and plate's final configuration after $10[s]$ of numerical simulation in the damped case; (b) VIOPlate and Tool-Arm equipped with reflective passive markers used to record real data for the identification procedure; CoM position and orientation (c) and energy (d) over time in the undamped case; CoM position and orientation (e) and energy (f) over time in the damped case.

To validate the proposed model and its physical consistency, numerical simulations are carried out and the system's energy is evaluated. As reported in Figure 5, starting from the initial tool-arm configuration (Figure 5(a)) and keeping it in that fixed configuration, the suction cup model holding a known payload subjected to gravity is made, causing the payload to bounce indefinitely over time. Figure 5(c) shows the position and orientation of the plate's *Center of Mass*, while its *Potential*, *Kinetic*, *Potential elastic* and *total energy* is shown in Figure 5(d). When also a 6D damping term is modeled, the system dampens its velocity reaching a rest pose, as can be seen from Figure 5(e) while the system's energy is shown in Figure 5(f). As expected, the *total energy* is constant in the lossless case, while decreases in the presence of dissipation.
Currently, the stiffness and damping parameters of a real bellow suction cup are being identified using OptiTrack measurements (see Figure 5(b)) and an object of known inertial parameters that can be opportunely modified (Variable Inertia Object) to fully excite the system's dynamics. The identification procedure takes place in two phases: a static one in which only the stiffness of the spring is identified and a dynamic one in which the damping term is determined. Despite the low signal-to-noise ratio (the displacement of the suction cup is quite small ($[0, 1.5][mm]$) compared to the OptiTrack's noise level of $\sim 0.2[mm]$), the preliminary results are promising for experimentally validating the bellow suction cup dynamics model during the holding phase, which will subsequently allow us, once the model will be integrated into the planner, to automatically generate tossing trajectories from the desired impact position of the package.

3.2.2 Suction cup release dynamics

During the release of the object from the suction cup, the suction cup exerts forces on the object that determine its trajectory, and therefore its rest pose. Therefore, a dynamical model should be created that allows to predict the trajectory of an arbitrary (known) object. A first step towards this model is presented in [1], and a typical release experiment is shown in Figure 6a. Once the trigger is given to release the object from the vacuum gripper, pressurized air flows into the suction cup, which makes the suction cup elongate until its full length is reached. In [1], the release dynamics of a plastic plate (with different attached masses) is studied in a one-dimensional setting. A set of experiments is executed where the motion of the objects and the suction cup are recorded via a motion capture system. A supervised learning approach is proposed that shows remarkable prediction capabilities when evaluated on a test set, despite not having any direct information about the suction cup state, see Figure 6b. Achieving an accurate prediction of the 1D force gives good hope that the same approach can be used in the 6D case, which is left for future work. The reader is referred to [1] for further details.



(a) (b)

Figure 6: Snapshots of a slow motion recorded video during a typical vertical dropping experiment (a), and one of the results of the learned models (b). Source: [1]

3.2.3 Object-conveyor impact model parameters

In Algoryx, the impact and friction dynamics are described using Newton's impact law and Coulomb friction law, respectively. Besides information about the mass, inertia, and geometry of the object, these laws require an estimation of so-called contact parameters: the coefficient of restitution (e_N) and the coefficient of friction (μ). To obtain an estimation of these parameters, experiments can be executed where different boxes are tossed on a conveyor. From these experiments, we use the pre-impact state (pose and velocity) of the box as input for simulations, together with certain values of the parameter set $\{\mu, e_N\}$. The post-impact velocity as result of the simulations is then compared to the measured post-impact velocity, resulting in the costs shown in Figure 7. These costs show a clear optimum, which provides us with the parameter values for μ and e_N used in the simulation environment. These results will be presented in a paper for which a pre-print can be found online [2].

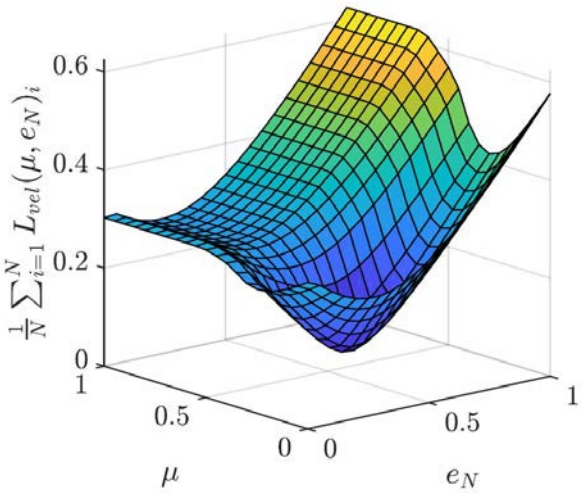


Figure 7: Resulting costs. The resulting costs show a clear optimum for the coefficient of restitution and coefficient of friction.

3.3 Payload estimation

In the tossing scenario, to make sure that the object lands on its specific position it is essential to know the inertial parameters of the object that the robot is going to handle. Therefore, a payload identification method was explored to determine the object inertial parameters online. Although additional sensing units such as vision system, IMU, or even prior knowledge about the object may facilitate the payload identification process, we decided to stick to the application of proprioceptive sensors and avoid using exteroceptive sensors that may not be in the scope of the project use cases.

Let's consider the standard rigid body dynamics model of a robot with n joints. Assuming the robot has grasped an object, we need to estimate the external torques (caused by the object) as well as position, velocity and acceleration of joints to calculate the inertia matrix (The bottom part of Figure 8, (Online identification)). To remove the measurement error and offsets a novel calibration method is used such that its offline data is exploited in online identification approach. we also proposed to design a filter function that is similar to the filtering characteristic of the momentum observer and applied it to all measurements. This generates a set of timely aligned measurements and improves the performance of payload identification.

None of the payload's inertial parameters can be estimated, as long as the robot trajectory does not excite them. An appropriate choice of the robot motion has a significant influence on the accuracy of the identification process. For the use cases in the I.AM. project, we may even need to execute another task (e.g. moving to the releasing point for the tossing scenario), while the identification approach is in process. Therefore, an efficient, but simple trajectory (8 superposed sinusoidal functions) for the payload identification is used and potentially it can be integrated with a task motion. Further details and results of this work can be found in the published paper at 2022 IEEE International Conference on Robotics and Automation (ICRA) [3].

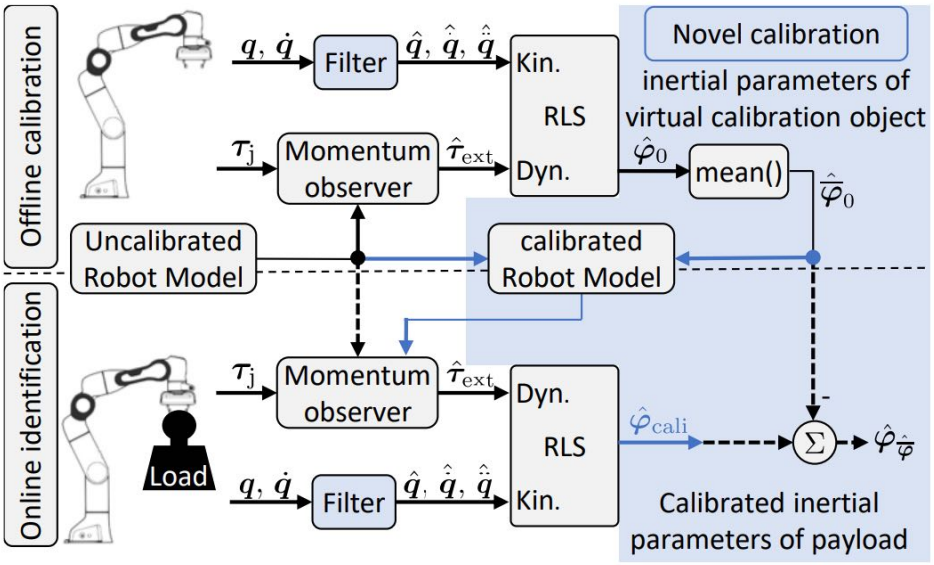


Figure 8: An overview of the proposed payload identification method.

3.4 Tossing planner

In a robotic tossing scenario, the object's trajectory flying in space, impacting, bouncing, tumbling, sliding, and finally landing at a certain rest pose depends on the robot's end-effector pose and end-effector velocity at the moment of release, hereafter referred to as end-effector release configuration[2].
Our goal is to select an effective end-effector configuration for releasing the object given a specific target rest orientation. The term effective here means that we are aiming at a release configuration that is robust with respect to slight variations in the initial conditions (the robot likely fails to drop off the object at the exact release pose with the exact release velocity) and model inaccuracies, and hence ensures the object is landing with the desired surface in contact in a pre-defined user-desired target area.

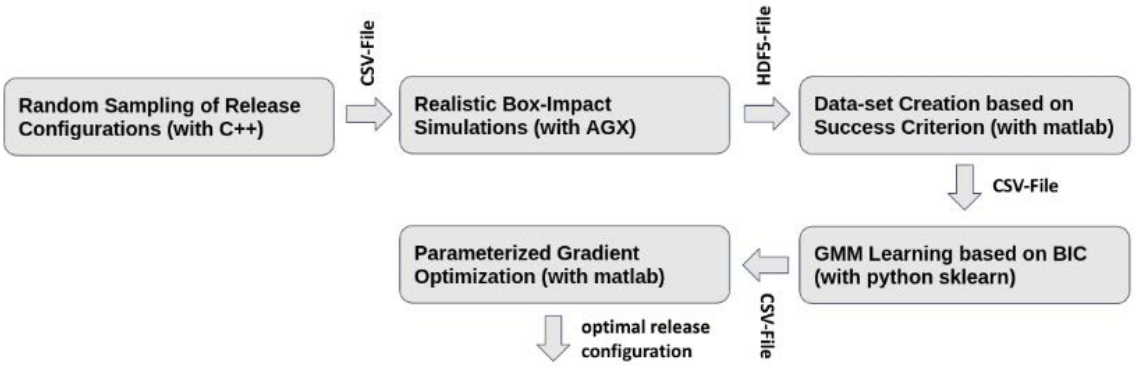


Figure 9: I.AM tossing planner pipeline.

[2]It depends also on the dynamical properties of the object and the environment.

The approach as shown in Figure 9 requires an inverse impact model. The impact dynamics can be accurately simulated forward in time. Due to the highly non-smooth dynamical impact effects (such as friction of two materials against each other) it is however almost impossible to infer an inverse impact model analytically.
Therefore we are utilizing learning techniques that can circumvent this problem. In order to acquire a data set for the given robot manipulator comprising release configurations and corresponding object rest states, we propose to simulate the object's flight and landing trajectories, thereby including also highly non-smooth dynamical impact effects. The output of the planner is an optimal end-effector release state consisting of a pose and velocity, which are collectively denoted as $\boldsymbol{\lambda}^*$.

3.5 Tossing controllers

In order to reach the desired optimal end-effector release configuration $\boldsymbol{\lambda}^*$, we need to plan (and execute with the robot) a tossing motion. The tossing task being mainly defined in task-space imposes motion constraints to both translation and rotation that must be satisfied for its successful execution. Moreover, the planning problem is also constrained by the robot's dynamics and its hardware features. Adopting motion generation inspired by the method for softly catching an object in flight [4] allows meeting $\boldsymbol{\lambda}^*$ with the end-effector as a via-point. Thus, to address both the tossing motion generation and its execution, a second-order[3] task-space dynamical system (DS) has been created, which has been integrated into the mc_rtc framework [5]. The mc_rtc controller can be used to control both real-life and simulated robots. Particularly, utilizing the newly developed AGX interface, simulations for tossing the object can be executed with the DS-based mc_rtc controller and AGX dynamics. Further details on the DS-based mc_rtc tossing controller are provided next.

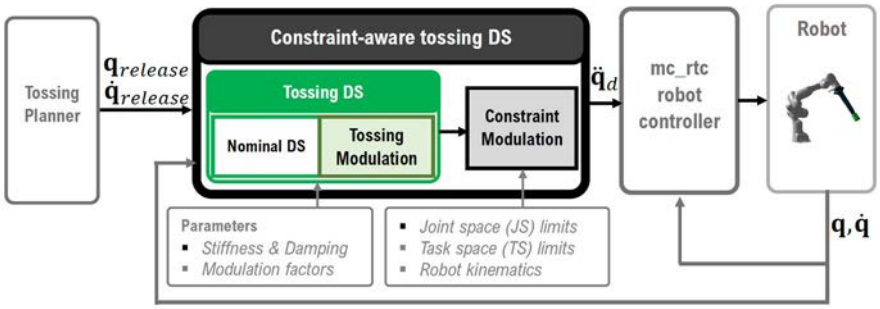


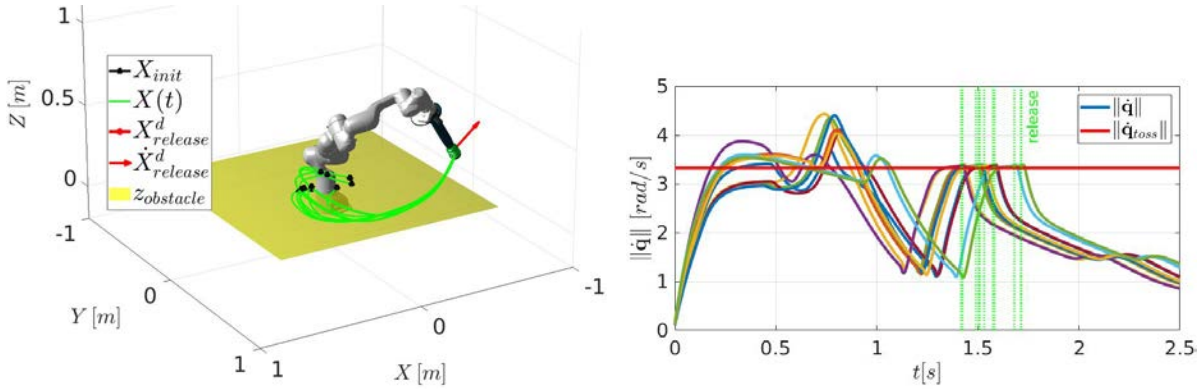
Figure 10: Overview of the constraint-aware tossing DS and its integration within the system.

3.5.1 DS-based tossing motion generation

The desired tossing release state, from a control perspective, represents an intermediate or transitory state defined in terms of release position and velocity that must be satisfied simultaneously at the release instant. The robot can only pass through and not settle to such a state. Besides reaching the desired release state, an additional requirement for the robot is to avoid task space collision (mainly with the supporting table) when executing the desired tossing motion.
To address the tossing motion generation problem with the aforementioned requirements, unlike classical approaches based on motion planning, we adopted a solution based on dynamical systems (DS).

[3] Second-order DS here refers to a DS that outputs desired acceleration for each position and velocity state of the robot.

Thus, the desired tossing motion is encoded into a time-invariant and autonomous differential equation that maps any robot's state to its desired evolution. Such an approach offers fast and time-independent replanning abilities and robustness to perturbations.
More precisely, to fulfill the requirements of the tossing task, we proposed a modulated DS approach where state-dependent modulation functions locally shape the motion of the robot such that it passes through the desired release states. An overview of the proposed DS is shown in Figure 10. The main idea is to generate motion towards an attractor located near the desired release position, and when in its vicinity (within the modulation region), reshape the robot's motion to also align it with the desired velocity while moving towards the desired release position. Thus, the proposed method simplifies the motion generation to reach a $2N$-dimensional release state problem into an $(N+1)$-dimensional attraction problem. This is achieved by projecting the motion, thanks to a task-related orthonormal basis, into a space where the problem reduces to following a moving attractor in one dimension (along the desired velocity) and to convergence to fixed attractors in the remaining orthogonal dimensions, as shown in [6] in the case of dual-arm tossing.
Moreover, to satisfy the task-space constraints of the tossing task in both translation and rotation, we proposed to generate the motion in joint-space, where regardless of the desired release state in task-space, the corresponding release state belongs to $\mathbb{R}^{2\mathrm{N_{dof}}}$, with $\mathrm{N_{dof}}$ is the number of degrees-of-freedom of the robot. This allows a straightforward application of the proposed DS approach. Furthermore, to ensure that the motion generated by the DS satisfies joint limits in position, velocity, acceleration, and task-space limits (collision avoidance), we introduced into the DS the awareness of such constraints through motion modulation. Thus, using the robot's kinematics, we transformed and embedded the joint limit and task space polytopic constraints into motion modulation matrices for the DS. Figure 11 illustrates ten simulated tossing trajectories generated using the proposed DS.



(a) (b)

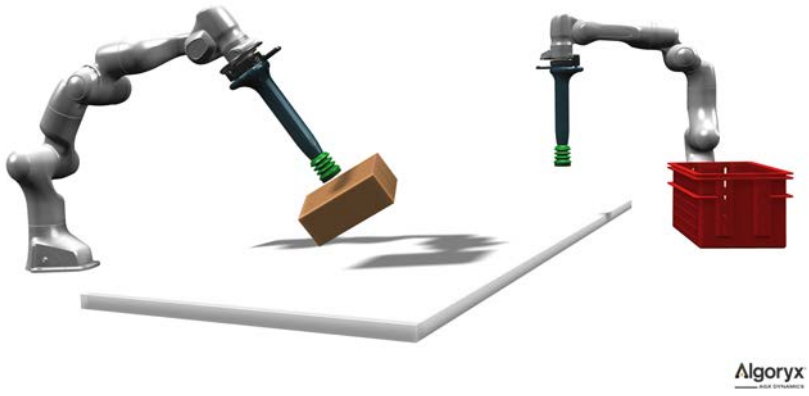
Figure 11: Example of simulated ten trajectories generated using the proposed constraint-aware joint-space tossing DS: (a) 3D trajectories (green), initial positions (black) and release position (red). The yellow plan represents the task-space obstacle that needs to be avoided; (b) corresponding joint velocity norms. The norm of the desired tossing release velocity is in red, whereas the velocity norms of the motions generated by the DS are in other colors. At the release times (green vertical lines), the norms of the joint velocity vectors equal the norm of the desired joint space release velocity.

3.5.2 Robot control through QP robot control (mc_rtc)

To control the robots, both in the simulation environment as well as in real-life, the mc_rtc [5] control framework is used. The latter comprises a set of tools among which a task-space controller formulated as a quadratic programming (QP), solving a QP problem at all times separated by a finite time-step to find the optimal control inputs at that time. The QP control approach allows to execute tasks as well as possible using a weighted cost function for the different objectives, while adhering to constraints, such as joint limit constraints and (self-)collision avoidance constraints. Coupling of the mc_rtc framework with the Algoryx physics engine, by means of the GLUE framework developed during the I.AM. project, is further detailed in the I.AM. project deliverable D5.8.
The control framework makes use of the finite state machine (FSM) feature in mc_rtc to execute the entire toss cycle, which is divided in different states. Through the first states, the box is grabbed from the tote and brought to a position above the tote. Once this is finished, the tossing state is entered. In this state, the output of the planner (λ^*), which contains the desired end-effector pose and velocity at the moment of release, is transformed to a desired release joint position and velocity through inverse kinematics. This desired joint state at the moment of release is used in the DS-based tossing motion generation algorithm, further detailed in Section 3.5.1, to determine the desired joint velocity at every time-step. When this desired velocity is tracked, the motion followed by the robot is given by the desired path shown in Figure 11. Hence, the QP in the tossing state enforces a task, minimizing the error between the desired and the measured joint velocity. When the release pose is nearly reached, suction of the gripper is disabled and the package is tossed, after which the robots moves back to a position over the tote.
One thing to note is that the inverse kinematics procedure used within the control framework will not always have a unique joint state corresponding to the desired Cartesian end effector state. This can cause uncertainty, as minor changes to the desired end effector release state can result in entirely different joint release states, and hence trajectories to reach the desired final pose. To minimize this unwanted effect, the position and velocity of a single joint is given as additional input in the inverse kinematics procedure, which leaves only a finite amount of joint states corresponding to the desired end effector pose. Of these finite poses, the infeasible states, resulting in self collisions or joint limit violations are filtered out. However, for certain poses, multiple feasible solutions are still found, resulting in the described unwanted effect to be present to a smaller extent.

3.6 Simulation environment

Having a virtual representation of the world gives a safe, scalable and distributable test environment for robotics. When it comes to manipulating objects with robots, it is clearly an advantage if the robot is accessible to anyone, and the scenarios are configurable so that synthetic data of a large test domain, spanning any imaginable scenario, can be generated for validation and development of control algorithms. For the virtual environment to be trusted, all the included components must be validated. By doing real-world experiments and then parameterize the simulation models to be able to repeat the real-world behavior, and then be able to use the validated simulation environment with the same controller that is running with the real robot. To reach a good level of accuracy the simulation needs to also include the dynamics of flexible degrees of freedom both for joints and suction cups at the least. For a control system developer, a simulated toss needs to generate realistic sensor data including typical noise from measurement devices to have any value.
To be able to record real-world behavior, advanced 3D motion capture systems have been set up at TU/e



(a)

Figure 12: Snapshot from the simulation environment

for the TOSS and BOX scenarios, as shown in Figure 14.
Within the consortium we have developed a concept for enabling validation and high fidelity real-time simulation, including flexible suction grippers, with a control system in the loop.

3.6.1 Simulated components

For the simulated Franka Emika robot, illustrated to the left in Figure 12, we like to repeat the toss operation in simulation. To achieve this, first of all the dynamics of the robot itself need to perform in an accurate manner. This requires the inertia of each link, the joint motor acceleration and stiffness, and any end effector to be parameterized. Say that the robot is validated, and it can repeat the dynamics of the real robot with a desired level of accuracy, then it is possible to use the simulated robot to also validate the suction gripper by observing and recording the suction gripper behavior. Also, the interaction between the suction gripper and a picked object and the interaction between a tossed object and the spot where it lands must be parameterized to be able to have a virtual toss environment that can replace the real world when validating and developing a toss controller.

3.6.2 Simulation software and models

AGX Dynamics[4], hereafter AGX, has been used to simulate the dynamics of the TOSS scenario. AGX gives accurate force calculations and enables physically based parameters to be tuned from extremely soft to super stiff joints and contacts. An impact will propagate through the simulated mechanical structure instantaneously. A suction gripper model has been implemented in the AGX toolkit, enabling quick and easy configuration of suction gripper dynamics. The simulation models are defined in a declarative format, BRICK, being developed parallel to the I.AM. project. This enables composition and hierarchical modeling of reusable components with YAML based text declarations, with references to control signals. Instead of having to convert other formats, like URDF which is widely used within robotics, it is possible

[4]https://www.algoryx.com/agx-dynamics/

to extend the existing files with string references. Deliverable D5.8 provides further details about these aspects.

3.6.3 Simulation runtime

A concept has been developed and a primary runtime is implemented. The runtime is a thin Python layer with the working name GLUE, for instantiating the simulation environment and for doing parameter validation. Given a BRICK model the runtime initializes a synchronous communication over a, soon-to-be open source, communication framework CLICK. CLICK is also developed in parallel with I.AM. to enable BRICK robot communication. CLICK is implemented in C++ and can be integrated with a client of any controller framework. Within another project where Algoryx is involved, e.g., the company ABB has integrated CLICK with their virtual controller, which also enabled Algoryx to reuse the CLICK integration within yet another project. Deliverable D5.8 provides further details also about CLICK.

3.6.4 Validation procedure

A collaboration between the partners of I.AM has started the development of a Python framework for validating simulation components. The idea of the framework is to start from real-world data in a structured and documented format, and using this be able to both initialize simulations in any recorded configuration and start the simulation from that state. The simulation trajectory is validated through its comparison with the following real-world trajectory. The framework is used for offline parameter validation, where the parameters are validated outside the framework, as explained in 3.2.3. The full concept of GLUE includes online validation of the simulation parameters with a control system in the loop.

3.6.5 Expected user experience

A control system developer implementing a QP controller using the `mc_rtc` framework with access to a URDF file describing a robot of interest should be able to augment the robot with flexible joints and a suction gripper using the BRICK modeling format. 3D torque/force sensors can be added to any joint, and it is possible to attach an IMU sensor anywhere on the robot.
By authoring a simulation environment with validated simulation models, i.e., box, conveyor etc., using BRICK, the control system can synchronously control the simulated robot if implemented using a controller framework with CLICK client integration. The joint signal inputs can either be position, velocity or torque, and each joint does automatically have position, velocity and torque sensors. Boolean signals, like the one for enabling or disabling the vacuum, can be declared generically. Simulation model parameters are possible to validate using a different runtime, which optimize a set of parameters by trying to match simulation trajectories/behavior with real world trajectories/behavior. The parameter validation runtime support quick reloading of the simulation environment in reconfigurable initial states to enable instantiation from variable recorded real world states and search for optimal parameters in a predefined parameter space.

3.6.6 Release of GLUE

The goal is to release the interface for GLUE with the I.AM. project deliverable D1.2 ”Physics Engine API for Learning, Planning, Sensing, and Control”, which is due in June 2023. The parts of GLUE will step by step become available as open source. For example BRICK is currently being re-engineered in C++, to

enable integration with native runtime environments, and a public BRICK release is not being considered before that work is done. Further details about GLUE, BRICK, and CLICK can be found in the meantime in the I.AM. project deliverable D5.8.

4 Performance testing

With all the components developed, testing can be done to evaluate the performance of the tossing algorithm, this will then be compared to the pick-and-place scenario. First, we will discuss how performance can be measured to compare tossing to picking and placing. Second, the test plan will be discussed, followed by the test setup used for performance testing. Lastly, the test results will be shown together with a sensitivity analysis. This section details the test plan used to compare the tossing algorithm with classical placement of items.

4.1 Performance evaluation (KPIs)

In this section, we discuss how performance in the current industry is measured and how it will be used to evaluate the performance of the TOSS scenario. In the current industry, performance is measured by means of key performance indicators (KPIs). Key performance indicators are measurable values that indicate how well a system is performing, which makes comparison between systems possible through these measurable values. First, we will discuss the key performance indicators that are used in the current industry. In the current industry, the following KPIs are measured:

- **Throughput [items/hour]** (The number of items the system is able to successfully process in 60 minutes)
- **Average cycle time [s]** (average time it takes from one pick to the next pick)
- **Average pick and place time [s]** (average time it takes from pick to place)
- **Mean time between failures [hours]** (average time between system breakdowns)
- **Mean time to recovery [s]** (time required to repair a system and restore it to full functionality)
- **Retry rate/pickability [items]** (average amount of picks needed before an item is successfully picked)
- **Accuracy [%]** (The input of items compared to the output)
- **Quality [%]** (the percentage of items that did not receive noticeable (upon inspection) damage from being handled by the system.)

The throughput is defined as the average amount of items the system is able to handle in 60 minutes. This includes waiting times such as switches of totes and waiting for errors to recover. The average cycle time is defined as the time it takes from pick to pick. More elaborate, the cycle starts once an item is picked and ends when the next item is picked from the tote, thus completing one complete cycle. The average pick and place time is the time it takes from the moment the robot has picked an item till it has placed the item at the designated target. The difference, therefore, between the average cycle time and the average pick-and-place time, is the time it takes for the robot to move to the next pick, in Figure 13

one can see a diagram visualizing the cycle time and pick and place time. The mean time between failures describes the average time between system breakdowns and is usually calculated by dividing the total active time by the number of operator interventions. The higher the mean time between failures the better, as this means no operator interventions are needed and the robot can continue happy flow. If the robot does go into an error state, the mean time to recovery will indicate how long it will take for the robot to recover from the error before it can continue its happy flow. The retry rate (also referred to as pickability in industry) describes how well the robot is able to perform a pick. Sometimes, the robot has trouble picking an item and it has to try multiple times before it is able to successfully grab an item, the retry rate gives an indication of how many times the robot needs to pick on average before it has an item attached. The system should always pick/place the correct number of items. The accuracy value is the percentage of item transfers that result in the correct number of items being picked from the source container as well as the correct number of items being placed at the target. Situations that could prevent this from happening are e.g: losing items while picking or placing, double picks, and accidental drop-ins. No visible and/or structural damage to the items should occur when being handled by the system. The quality value is the percentage of items that did not receive noticeable (upon inspection) damage from being handled by the system.

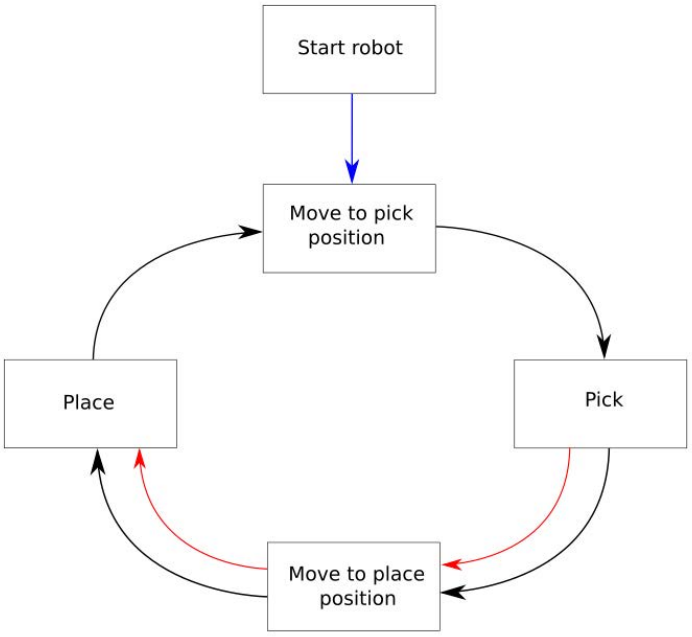


Figure 13: Diagram of the cycle time and pick and place time. Following the black arrows, a full cycle is done when the same state is reached where you started (e.g from one pick to the next pick). The red arrows indicate the pick and place time. The blue arrow indicates the initial step necessary to enter the cycle.

Using all these performance indicators and looking at the various combinations, the performance of the system can be determined. For this project, a goal of a 10 % speed-up in cycle time is set for the TOSS scenario. Therefore, the average cycle time and average pick and place time are good key performance indicators to monitor if and where a speed up might take place. We will not use the throughput as a KPI

for the TOSS scenario. The throughput includes the cycle time as well as tote waiting time and waiting for errors, which the TOSS scenario does not aim to improve. Because the tossing scenario only concerns the placement of items, we will not concern ourselves with KPIs that describe the picking phase such as the retry rate. We will not use the mean time between failure as a key performance indicator, as well as the mean time to recovery, as these are also KPIs the TOSS scenario does not aim to improve (and also require extensive endurance testing). However, during testing, it will be monitored if tossing does lead to more operator interventions compared to the traditional placement of items. When placing items from a tote on a belt there is a chance that the item will be lost during the motions of the robot. To monitor if this will occur more often during tossing, the accuracy will be monitored. Within the accuracy, we can also monitor if items do end up where we want them to (and for example do not tumble or land outside the target). The quality is also something that we do not aim to improve within the I.AM. project, but also for this KPI we will monitor during testing if the items suffer visible damage while being handled. In summary, the KPIs that will be used to compare the performance of tossing with placing are:

- **Average cycle time [s]**
- **Average pick and place time [s]**
- **Accuracy [%]**

Lastly, some monitoring needs to be done to test for the applications described in Section 2, for some of these applications it is important that, for example, the item is placed at a certain angle or that the barcode of the item does not face down. These extra requirements per application are shown in table 4.1.

Characteristics of the different TOSS scenarios		
	Infeed	**Traysorter**
Application	Bin-to-Belt	Bin-to-Window
Conveyor speed	Up to 1.5 m/s	Up to 1.5 m/s
Tumbling allowed	Allowed (not desired)	Allowed (not desired)
Window size	$x = 1300$mm, $y = 2100$mm	$x = 400$mm, $y = 400$mm or $x = 400$mm, $y = 600$mm
Stance requirement	Barcode not down	Barcode up
Rotation requirement	Short edge leading w.r.t. crossorter, $30° \pm 5°$	Rotation arbitrary
Mass range	< 4kg	< 4kg

The requirements together with the presented KPIs will be monitored during testing as shown in the testplans in Section 4.2. In Section 4.4, the results for these KPIs and requirements for the TOSS scenario can be found.

4.2 Test plan

In this section, the test plan to analyze the performance of the tossing algorithm in an industrial-like scenario is described. First, the tossing algorithm is detailed, thereafter the place algorithm is explained and lastly the metrics by which the test executions are measured are discussed.

4.2.1 TOSS algorithm

For the tossing algorithm, the planner, controller, the simulation environment, and the object-conveyor impact model parameters described in Section 3.2 are used. As the I.AM. project is not focusing on speeding up the picking process, it was decided to use a simple pick algorithm. In the industry, picking is done by using computer vision to detect items and determine grasp poses. Now, items are placed at a known position and picked with a predetermined grasp pose in the middle of the item. The items that will be picked and tossed are boxes as 80% of the items handled in the Vanderlande industry are boxes. In this test plan, the boxes will be of uniform weight, because the used planner (see Section 3.4) assumes items of uniform weight. It is assumed that the properties of the box, listed as box 5 in the `item archive`[5] are known. First, the planner is used to determine the release pose and velocity of the box given the data of simulated **N** randomized tosses with the box conveyor model in combination with a desired rest position fulfilling the characteristics of table 4.1. Given the release configuration, the controller in combination with the dynamical system is used to generate the motion after picking up the box.

4.2.2 Place algorithm

In order to compare the KPIs of the tossing algorithm with picking and placing, a pick and place algorithm also has to be made. To make a fair comparison, the same robot will be used with the same pick position and target, where the robot is allowed to have the same motion limits as the toss algorithm. The place algorithm uses the same motions for moving to the pick position and picking the item as the toss algorithm. The only difference is the place motion as opposed to the toss motion. The place motion has been created using a cubic spline that the robot will follow while staying within the motion limits.

4.2.3 Metrics

As listed above, the tossing and placement algorithms have been compared on the average cycle time, average pick and place time, and accuracy. Below, it is explained in more detail how each KPI is evaluated. The item handling consists of several states. The state diagrams for both tossing and placing are illustrated in Figure 13

The average cycle time for both tossing and placing is the time it takes to go from moving to the pick position of the first box to moving to the pick position of the second box. The average pick and place time is the time it takes to go from the state of picking up the item to the state of placing the item. The time and state switches are stored using the logging framework of the QP robot controller `mc_rtc`[6].

The accuracy of each placement and toss is evaluated as the percentage of items that are successfully placed or tossed. The properties that determine a successful accomplishment of this task are dependent on our industry scenario. The two scenarios taken into account are the traysorter and infeed scenarios described in Table 4.1.

[5] http://impact-aware-robotics-database.tue.nl/objects

[6] https://jrl-umi3218.github.io/mc_rtc/tutorials/usage/logging.html

For the traysorter scenario an item is successfully handled if:

- The rest position of the handled item is on the conveyor in a defined window of 2100×1300 mm.
- The yaw of the orientation of the rest position of the box is within the 5 degree error margin.

For the Infeed scenario an item is successfully handled if:

- The rest position of the box is inside the traysorter, equal to a window of 400×600 mm.

4.3 Test setup

To show the potential of the TOSS scenario, experiments need to be done with a setup that resembles what is used in the current industry. For this, a setup was made that consists of a robot located on a table with a compartment to pick items from. A conveyor belt is added at the back side of the table such that the robot can pick from the totes and then either toss or place items on the conveyor belt at distances that resemble the ones in the current industry. The test setup can be seen in Figure 14, while the schematic can be found in Figure 15.

Figure 14: The test setup used for the validation of the TOSS scenario with the workspace of the UR on the left, the belt in the middle and the work space of the Franka Emika robot on the right

The test setup contains both a UR10 by Universal Robots and a robot by Franka Emika. The UR10 is a 6-DoF robot. The Franka Emika robot is a 7-DoF robot capable of impedance control. A GS02 gripper by Smart Robotics will be attached to the end effector of each robot. A suction cup can be attached to the gripper. The most commonly used suction cup by Smart Robotics is the FLI70 with a foam lip by Piab [7]. The setups will both support blow-off to gain control over the release phase when tossing an item. In Figure 15, one can see a schematic of the test setup.

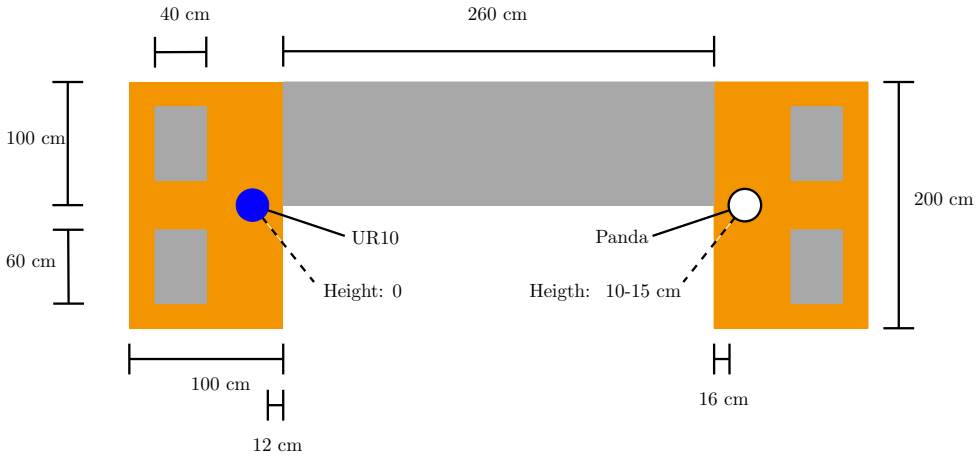


Figure 15: A schematic of the test setup, where the table is shown with the color ocher and grey is the conveyor belt.

Also, an OptiTrack system is present to track the items that are tossed, such that its position can be monitored during flight for debugging purposes. In order to use this OptiTrack system, items have to be equipped with M3 stickers, an example of an item equipped with M3 stickers for the OptiTrack system can be seen in Figure 16.

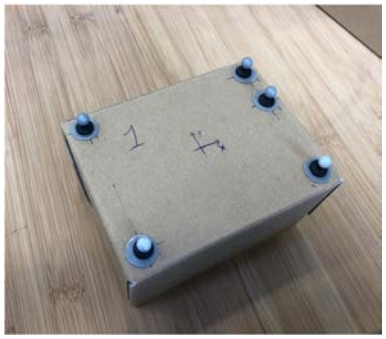


Figure 16: Example of item equipped with M3 OptiTrack markers.

To evaluate the performance of tossing on a traysorter, a traysorter is placed on top of the belt. This is illustrated in Figure 17. The crossorter infeed scenario will be mimicked by tossing items on a conveyor belt moving at a speed of $1.5m/s$. The conveyor-box model is also optimized for a box landing on a conveyor belt.

[7] http://www.pi-vacuum.com/pdf/Suction%20cup%20piGrip_GB.pdf

4.4 Tossing test results

In this section, the tossing test results will be discussed. All tests were repeated 20 times to find the average values for the KPIs, where the results were consistent. After running the planner to find a release pose, a toss was found which is shown in the following images, the same toss was used for the traysorter scenario as well as the cross sorter scenario as described in Section 2. To mimic the crossorter infeed scenario, tossing is done on conveyor belt moving with a speed of $1.5m/s$ as described in Table 4.1, the toss can be seen in Figure 21 in Appendix A.

(a) Picking item.

(b) Moving out of tote with item.

(c) Dynamical system, moving towards toss.

(d) Tossing at release pose.

(e) Item landed in its rest pose.

(f) Robot moving back to new pick.

Figure 17: The Franka Emika robot, tossing items in the traysorter in various steps.

In the table below, one can find the results of the benchmark done for the traysorter scenario:

Benchmark results for traysorter scenario		
	Toss	**Place**
Average cycle time	6.87 s	7.64 s
Average pick and place time	2.33 s	3.27 s
Accuracy	100%	100%
Failures	0	0
Barcode up	100%	100%

The average pick and place time is 2.33 seconds for tossing while it is 3.27 seconds for placing. The average cycle time is 6.87 seconds for tossing and 7.64 seconds for placing an item. It is important to note that the throughput of an item handling application is determined by the number of unsuccessful places, errors, but most importantly the average cycle time. In this case the picking motion takes up a relatively large part of the total cycle time as it has not been fully optimized. Still a speed up of 10.1% in the total cycle time is realized, as can be seen in Figure 18. Looking just at the pick and place time, the tossing algorithm is however actually 28.7% faster.

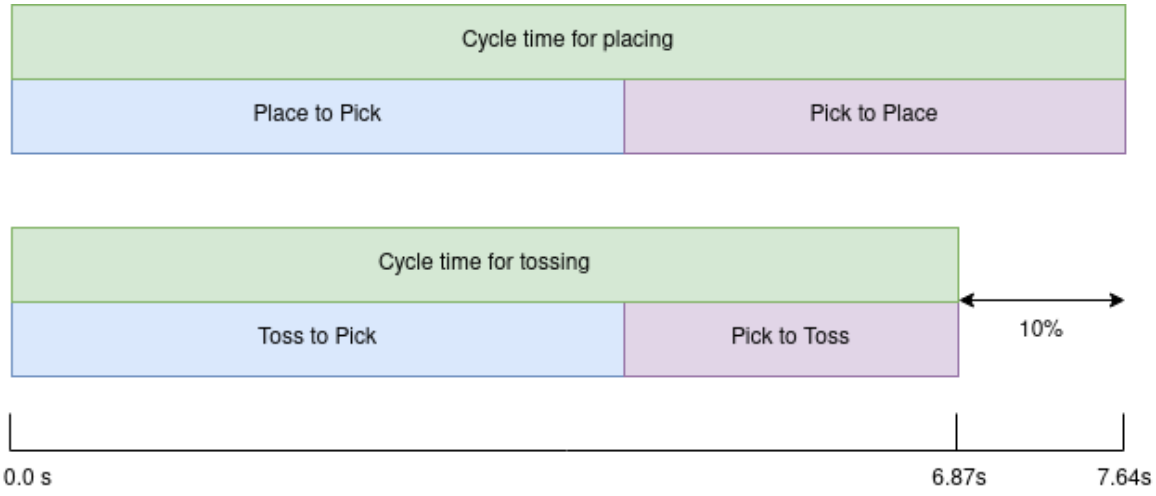


Figure 18: Timeline comparison of tossing and picking motion. Illustrating the 10% performance gain.

For both tossing and placing, all items successfully landed in the traysorter, resulting in 100% accuracy for both cases. No errors caused by the tossing motions occurred during the experiments. In all cases when the item was picked with the barcode up the resulting toss and place also resulted in the item landing with the barcode up.

Benchmark results for crossorter infeed scenario (tossing on a conveyor belt)		
	TOSS	**Place**
Average cycle time	6.87 s	7.64 s
Average pick and place time	2.33 s	3.27 s
Position accuracy	100%	100%
Orientation accuracy	0%	70%
Failures	0	0
Barcode not down	100%	100%

For the crossorter infeed scenario (tossing on a conveyor belt), the same optimized toss and place were used as in the traysorter scenario (with the robot making the same motions), which results in the same average cycle times and average pick and place times as the traysorter scenario. In the infeed scenario, the aim is to place items on a moving conveyor belt at the same consistent angle. When tossing, all items always landed on the conveyor belt (and also never tumbled off the the conveyor belt), this was also true while placing items. However, in the tossing scenario, the items always landed or tumbled such that it landed in rest pose with a rotation error bigger than $5°$. When placing the item, the rotation error was bigger than $5°$ 30% of the time, which makes both scenarios unusable for the crossorter infeed scenario without external applications to correct the angle of the box. However, for a bin-to-belt scenario, where the rotation is arbitrary, the tossing scenario is valuable as it realizes a speed up compared to placing. For the crossorter infeed scenario the item should not land with the barcode down. All tosses and places resulted in a rest pose of the item where the barcode was not down. On two occasions the item tumbled on its side, but still not with the barcode down.

A comparison was made between simulation and real-world experiments to see if differences occur. By using markers and the optitrack system described in Section 4.3 the box can be tracked inflight and its position tracked in x,y, and z. In Figure 19, the comparison of simulation (blue line) and real world (red line) for x, y, and z is shown

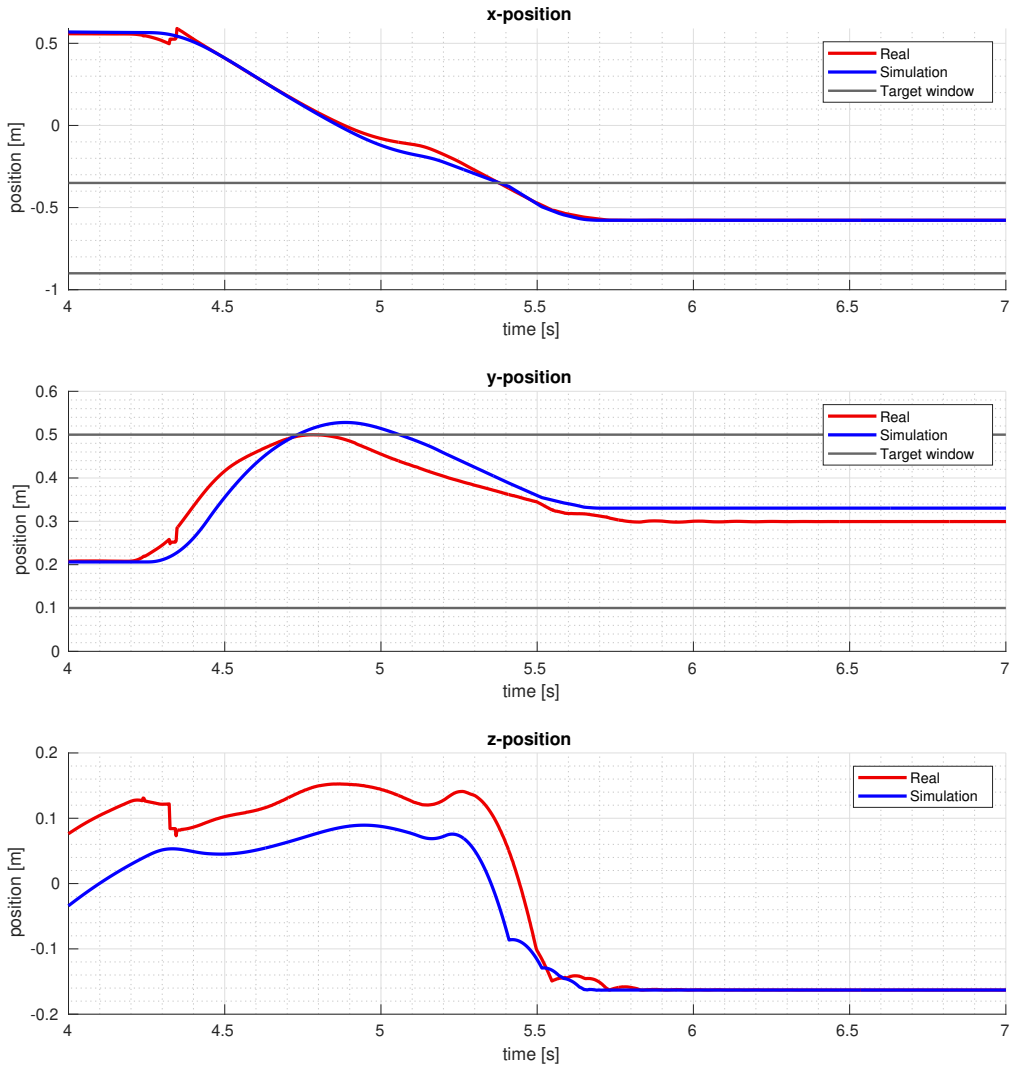


Figure 19: The x,y,z positions of the box over time. Both of a real (red) toss, captured by optitrack and that of simulation (blue) in the AGX environment. The target window (black) is also shown.

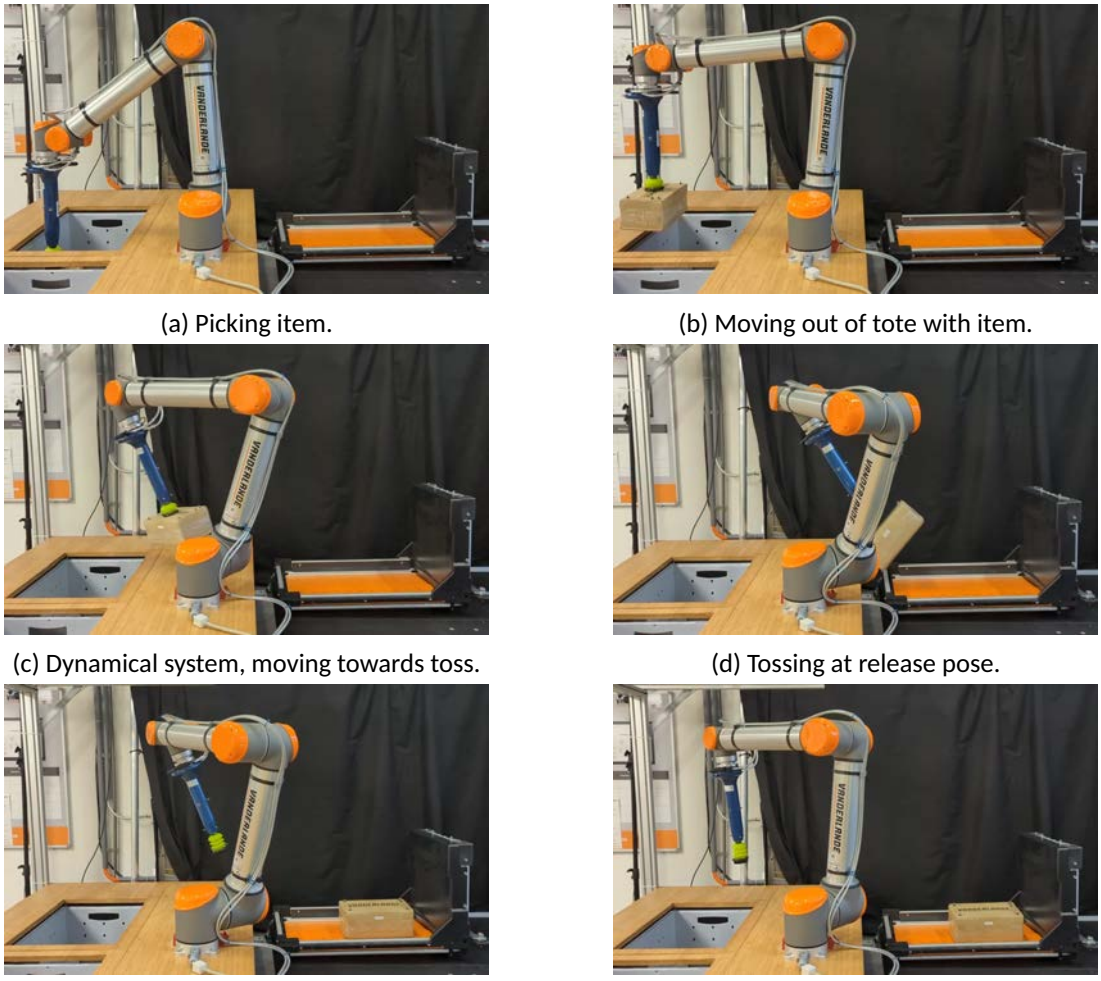
As can be seen, there are some slight differences between simulation and real world, this could be explained by fact that in the simulation the suction-cup is modeled as rigid while in the real world the suction cup is compressed while holding a box. It is expected to be improved by using the box-suction cup holding model as well as the suction cup release dynamics model described in Section 3.2.1 and 3.2.2, that were not yet integrated for the benchmark tests.

4.5 Sensitivity analysis

To get an insight into the needed accuracy of the parameter identification, a rough sensitivity analysis is done. This is done by executing the same tossing motion but with slight variations in the properties of the box and picking locations. For these tosses only the accuracy is taken into account as the cycle time and pick and place time vary little. Each of the variations is tossed 5 times. The variations as well as their success rate has shown in Appendix B. The properties of each box can be found in the `item archive`[8].

4.6 Using the UR10

To show that the tossing framework developed within the I.AM. project can be easily reused on other robots, the toss inside a traysorter has also been demonstrated for the UR10. This demonstration is illustrated in Figure 20.



(a) Picking item.

(b) Moving out of tote with item.

(c) Dynamical system, moving towards toss.

(d) Tossing at release pose.

(e) Item landed in its rest pose.

(f) Robot moving back to new pick.

Figure 20: The UR10, tossing items in the traysorter in various steps.

[8] `http://impact-aware-robotics-database.tue.nl/objects`

A full benchmark with the UR10 has not been done, however, when doing testing with the UR10 the performance was similar compared to the benchmark done for the Franka Emika robot.

4.7 Evaluation

In conclusion, the benchmark results show the potential of the I.AM. tossing scenario on a experimental setup that was build to mimic real setups existing in the current industry, especially for the traysorter scenario and a bin-to-belt system. For the crossorter infeed application, the angle of the rest pose of the items is not consistent enough to be already be employed in real applications. At the same time, this is to be expected because the developed planner does not consider such a quantity and a clear strategy for improvement is available.
The sensitivity analysis shows that item parameters do not have to be estimated perfectly in order to toss accurately within a desired window, nor picked up with millimiter accuracy. The results show that by tossing items a 10% decrease in cycle time can be achieved when comparing on the same experimental setup. As mentioned, the tossing scenario is not yet ready in its current state to be used in the industry. This is also because in the current tossing algorithm, the item parameters are assumed to be known such that toss planning optimization can be done offline. In the industry, thousands of different items are used and the inertia, stiffness, and geometry of the item to be handled is not known a priori. It is therefore required, in the near future, to develop fast online parameter identification and tossing optimization. For inertial parameter identification, the first steps has been already taken as described in this deliverable and related publications.

5 Conclusion

This deliverable described the implementation details of I.AM. tossing technology on UR10 and Franka Emika robots, including testing and benchmarking results of dynamic (non-zero speed, and object-and-environment aware) tossing in comparison with standard (almost zero-speed, object-and-environment unaware) placing. Second, the deliverable provides an overview of the potential value of impact-aware manipulation technology for specific existing applications, as provided by industrial partners Vanderlande and Smart Robotics. These applications are the traysorter and (cross-sorter) infeed applications. Furthermore, the components that have been developed during the I.AM. project, which are deemed crucial for accurate tossing, have been explained. An architecture was made to make sure that separate developed components can be integrated together to perform accurate tossing. Models were developed to predict behaviour of an item when it is held by a suction cup, as well as the release dynamics of said suction cup. Also, the impact and the friction dynamics between the box and the conveyor belt have been detailed. In order to use these models, and make a prediction on the behaviour of an item when it is being tossed, its properties must be known. Therefore, a payload identification method was explored to determine the objects inertial parameters online. A planner was developed that can compute a release configuration, given a target rest orientation of an item. In order to reach the desired end-effector release configuration, a controller was developed that plans and executes a non-linear point-to-point motion in joint space, where the controller is constrained by the robot dynamics and its mechanical and electrical hardware features. The controller is a DS-based `mc_rtc` controller with AGX dynamics simulation for tossing the object. Using the models, a simulation environment of the tossing scenario has been created for validation and development of control algorithms.

The box-conveyor models, the planner, and the controller, and the simulation environment, have been integrated for real-world testing. Payload estimation could not be fully integrated and items with known parameters were used to perform tossing. Tossing has only been done with carton boxes, which are however extremely represantive: according to Vanderlande, 80% of items that are handled in their systems in the applications mentioned above are boxes.

A test setup consisting of a Franka Emika robot and a UR10 were used for real-world bench marking of the I.AM. tossing technology. Key performance indicators have been identified for a performance comparison between tossing and picking-and-placing. A pick-and-place algorithm using the same setup with the same control framework has been made for a fair comparison. A benchmark was done showing a 10% speed up in cycle-time while still being accurate enough for industrial applications such as the traysorter and bin-to-belt systems. For the infeed scenario the tossing was not able to throw the box at a consistent orientation. A sensitivity analysis was done to check whether the tossing algorithm is still accurate when parameters such as the pick position, item dimensions, and item weight deviate. The sensitivity analysis shows that tossing is still accurate with deviating parameters. The tossing scenario is, however, not yet ready in its current state to be used in the industry. This is because in the current tossing algorithm, the item parameters are assumed to be known such that optimization for a toss can be done offline. In the industry, the item variety is large and it is most often not known what item is being handled next. Therefore, parameter identification and tossing optimization have to be done online. However, the report shows that tossing can decrease the cycle time of robotic systems in logistics by 10%, showing the potential impact it can have on the current logistics industry. The I.AM. partners are committed to address at best these missing components during the remaining project time and to set up other join initiatives for addressing the remaining aspects after the project's end.

6 REFERENCES

[1] M. L. S. Lubbers, J. van Voorst, M. J. Jongeneel, and A. Saccon, “Learning Suction Cup Dynamics from Motion Capture: Accurate Prediction of an Object’s Vertical Motion during Release,” in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2022)*, October 2022, pp. –.

[2] M. J. Jongeneel, L. Poort, N. van de Wouw, and A. Saccon, “Validating Rigid-Body Dynamics Simulators on Real-World Data for Robotic Tossing Applications,” *Preprint on HAL*, Dec. 2022. [Online]. Available: https://hal.archives-ouvertes.fr/hal-3903526

[3] A. Kurdas, M. Hamad, J. Vorndamme, N. Mansfeld, S. Abdolshah, and S. Haddadin, “Online payload identification for tactile robots using the momentum observer,” in *2022 International Conference on Robotics and Automation (ICRA)*. IEEE, 2022, pp. 5953–5959.

[4] S. S. M. Salehian, M. Khoramshahi, and A. Billard, “A dynamical system approach for softly catching a flying object: Theory and experiment,” *IEEE Transactions on Robotics*, vol. 32, no. 2, pp. 462–471, 2016.

[5] K. Bouyarmane, K. Chappellet, J. Vaillant, and A. Kheddar, “Quadratic programming for multirobot and task-space force control,” *IEEE Transactions on Robotics*, vol. 35, no. 1, pp. 64–77, 2019.

[6] M. B. Bombile and A. Billard, “Dual-arm control for coordinated fast grabbing and tossing of an object: Proposing a new approach,” *IEEE Robotics & Automation Magazine*, 2022.

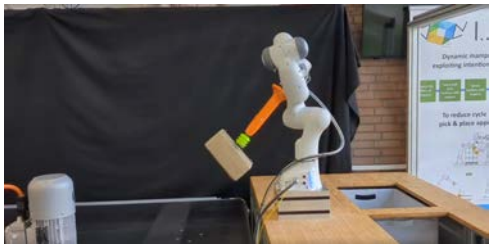
A Infeed scenario tossing

In the following figure, one can see the tossing that was performed on a moving conveyor belt to mimic the cross-sorter infeed scenario, as described in Section 2.

(a) Picking item.

(b) Moving out of tote with item.

(c) Dynamical system, moving towards toss.



(d) Tossing at release pose.

(e) Item landing on conveyor belt

(f) Robot moving back to new pick and item moving along the belt.

Figure 21: The Franka Emika robot, tossing items on a belt to mimic the cross-sorter infeed scenario in various steps.

B Results of the sensitivity analysis for tossing

To get an insight of the needed accuracy of the parameter identification, a preliminary sensitivity analysis is performed. This is done by executing the same tossing motion but with slight variations in the properties of the box and picking locations. The results of this sensitivity analysis can be seen in the following Table B

Results for sensitivity analysis infeed scenario			
Variation	**Accuracy Traysorter**	**Position Accuracy Infeed**	**Orientation Accuracy Infeed**
Box 1	100%	100%	20%
Box 9	100%	100%	0%
Box 6	100%	100%	0%
Box 5, 2cm offset x axis	100%	100%	0%
Box 5, 4cm offset x axis	100%	100%	0%
Box 5, 6cm offset x axis	80%	100%	20%
Box 5, 8cm offset x axis	40%	60%	0%
Box 5, 2cm offset y axis	100%	100%	0%
Box 5, 4cm offset y axis	100%	80%	0%
Box 5, 2cm offset x and y axis	100%	100%	0%
Box 5, 4cm offset x and y axis	80%	100%	20%
Box 6, 2cm offset x and y axis	100%	80%	0%
Box 6, 4cm offset x and y axis	80%	80%	0%