PET-Physics: A Packing Evaluation Tool Considering Physics

Florian Enzenhofer[1,2], Josef Ondřej[1], and Andreas Nüchter[2]

***Abstract*— Without having a robotic bin packing setup, the prediction whether a packing plan results in a stable pallet is based on an informed guess based on prior experience. Since such a setup is costly, for many algorithms it is unclear if they are suitable for production. To close this gap, we provide our simulation framework `PET-Physics` that enables researchers to assess a solver's quality. Based on the pose recording of all bodies, it is possible to check not only for static stability, but also dynamic stability of a packing plan. A novel feature of our tool is the provision of data that supports the detection of boxes that might be overloaded. We demonstrated the realism of our framework by comparing simulation runs with real-world palletization. Our packing evaluation tool considering physics is publicly available at `https://github.com/floriankagerer/pet-physics`.**

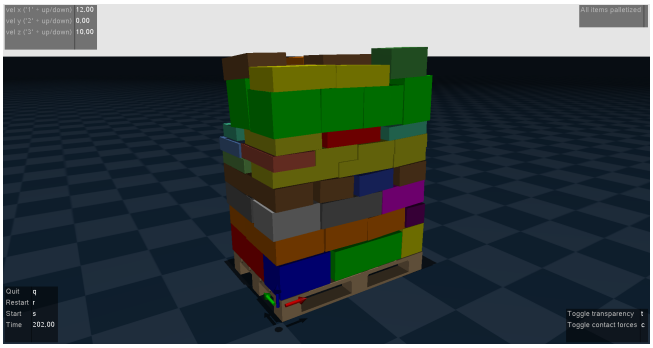

Fig. 1: Visualization of a packing plan with `PET-Physics`.

I. INTRODUCTION AND BACKGROUND

Researchers in the field of robotic bin packing strive to develop algorithms that optimally pack a set of items into a bin. In order for a robot to be able to palletize the boxes, it requires instructions of where, in which pose, and when to place a certain item. The execution of this packing plan results in stacks of items that are either stable or tend to collapse.

Assessing the stability of a packing plan in reality is not only time consuming, but also costly and challenging to realize. On the one hand, you require a robot to accurately place boxes in pre-computed positions. On the other hand, you need a tremendous set of boxes containing different items with distinct weights to build many pallets. Notably, repetitive testing with the same boxes yields divergent results, as box firmness degrades with each palletization cycle.

We contend that the use of a physical simulation framework overcomes these disadvantages, provided that it simulates reality in an accurate way. For this reason, we developed a packing evaluation tool that considers physics (`PET-Physics`) to virtually assess a pallet's stability. It uses the physics engine MuJoCo, which was published in [1], and is capable of simulating the carrying out of a packing plan and of evaluating static and dynamic stability. The simulated pallet of a packing plan is shown in Fig. 1.

Although Bischoff already mentioned the significance of stability in practice in 1991 [2], one of the first algorithms that solves the three-dimensional bin packing problem did not take stability into account [3]. Furthermore, Bortfeldt and Wäscher stated that stability is an essential objective to achieve and of similar importance as space utilization [4]. Junqueira, Morabito, and Yamashita strengthened the importance by distinguishing two types of stability [5]. On the one hand, static stability, also called vertical stability, describes whether a pallet is stable if gravitational force acts on the stacked boxes. On the other hand, dynamic stability, or horizontal stability, characterizes whether packed boxes are able to withstand horizontal displacements, e.g., when the pallet is accelerated by conveyor technology.

Due to the fact that in air cargo stability consideration is essential, Paquay, Schyns, and Limbourg included stability, fragility, and weight distribution constraints to the formulation of the optimization problem [6]. An underlying principle of the authors' stability constraint was the hypothesis, that the center of gravity of a box lay in the support basis, if at least three vertices of the bottom side were supported.

In [7], Tkaczyk et al. presented an experimental test setup to assess the dynamic stability of a pallet that was wrapped with stretch film. The main objective was the reduction of stretch film consumption while fulfilling the required securing of cargo. For the purpose of finding optimal packing of cargo, which included the material of the boxes too, many more tests were required. This would result in exorbitant costs and significant time requirements. For this reason, the use of a physics simulation is appropriate, provided that reality is accurately reflected.

An evaluation framework to analyze the dynamic stability of stacked items was developed in [8]. In order to do this, Roy, Vyas, and Cherian simulated the application of a disturbance function to the pallet on that items were stacked. After removing the disturbance, it was checked whether the angle of rotations returned to the value that they had before applying the disturbance.

In addition to a benchmarking dataset, Kagerer et al.

[1]Florian Enzenhofer (`florian.research@icloud.com`) and Josef Ondřej are with the Department of Artificial Intelligence Competences, TGW Logistics Group, Marchtrenk, 4614, Austria

[2]Florian Enzenhofer and Andreas Nüchter are with the Department of Informatics XVII Robotics, Julius-Maximilians-University Würzburg, Würzburg, 97074, Germany

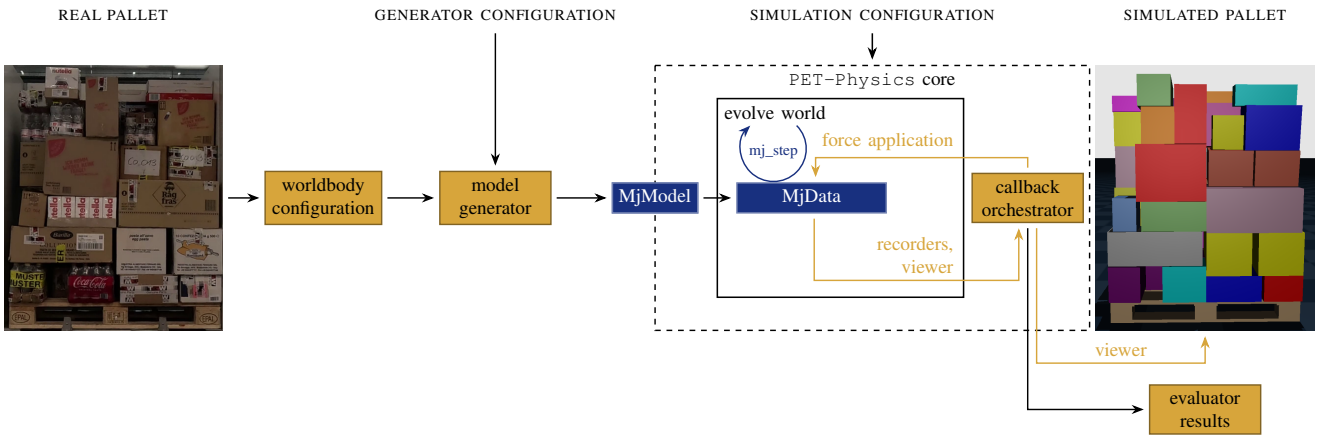

Fig. 2: From reality to simulation with PET-Physics. Based on a packing plan, we compute the worldbody configuration, which consists of the model's name, the carrier on that the bodies are packed, the bodies, and their initial and target positions. Together with a configuration of the model generator, which specifies, e.g., the used solver, we receive the objects that MuJoCo requires. Our core is responsible for evolving the world, which includes the triggering of callbacks at the correct times. After completing a simulation run, the evaluators provide results that are the basis of the stability assessment.

provided an evaluation function that used Blender's[1] rigid body simulation [9]. This approach is limited to assess static stability after palletizing all items. Another more elaborate benchmarking system was provided by Wang et al. [10]. The authors developed a simulation environment based on PyBullet [11], enabling researchers to simulate robotic bin packing, hence, the simulation replicates the palletizing process in reality. However, none of the simulation environments were compared with reality.

Besides, Chien and Wong developed a packing task simulator with PyBullet. In it, two RGB-D cameras were used to retrieve input data for an deep reinforcement learning algorithm [12]. Due to the use of the cameras' images, the trained algorithm was capable of packing arbitrary shaped items. Even though this simulator virtualized the generation of a bin's state, it lacked an evaluation of realism as well.

For the purpose of developing a physics-informed learning approach, Mazur, Melsbach, and Schoder used a physics simulation. With this approach, they outperformed existing methods in cargo loading stability [13]. Anyhow, since the results of such kind of approaches strongly depend on the accuracy of a physics simulation mirroring reality, it is crucial to have a simulation which has its realism validated.

To this end, we provide an open source packing evaluation tool that is designed for robotic bin packing. It is applicable to (a) evaluating whether a packing plan leads to a stable pallet, to (b) analyzing under which conditions the pallet is still stable if exposed to external forces, and to (c) providing a data foundation to estimate whether the boxes of a packing plan are overloaded. We demonstrate the realism of our packing evaluation tool PET-Physics by visually comparing the simulation with reality. To work with our framework, visit https://github.com/floriankagerer/pet-physics.

[1]https://docs.blender.org/manual/en/latest/physics/rigid_body/introduction.html

II. SYSTEM OVERVIEW

For the sake of virtualizing the palletization process, we developed a physics simulation with focus on the packing part. This supports us in detecting problems in a packing plan, which lead to a pallet collapse, long before we physically pack objects.

In our robotic setup, conveyor technology transported items that the robot packed to a pick up point. When arriving there, two fork tines took an item and secured it against unwanted movements. The robot, together with the fixed item, moved into the packing space. This space describes the possible target positions of the objects and is usually two meters times the base area of the carrier.

The carrier, e.g., an Euro-pallet, was clamped within a palletization apparatus. The main purpose of this casing was to avoid having dropped boxes all over the ground if an item stack collapsed. In fact, we did not utilize this casing to let boxes lean against the walls.

In our opinion, the most critical part regarding stability in palletizing happens within the packing space. Not only the robot's accuracy of placing a box exactly at the computed coordinates has a great impact on a packing plan's stability, but also the way how the robot releases an item.

Finally, after successfully packing all items, the palletization apparatus including the pallet was moved to another machine that separated the apparatus from the carrier. Subsequently, the carrier and the packed items were shrink-wrapped to ensure stability during transportation.

III. MODELING

In order to simulate the palletizing instructions of a packing plan, our simulation framework PET-Physics needed to translate this plan into a simulation model.

We split the conversion into two steps, which are depicted in the left part of Fig. 2. The first step processes the information about the carrier and the placed boxes, while

the second is responsible for injecting information about the options to use in the simulation. We benefit from this split if we need to simulate a packing plan multiple times, e.g., due to an invalid run.

In the following, we describe the content of a simulation model and elaborate on the conversion steps.

A. Simulation Model – MjModel

The underlying physics engine of `PET-Physics` is MuJoCo [1]. It uses the MJCF modeling language to describe the simulation options and the world's bodies. Since MJCF is based on the XML schema, its content is well structured and human readable. Starting with the root element `mujoco`, this tag contains multiple sections, including `option`, `asset`, and `worldbody`. We briefly describe the content of the first ones, whereas the content of `worldbody` is explained in the next section.

In order to modify a setting of the physics engine, we set the value of the corresponding key in `option`. One parameter with that we experimented was the timestep. It defines the time span that passes while MuJoCo evolves the world by a single step. Clearly, finding a good value for the timestep is a tradeoff between accuracy and required simulation time. We identified the used solver as parameter, since it influenced simulation results. We used the Newton solver and the conjugate gradients method as solvers. Remarkably, this affected the simulation results. Form some packing plans, we observed explosions of multiple items, probably due to a floating point issue in the distance between them. For the same packing plan it happened, that we got the explosion using one solver but not the other, and vice versa.

The `asset` section gathers definitions of material, textures, and meshes, thus, it simplifies the definition of the bodies. After a material is defined, which includes the setting of material specific values, e.g., for the friction, it is assigned to a body in the definition of the world. Notably, since MuJoCo is unit-agnostic, we defined everything in SI units which simplified the deduction of other units, e.g., of the force. We considered this when we configured the bodies in the model.

B. Worldbody Configuration

The main effort to retrieve a simulation model from a packing plan was the conversion of the placement instructions to bodies in the MJCF format. Since the bodies' descriptions are independent of simulation options, this conversion marks the first step to generate a simulation model.

In contrast to our packing plan, MuJoCo defines the position of a body as the position of its center of mass. Additionally, for defining the size of a body in MuJoCo, half of the values of a box's length, width, and height are used. When we built a configuration, the bodies were not directly stored in the MJCF format, but as objects that easily serialized to MJCF. Due to the fact that it is impossible to spawn bodies during runtime in MuJoCo, we placed all bodies at some initial positions and stored the actual target positions in placement objects.

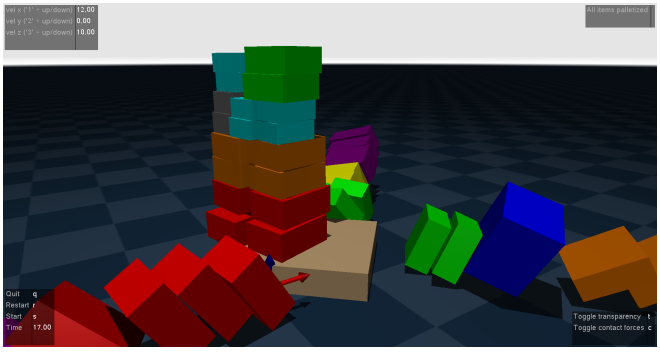

Fig. 3: After applying a significant force, the carrier shifted and multiple item stacks collapsed.

For the sake of clarity, the carrier was stored as a separate body object in the configuration. The carrier was directly placed at some fixed position, since its position did not change during palletization. Finally, the storage of the packing plan's identifier completed the worldbody configuration, which was used as input for the model generator.

C. Model Generator

The last step before having a model is to combine the information about the world with the simulation options. To this end, we developed a model generator that was responsible for computing simulation models.

The main responsibility of the generator was not only mapping the bodies to the simulation model, but also to correctly set the simulation settings. In addition, it replaced path placeholders, e.g., for the arrows of the coordinate system, such that the physics engine was able to load the correct bodies from given files.

Finally, we want to emphasize that the generation of the worldbody configuration before making a simulation model was beneficial in automatic stability assessment. Whenever an invalid simulation run happened, it was easy to either regenerate the model or to replace parts of the simulation options and run the assessment again.

IV. SIMULATION

The transition from a simulation model to evaluation results, based on the simulation state MjData, is the focus of this section. The main functionality of our tool is implemented in the `PET-Physics` core, which wraps the physics engine and simplifies reading data from the simulation state or manipulating it, see Fig. 2.

Since our core is responsible for controlling the simulation, it is the only entity that evolves the simulation state, i.e., calling MuJoCo's function `mj_step`. Subsequently, we explain the configuration of the simulation core and how callbacks and evaluators are used to assess a packing plan's stability.

A. Simulation Configuration

Depending on the simulation goal, the configuration of our core differed. Obviously, assessing a packing plan's dynamic

stability requires a different configuration than for static stability. Nonetheless, there are settings that are independent of the simulation goal. Firstly, the procedure that defines the order of used solvers and the attempts to obtain a valid simulation run. Secondly, the time span between moving two items from initial to target position. Thirdly, whenever one evaluates the stability of a packing plan, it is configurable whether all boxes are placed at once and then vertical stability is checked or the boxes are sequentially placed while stability is assessed. Obviously, stability in the first case does not imply stability in the latter.

Clearly, the simulation's accuracy strongly depends on the way how the placement of the boxes is implemented, cf. Section V. However, since we wanted to provide a simulation framework that is not trimmed to a single robotic setup, our tool "teleported" the bodies to their target positions.

In order to see whether a packing plan is dynamically stable, the configuration of a force profile, which is applied to the carrier, is required. This setting is useful if you want to approximate how the stacked boxes behave when the pallet is transferred, e.g., to a wrapping machine. Fig. 3 shows a packing plan that is dynamically unstable.

B. Callbacks

Every functionality, e.g., recording the pose of a box, was facilitated a callback. To retain performance, we implemented an orchestrator that kept track of when to run a certain callback, see Fig. 2. We measured the frequency as number of `mj_step` calls. Our packing evaluation tool provides different types of callback: (a) a viewer callback to visualize the current state of the simulation, (b) multiple recorder callbacks of that each records a specific physical quantity, (c) an early termination callback that decides whether to terminate the simulation if a box falls off the pallet and touches the floor, and (d) a force application callback that applies forces to specified bodies.

The underlying data for the stability assessment consist of recorded poses of each body. A pose consists of the position of the center of mass in $\mathbb{R}^3$ and the orientation of the body. In MuJoCo, the orientation is given as quaternion, which is easily convertible to roll, pitch, and yaw.

We utilized acceleration values of every body during a simulation run to decide whether the run was valid. To this end, we included the acceleration recorder callback that logs both linear and angular acceleration, capturing the items' three translational and three rotational degrees of freedom.

The recording of the force and penetration, respectively, is slightly different. Since we were not able to directly record the force's value that acts on top of a body, we needed to consider all contacts of the body's geometry objects. In order to correctly decide whether a force was present on top or at bottom of a body, the locations of a contact's bodies were required. By using these contact objects, we automatically retrieved the penetration that took place. The visual representation of contact forces is shown in Fig. 4.

A callback that optimizes simulation time is the early termination callback. As its name indicates, this callback will terminate a simulation run early if a stopping condition is satisfied. Obviously, a box that fell off the carrier indicates the stop of the simulation run. In contrast, the viewer callback was not only used to visualize the current simulation state, but also to restart a simulation run. Finally, the force applicator callback was added to assess dynamic stability.

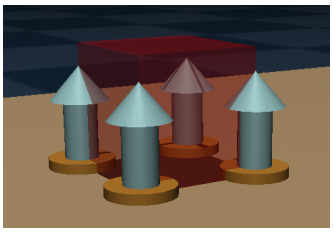

Fig. 4: Rigid body with four contact points and forces, respectively. Arrow lengths correlate to the force magnitudes.

C. Evaluators

For the purpose of assessing a packing plan's stability, the simulation core automatically triggers to run the evaluators after the simulation has terminated. These results form the basis for evaluating (a) simulation validity and (b) packing plan stability.

In order to decide whether a simulation run was valid, we used the acceleration evaluator's result. Internally, one threshold marked the upper bound of the maximal norm of a body's linear acceleration. Analogously, another threshold existed for the angular acceleration. If the norm of any recorded acceleration value exceeded its corresponding threshold, the simulation run was marked as invalid and we discarded the recorded quantities. Whenever we evaluated the dynamic stability, we disregarded the acceleration evaluator since the applied forces might cause an exceeding of the thresholds which did not indicate an invalid simulation run.

The pose delta evaluator's result was used to assess whether a packing plan is stable. We compared the position delta of a body's target pose, i.e., the position according to the packing plan, with its final position in the simulation run. In addition, we evaluated the tilt of each body, which was described by the angle of a body's normal vector with the xy-plane. Due to fact that some bin packing algorithms respect height tolerances, an internal definition of thresholds might lead to too strict values. Thus, the thresholds of the maximal position delta and the maximal tilt were parametrizable. If any value exceeded a threshold, the packing plan was categorized as unstable.

For the sake of detailed investigations, we implemented an evaluator that identified when a box fell off the pallet. Based on the recorded poses, this evaluator detected if (a) a body's center of mass was below the top side of the carrier or (b) the bottom side of the 3d bounding box that contains the oriented body touched the floor.

Finally, our novel evaluator computed the maximal force that acted on top of a body. The value was calculated by summing all values of top contacts' forces and taking the maximum over simulation time. Since it was impossible

to accurately reproduce the firmness of all boxes, we used this value to identify if in any simulation step a box bore too much weight. In order to analyze whether a box is overloaded, one needs to know how much load a box is able to carry in reality.

V. EVALUATING REALISM

Based on the simulation of three packing plans, which contained various package types, such as boxes, bottles, and cans, we evaluated the realism of our packing evaluation tool `PET-Physics`. After sequentially placing the items in the simulation run, we visually compared the virtual pallet with its corresponding real pallet. We selected this qualitative evaluation method because it is highly feasible, and we believe that identical scenes are crucial to build trust in our tool. Since we did not have any recording of a packing plan that led to a collapsed pallet, we limited ourselves to only use packing plans that are statically stable.

Notably, the comparison in Fig. 5 shows that cartons with sufficient firmness could be modeled as rigid bodies without losing accuracy. In contrast, we have seen that the modeling of items as rigid bodies was not capable of representing box deformation under high-magnitude loads. This occurred dramatically if the pressure was too high, e.g., along the tape sealing of a carton's top flaps.

A. Packing Plan with Small Deviation in Execution

The first pallets that we compared are visualized in Fig. 5a. The underlying packing plan contained 40 items, which separated into cardboard boxes, cans with a carton bottom wrapped with plastics, and bottles that were wrapped in plastics. In total, these items had a weight of 331.49 kg.

The main difference, evidently, was visible by a tilt of the black box on the real pallet. A nonoptimal placement of the box below caused the tilt. Due to the robot's placement strategy – move from an initial target position to the bottom right reaching the actual target position – the box below slightly overlapped with a box to the right and consequently, was tilted. This error propagated, causing the real black box to tilt further left than simulated.

We conclude that the difference in the pallets was independent of the items' simplified representation as rigid bodies, but resulted from the robot's placement strategy. In fact, if the box to the right of the tilted box is marginally higher, the robot's shift will not cause the observed behavior and the real and simulated pallet would look identical.

B. Packing Plan with Solid Items

This packing plan contained 41 items, which had a total weight of 358.5 kg. A photo of the built pallet and a picture of the simulated pallet are visualized in Fig. 5b. The products in this packing plan comprised 21 cartons, eleven shrink-wrapped bottles, four shrink-wrapped cans, two shrink-wrapped mini milk bottles, one shrink-wrapped ketchup bottles, one shrink-wrapped mason jars, and one carton with no top side, all featuring a cardboard bottom except for the shrink-wrapped bottles.

During palletization, we did not observe any issues. We are convinced that this was due to the firmness of the used items. However, we would expect problems if the bottles are not placed directly on top of each other.

C. Packing Plan with Manual Interventions Due to Robotic Placement

Last but not least, the basis of this comparison was a packing plan consisting of 41 items with a total weight of 332.77 kg. As depicted in Fig. 5c, the foundation of this pallet was formed by cartons only. Two manual interventions were required due to robot release issues, despite the pallets being nearly identical.

In general, the robot picked up a box and moved it with two fork tines close to its target position. Once it reached the target position area, it pushed the box to the front of the fork tine, followed by moving the tines back such that the back side of the box was released first and fell. Consequently, if the area below the box was not flat, it might be that the back side of the box fell further down than the front side.

The first intervention was required to correctly place the box on the right hand side, directly at the pallet. Due to the robot's release process, the edge that released first, fell into the gap between the top leadboard and the top deckboard. Even though this box remained stable, the item on top of it slipped and fell off the pallet. We manually fixed this and misaligned the shrink wrapped glass bottles.

The second intervention is not visible in the image. Again, the way how a box was released caused this correction. In order to continue robotic palletizing, we needed to push one box behind the shrink wrapped purple-colored bottles further to the back of the pallet. We believe that this push would have be done by the robot when it approached the target position of the earlier mentioned bottles.

A phenomenon that our physics simulation failed to capture was the tilt of the water bottles on top of the box in the middle of the scene. Since we modeled the items as rigid bodies, a load that acted on a body did not affect its shape. Hence, the body that represented the box under the water bottles kept its shape in simulation whereas in reality the bottles pushed in the top side of the carton.

VI. CONCLUSIONS AND OUTLOOK

In this paper, we presented our packing evaluation tool `PET-Physics`. It was not limited to assess the static stability of a packing plan, but our tool was applicable to check for dynamic stability. In order to evaluate stability, we made use of the pose delta evaluator's result.

A novel feature of our tool is the possibility to compute the maximum force that acted on top of a body during palletization. As a result, the tool provides a data basis to decide whether a box might collapse due to overloading. Moreover, we demonstrated the simulation's realism by visually comparing three packing plans. Even though the virtualized pallet differed at some places from its real counterpart, we are convinced that this is mainly due to the

(a) Packing plan V-A (b) Packing plan V-B (c) Packing plan V-C

Fig. 5: Three packing plans containing various items with that we evaluated the realism of `PET-Physics`.

missing of an accurate mirroring of the way how our robot places the items.

To further improve realism of `PET-Physics`, we plan to model items as soft bodies. Since this results in a huge number of additional contacts in simulation, it is essential to use a GPU. Another important aspect is replicating the robot's pick-and-place process.

In order to validate our hypothesis and to minimize the Sim-to-Real gap, we need to compare the simulation with reality more often. In doing so, it is not enough to visually compare the palletizing, but also to include sensors to verify that the quantities, e.g., contact forces, match reality. Finally, after reaching a tolerable Sim-to-Real gap, this framework is utilized to train a solver that only produces stable packing plans.

ACKNOWLEDGMENT

The authors like to thank the colleagues of TGW Logistics for the helpful discussions and the use of the palletizing robot and conveyor system.

REFERENCES

[1] E. Todorov, T. Erez, and Y. Tassa. "MuJoCo: A physics engine for model-based control". In: *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*. 2012, pp. 5026–5033. DOI: 10.1109/IROS.2012.6386109.

[2] E. E. Bischoff. "Stability aspects of pallet loading". In: *Operations-Research-Spektrum* 13.4 (Dec. 1991), pp. 189–197. ISSN: 1436-6304. DOI: 10.1007/BF01719394.

[3] S. Martello, D. Pisinger, and D. Vigo. "The Three-Dimensional Bin Packing Problem". In: *Operations Research* 48.2 (2000), pp. 256–267. DOI: 10.1287/opre.48.2.256.12386.

[4] A. Bortfeldt and G. Wäscher. "Constraints in container loading – A state-of-the-art review". In: *European Journal of Operational Research* 229.1 (2013), pp. 1–20. ISSN: 0377-2217. DOI: 10.1016/j.ejor.2012.12.006.

[5] L. Junqueira, R. Morabito, and D. Yamashita. "Three-dimensional container loading models with cargo stability and load bearing constraints". In: *Computers & Operations Research* 39.1 (2012). Special Issue on Knapsack Problems and Applications, pp. 74–85. ISSN: 0305-0548. DOI: 10.1016/j.cor.2010.07.017.

[6] C. Paquay, M. Schyns, and S. Limbourg. "A mixed integer programming formulation for the three-dimensional bin packing problem deriving from an air cargo application". In: *International Transactions in Operational Research* 23.1-2 (2016), pp. 187–213. DOI: 10.1111/itor.12111.

[7] S. Tkaczyk, M. Drozd, Ł. Kędzierski, and K. Santarek. "Study of the Stability of Palletized Cargo by Dynamic Test Method Performed on Laboratory Test Bench". In: *Sensors* 21.15 (2021). ISSN: 1424-8220. DOI: 10.3390/s21155129.

[8] D. Roy, P. Vyas, and J. Cherian. "How Stable is a Pallet? A Framework for Analyzing the Dynamic Stability of Stacked Pallets". In: *2024 IEEE 20th International Conference on Automation Science and Engineering (CASE)*. 2024, pp. 3622–3627. DOI: 10.1109/CASE59546.2024.10711551.

[9] F. Kagerer, M. Beinhofer, S. Stricker, and A. Nüchter. "BED-BPP: Benchmarking dataset for robotic bin packing problems". In: *The International Journal of Robotics Research* 42.11 (2023), pp. 1007–1014. DOI: 10.1177/02783649231193048.

[10] Z. Wang, H. Zhao, J. Xu, S. Zhang, Z. Xiong, R. Hu, C. Zhu, Z. Zeng, and K. Xu. *RoboBPP: Benchmarking Robotic Online Bin Packing with Physics-based Simulation*. 2026. arXiv: 2512.04415 [cs.RO].

[11] E. Coumans and Y. Bai. *Pybullet, a python module for physics simulation for games, robotics and machine learning*. 2016.

[12] S.-Y. Chien and C.-C. Wong. "Online 3D Bin Packing for Novel Objects Based on Deep Reinforcement Learning". In: *2023 International Conference on Advanced Robotics and Intelligent Systems (ARIS)*. 2023, pp. 1–4. DOI: 10.1109/ARIS59192.2023.10268559.

[13] P. G. Mazur, J. W. Melsbach, and D. Schoder. "Physical question, virtual answer: Optimized real-time physical simulations and physics-informed learning approaches for cargo loading stability". In: *Operations Research Perspectives* 14 (2025), p. 100329. ISSN: 2214-7160. DOI: 10.1016/j.orp.2025.100329.