Robotics and Computer–Integrated Manufacturing 71 (2021) 102140

Contents lists available at ScienceDirect

Robotics and Computer-Integrated Manufacturing

journal homepage: www.elsevier.com/locate/rcim

Generic development methodology for flexible robotic pick-and-place workcells based on Digital Twin

Bence Tipary [a,b,*], Gábor Erdős [a,b]

[a] Institute for Computer Science and Control (SZTAKI), Eötvös Loránd Research Network (ELKH), Budapest, Hungary
[b] Department of Manufacturing Science and Engineering, Budapest University of Technology and Economics, Budapest, Hungary

ARTICLE INFO

Keywords:
Digital twin
Modeling
Flexibility
Robot
Calibration
Planning

ABSTRACT

Together with the trends of mass personalization, flexible robotic applications become more and more popular. Although conventional robotic automation of workpiece manipulation seems to be solved, advanced tasks still need great amount of effort to be reached. In most cases, on-site robot programming methods, which are intuitive and easy to use, are not applicable in flexible scenarios. On the other hand, the application of offline programming methods requires careful modeling and planning. Consequently, this paper proposes a generalized development methodology for flexible robotic pick-and-place workcells, in order to provide guidance and thus facilitate the development process. The methodology is based on the Digital Twin (DT) concept, which allows the iterative refinement of the workcell both in the digital and in the physical space. The goal is to speed up the overall commissioning (or reconfiguration) process and reduce the amount of work in the physical workcell. This can be achieved by digitizing and automating the development, and maintaining sufficient twin closeness. With that, the operation of the digital model can be accurately realized in the physical workcell. The methodology is presented through a semi-structured pick-and-place task, realized in an experimental robotic workcell together with a reconfiguration scenario.

1. Introduction and problem statement

Although robotic applications have been widespread for decades, their development process is hardly generalized. Interactive and on-site robot programming methods receive great attention [1], however, these are only applicable in relatively simple tasks. For more complex scenarios, simulation-based offline programming methods are much more suitable. Therefore, offline programming is getting more and more popular due to its potential to reduce the on-site workload and speed up the commissioning phase. On the other hand, offline programming is often performed inaccurately, needing tedious adjustments either in the physical workcell or in the program code [2]. Ultimately, more complex solutions still need significant offline and online engineering effort [3].

In flexible robotic solutions, the lack of sufficiently precise information makes cell development even more complicated. Imprecise information can occur in different forms, such as picking or placing poses, workpiece shape, etc. One of the most prominent examples is the standard bin-picking scenario [4], where workpieces lay randomly in a bin, meaning that their precise pose (position and orientation) is originally not available. Such cases are mostly tackled using sensors and metrology, to resolve the information, which is not known explicitly in the design phase. As these measurements and the corresponding computations need to be carried out online (i.e., in the physical workcell) and autonomously, they cannot be completely planned beforehand in an offline manner (i.e., in a virtual representation). This increases the commissioning time further, as the corresponding development also needs to be carried out at least partly in the physical cell. The above concept can also be extended to a subset of agile robotic systems, where unexpected (error) events result in similar uncertainties (e.g., handling misoriented workpieces on a pallet or incorrect workpiece types).

The lifecycle of a robotic workcell consists of several steps, starting from initial design, through process planning and commissioning to operation and retrofit. The workcell development process includes every phase before operation, and involves a number of corresponding experts, including designers, process planners, manufacturers etc. Traditionally, the phases of workcell development are managed separately, by different personnel. Since each phase builds on top of the previously achieved results, the communication and information flow between the experts is of key importance [5]. On the other hand, concurrent engineering aims to parallelize phases, and emphasizes feedback. This

* Corresponding author.
E-mail addresses: bence.tipary@sztaki.hu (B. Tipary), gabor.erdos@sztaki.hu (G. Erdős).

https://doi.org/10.1016/j.rcim.2021.102140
Received 4 August 2020; Received in revised form 17 February 2021; Accepted 17 February 2021
Available online 13 March 2021
0736-5845/© 2021 The Authors. Published by Elsevier Ltd. This is an open access article under the CC BY-NC-ND license (http://creativecommons.org/licenses/by-nc-nd/4.0/).

methodology is already present in the case of factory and product design and development. However, individual cell operations with detailed sub-processes are seldom considered.

There exist many planning tools for the different aspects of robotic applications, which aim to tackle the different workcell lifecycle phases. Yet, they do not provide sufficient feedback supporting features and their interoperability is limited. Also, no comprehensive methodology exists for complete cell development. Hence, in many cases, the implementation of robotic workcells becomes a sequence of unthoughtful tweaks and adjustments, because the development process is not well selected or it is based on an ill-defined scenario.

Meanwhile, enabled by the industrial digitalization, the concept of Digital Twin (DT) receives significant attention [6,7]. By mimicking the physical world, the DT is a virtual representation of a physical product or system. The DT contains a number of (relevant) models, which allow simulation, optimization, prediction, observation and control of the physical twin [8,9].

Utilizing the concept of DT, this paper proposes a methodology to generalize, facilitate and guide the development process of robotic workcells from design to commissioning, and also during adjustment, maintenance and reconfiguration. The methodology aims to organize the development and verification steps with the help of the robotic workcell DT. The proposed iterative development process, during which the digital and physical workcell are systematically verified, maintains the feasibility of the prepared system. Additionally, to accurately realize the operation in the physical workcell, geometry and tolerance-related deviation between the digital and physical counterparts is also controlled in each iteration. To capture this deviation, the term **twin closeness** is defined.

The focus of the proposed methodology are robotic manipulation tasks, specifically conventional and flexible industrial pick-and-place operations. For such scenarios, as well as for their development process, a generalized description is provided, together with the relevant models, data and applicable tools. The models are presented in an object-oriented form, to lay down a general, data- and function-based description. The methodology covers basic and advanced techniques necessary for most industrial pick-and-place applications. These are organized into functional blocks: kinematic, grasp, path, servo, metrology and tolerance models. The proposed approach can be especially advantageous in case of the original design or re-design of flexible workcells, frequently reconfigured workcells, or in advanced scenarios, where offline robot programming and optimization is required.

The structure of the paper is the following. The state of the art is presented in Section 2. The basic considerations and proposed methodology are outlined in Section 3, the generalized pick-and-place process is described in Section 4 and Section 5 gives a detailed description on the DT functionalities, which provide guidance along the modeling and planning phases of the development process. Section 6 presents the implementation examples and conclusions are drawn in Section 7.

2. State of the art

2.1. Flexible robotic pick-and-place applications

Flexible applications can occur on different levels and in different scenarios. The ones that receive greater interest are the factory and warehouse scenarios. In factory scenarios the focus is on precision and reduction of the cycle time [10]. There are countless different solutions depending on the type of uncertainty during the operation. Chen et al. presented an approach for manipulating workpieces arriving on a conveyor belt with unknown locations [11]. In [12], positioning is realized with servo (visual and force) techniques to achieve precise placing. Different semi-structured manipulation tasks, where the workpieces lay separated in one of their stable poses, are presented in [3,13,14]. Solutions involving unstructured workpieces laying in a bin are tackled by [10,15,16].

In warehouse scenarios, solutions mainly target the grasping of unknown objects using conventional and machine learning techniques [17,18]. In this case, the main focus is on improving grasping reliability, with less attention to precise picking and placing. Many such approaches are developed for different Amazon Robotics Challenges [17,19], where everyday objects with initially unknown geometries need to be identified, grasped and manipulated from a mixed bin. Further solutions for everyday objects include grasping without object identification [20], picking from conveyor [18], and stable placing onto different workholders [21].

Furthermore, other niche scenarios exist. For example, manipulating organic objects, such as fruits or bags in agriculture, provide further uncertainties to be overcome by flexible manipulations [22]. More on the agility side, Downs et al. [23] presented the evolution of Agile Robotics for Industrial Automation Competition and corresponding agility and performance metrics. The challenges include changes in the orders during its execution, handling of workpieces in unexpected poses and handling of faulty workpieces, among others. Similar error cases are tackled in a pick-and-place application on a skill-basis in [24]. The established a mobile robot system is capable of handling error scenarios caused by dynamic environment or incorrectly presumed information about the operation, components or workpieces.

2.2. Robotic application development tools

Clearly, flexible robotic applications can be extremely diverse even when limited to pick-and-place operations. Existing tools aim to provide a framework to support the development process of such applications. Robot Operating System (ROS) [25] was introduced to support modular, tool-based software development for robotic applications. ROS quickly became popular and developed a great community. It is now commonly used in numerous robotic applications and received industrial support as well. Furthermore, a planning environment called OpenRAVE (Open Robotics Automation Virtual Environment) was developed by Diankov [26], for implementing planning algorithms for robotic applications. Other modeling tools also exist, such as Modelica [27] or Matlab/SIMULINK together with a robotic toolbox [28]. These are also applicable for development of control and planning algorithms. Further, more specific tools are listed later, together with the development phase in which they are applicable. Although the above tools provide extensive support, they do not provide methodological guidance or workflow for how a flexible pick-and-place workcell or operation is to be developed.

2.3. Robotic workcell development

Considering the generalization of the robotic workcell, as well as robotic task development, only a few relevant works were found in the literature. Kak et al. [29] presented a knowledge-based robotic assembly cell concept, including a knowledge base, supervisor, sensor system, motion controller and the environment model. This representation aims to describe flexible manipulation tasks, however, it is not fully generalized and incomplete in terms of development phases. A database design was presented by [30], which captures the basic description of robotic workcells, but flexible operation is not considered. Furthermore, a robotic task representation and ontology framework is proposed in [31], however, it does not provide a development methodology.

In the field of assembly planning, a function block-based methodology was presented by Wang et al. [32] for improving adaptability and flexibility of assembly systems. The function blocks are applied on the system control level to facilitate and speed up the reconfiguration process. Also, for the design of generic flexible assembly systems, Edmondson and Redford [33] overviewed and summarized the key components and considerations. To the best of the authors' knowledge, there is no generalized development methodology for flexible robotic pick-and-place workcells.

3. Methodology overview

3.1. Basic considerations

In order to provide a consistent methodology description, the following setting is considered:

In case of conventional robotic pick-and-place approaches, if the physical workcell is already established, the operation can be prepared using **online programming** [34]. The result is a program code, which contains mostly robot motions and gripping actions and can be executed in the workcell. This method is usually sufficiently accurate, as the key points are set up in the physical workcell w.r.t. the physical artefacts. Online programming can also be considered as a process planning method, and it is a common practice in case of simple systems performing simple operations.

Alternatively, a similar program code can be created offline, if every information of the workcell design and the operation is available, even if the workcell is not yet physically built. In this case, **offline programming** can be realized based on the digital robotic cell. The program code can be created using model-based **offline planner tools** (e.g., path planner) and/or standard CAD (Computer Aided Design) techniques, depending on the cell complexity. This approach is advantageous in case of more intricate workcells and operations [34]. However, such a program code can only be executed successfully, without any refinements, if the characteristics of the designed and built workcell match sufficiently (discussed in Section 3.4 as twin closeness). Otherwise, after the physical system is created, either the program code needs to be compensated with online programming, or rather, the digital and physical workcell need to be calibrated to revise the offline programming. In this scenario, the initial design information is considered **calibratable**, as the corresponding uncertainties can be overcome by calibration [35].

In case of flexible or advanced pick-and-place, there is a lack of information in the workcell design and/or the operation. For example, in a bin-picking scenario, even if the workcell is calibrated, the tolerance of the workpiece pose depends on the size of the bin, which in itself is intractable for operation. Similar uncertainties can be present in case of such agile applications, where the presumed information about the components or operation becomes faulty when errors occur (e.g., a workpiece lays in the wrong pose in a pallet). Such **task-specific uncertainties** cannot be resolved by means of calibration or manual compensation as in the conventional case, since they can change cycle by cycle. Therefore, in a sense, the corresponding program code also needs to be updated cycle by cycle, meaning that metrology (observation)-based **online planner tools** are required for the operation, to overcome these task-specific uncertainties. However, since online planning time directly affects the cycle time, the amount of online computations needs to be minimized. The program code can be prepared partially using offline planner tools (if the workcell is calibrated) and partially by implementing online planner tools. While offline planner tools are prepared offline and are often manually operated, online planner tools are usually prepared partly offline, partly online and are mostly automated. Most of the time, these tools need to be prepared by integrating and adapting existing solutions. In extreme cases, purpose-built tools might also be necessary.

3.2. Digital Twin

The basis of the development methodology is the Digital Twin (DT) concept [36]. By containing the relevant data and specialized simulation models in an organized way, DTs are often applied along the whole lifecycle of real systems [37]. There are existing DT-based approaches for different phases of manufacturing and assembly systems, including commissioning [38], maintenance [39] and also reconfiguration [40, 41]. Furthermore, Melesse et al. presented a review of robotic workcell DTs in [42].

In the present case, the DT is the digital representation of a complete pick-and-place scenario, which can be utilized in the development, operation and service periods of the corresponding pick-and-place workcell. Here, only the phases from design to commissioning are presented in full depth, however, the DT is also applicable in a similar fashion in later phases, e.g., in case of reconfigurations, adjustments, or accidental misalignments happening in the workcell.

The general purpose of the proposed methodology is to facilitate the development steps and provide a systematic workflow for realizing different pick-and-place tasks. Correspondingly, the underlying DT needs to be able to support a variety of scenarios. During the development process, the main purpose of the DT is to allow the preparation of offline and online planner tools. Furthermore, the DT needs to provide modification and calibration capabilities for the digital model. Consequently, the DT contains a number of models, which enable offline and online planning, simulation and preparation.

The basis of the DT is the **kinematic model**, which describes the geometry, kinematic behavior and component relations of the cell. The **grasp model** is responsible for describing the relation of the workpiece and the manipulator through the gripper. The **path model** describes the manipulation sequence, as well as the manipulator motion throughout the process. To overcome task-specific uncertainties, two observation-based models are considered, the servo and the metrology models. As a sub-part of the path model, the **servo model** is used to describe condition-based manipulator motion strategies. On the other hand, **metrology model** is responsible for obtaining and providing information to other models and planner tools. Lastly, the **tolerance model** constructs and evaluates the tolerance stack-up of the operation, to assess its geometric feasibility.

As the DT needs to allow continuous improvement and refinement, without losing earlier preparation results, the model definitions need to be parametric and updatable. By using parametric representations, planning steps or evaluations can be easily recalculated by changing the input parameters. Moreover, the simulation capabilities, along with the parametric representation, allow workcell optimization in different aspects, such as cycle time or energy consumption.

3.3. Assumptions

While the present methodology is generic, there are basic assumptions that need to be considered. The conventional communication tool of designers are drawings or CAD models, using nominal geometries, tolerances and supplementary information. Larger part of the design intent is not forwarded explicitly, but encoded into toleranced geometry, which is tangible for the other experts.

In the proposed methodology, the above approach is extended to the modeling phase as well, i.e., models other than the ones introduced in Section 3.2 are not dealt with separately, but are considered on the level of tolerances. These include effects, which are difficult to model or difficult to model accurately, such as frictional, thermal or dynamic effects. Instead of generalizing their complex models, these are considered in the geometric and dimensional tolerances. Using rigid bodies and a quasi-static operation, together with sufficiently strict tolerance related feasibility criteria, the above effects can be tackled up to a certain extent, which is sufficient in the majority of scenarios. In case of applications where these effects are dominant, their proper modeling may be recommended. Accordingly, the DT can be extended with the corresponding specific models and planner tools. However, these are not in the scope of this paper.

Furthermore, the model does not expect an a priori feasible system design, feasibility is assessed in multiple stages, and the design is iteratively improved until feasible operation is reached. The nature of the development is inherently iterative, as even in case of the models included in the DT, not all artefacts and environmental factors can be precisely modeled, such as the manipulator cabling or environmental effects in case of metrology. These can be resolved on an experimental basis via engineering interactions. After each interaction, the system

needs to be verified. However, if feasibility is not reached, and the system cannot be further improved in a rational way in terms of labor and cost, the development process will terminate with a failure.

3.4. Twin closeness

In order to have offline planning and simulation results applicable for the physical system, the deviation between the digital and physical workcell characteristics (geometry, behavior, etc.) needs to be within acceptable limits. In terms of geometry, this means that the digital and physical counterparts need to be within a feasible tolerance region (bound by the feasibility criteria). To represent these characteristics, the term of **twin closeness** is defined here (see Fig. 1). Twin closeness is based on a deviation function between the digital and physical system counterparts, defined on a geometric tolerance basis. The deviation function is determined by the technology (e.g., robot control, metrology or gripper) and the implemented artefacts (e.g., geometric features) of the workcell. This function includes the geometrical deviation of the objects in the cell, as well as the trajectories of the dynamic objects, such as the deviation between the designed and realized robot tool path. Twin closeness can be assessed by evaluating the deviation function for each relevant workcell scene (i.e., the actual workpiece and component arrangement in the workcell) for each corresponding artefact, and comparing the results to the corresponding feasibility criteria.

The **deviation function** is defined as a general distance function with two arguments. The first argument corresponds to the relevant workcell scenes, which can usually be described with the robot configuration along the robot trajectory or in a given region of interest of the pick-and-place operation. This specifies the deviation of an artefact from the point of view of the operation, on the basis of operational phase (e.g., when seizing or releasing the workpiece). The second argument corresponds to the artefacts (e.g., frames) of the workcell. This specifies the deviation of an artefact on a spatial basis (e.g., for workpiece pose or robot configuration).

The deviation function returns a vector of geometric intervals for the general distances between the specified artefact of the digital model and the actual physical system. With the overestimations of the tolerances, returned values are considered valid under boundary conditions posed by effects not modeled, such as ambient temperature range in the workcell, or acceleration range of the robot joints (described in Section 3.3).

Each feasibility criterion provides a lower and upper bound for the corresponding deviation. These need to be defined in the task specification for each relevant deviation for the twin closeness assessment. The feasible tolerance region is bound by the feasibility criteria, within which the operation is feasible and considered accurate.

Given the feasibility bounds, the twin closeness is assessed by evaluating a set of inequalities. Each element of the returned vectors from the deviation function (for each relevant case) is checked for the corresponding feasibility criteria. If all elements are within the defined boundaries, the twin closeness is concluded sufficient. The boundaries here define a **hyperrectangle-shaped feasible tolerance region**.

For example, when a cylindrical workpiece is manipulated by a Cartesian coordinate robot, assuming planar manipulation, the deviation function returns a vector with two interval elements (corresponding to the deviations in the plane, along the axes of the Cartesian coordinate system) for the workpiece centerpoint in any workcell scene. These intervals represent a rectangular region within which the physical workpiece centerpoint lays in the given scene. For each direction, the feasibility criteria (lower and upper bounds) are given. Therefore, if the returned rectangle region is within the rectangle region defined by the feasibility criteria, the closeness in this particular scene for this particular artefact is sufficient. The twin closeness also shows the extent of fault tolerance in terms of precision, i.e., how much deviation is allowed beyond the normal, feasible operation when errors occur in the component or workpiece poses.

In more complicated scenarios, the returned vector can contain mixed deviation types, (e.g., for cylindrical coordinate systems, homogenous transformations, robot joint coordinates, etc.), where the returned region becomes abstract. However, in general, the feasible tolerance region can be of arbitrary shape, where the feasibility criteria are actually interdependent (e.g., when a cylindrical workpiece is placed into a larger, 'U' shaped slot). In such cases, this approach can be utilized in a conservative way, by inscribing a hyperrectangle-shaped region. If this is not sufficient, twin closeness can be assessed by using partially analytic tolerance analysis methods, along with testing of the physical workcell operation.

Models inherently simplify the reality, to facilitate their utilization. Therefore, neglecting irrelevant artefacts is encouraged. However, missing key artefacts from the deviation function and the feasibility criteria might result in insufficient twin closeness, as neglecting sources of inaccuracy can render the operation in the physical workcell infeasible. The better the twin closeness, the more precise and refined is the digital representation of the system.

The returned intervals of the deviation function are determined based on the design specifications (e.g., tolerance values in the CAD model or on the drawings), as well as measurements on the digital and physical system and its components. Consequently, twin closeness can be improved via DT calibration in general. This can be realized by parameter adjustment on the model side (see Fig. 1(a)), or by adjusting the physical workcell (see Fig. 1(b)). Additionally, the corresponding tolerance region can be extended by tolerance enhancing techniques (such as visual, tactile or force servo control of robots), equipment improvement (e.g., more precise metrology system or manipulator), or simple geometric features, like chamfering (see Fig. 1(c)).

Other types of deviations (such as missing objects in the digital workcell model or non-geometric deviations), are not considered by twin closeness, yet still need to be regarded through feasibility assessment. These deviations are referred to as **qualitative deviations**. When infeasibilities are caused by these, design refinement is necessary, which again, can be realized both in the digital or physical spaces.

Twin closeness needs to be assessed on multiple occasions after the

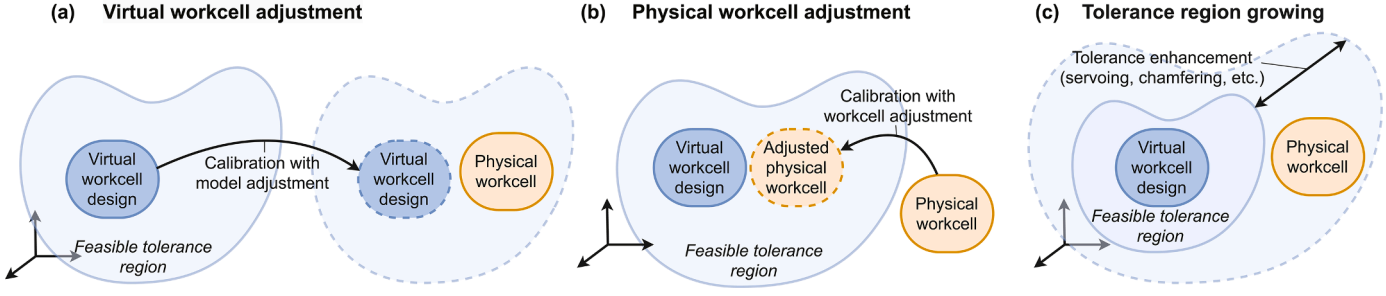

Fig. 1. Digital Twin closeness and deviation reducing methods, such as calibration-based adjustment of the virtual workcell model (a), of the physical workcell (b), and growing the tolerance region (c).

physical workcell is set up. Sufficient closeness ensures the proper operation of the physical counterpart of the system after complete implementation. In these cases, if the program code corresponding to the operation of the digital model is executed in the physical workcell, the physical operation shall match the digital one accurately.

3.5. Overview of the development process

The development process takes part in the first half of the workcell lifecycle, ending between the commissioning and the operation phase. The particular phases of this proposed development methodology are shown in Fig. 2. The development process starts with an initial design concept, containing the conceptual description of the workcell design and the operation, together with the list of requirements. This information is utilized in the design phase. Here, having the basic concept, the task of the workcell is specified and the cell components are selected. Meanwhile, the CAD model of the workcell is being prepared, together with the necessary drawings. All relevant data (hereinafter referred to as design specification) is collected, organized and pre-processed for the modeling phase.

As the prepared input data becomes available, the modeling phase is started, where the relevant DT models are established. The input data and the modeling process are described in detail in Section 5. After preparation, these models can be applied in the offline planning step to realize offline simulations and (pre-)computations (as well as later, in the online planning step). During offline planning, usually a partial program code can also be generated and post-processed.

The next step is DT verification, where the relevant computation results (workcell layout, robot path plan, etc.) are evaluated for feasibility, including the fulfilment of the task requirements. In case of any occurring infeasibility, the design is checked for possible modifications. If the design is improvable in this way, the design specification is adjusted accordingly (this is referred to as refinement), and the offline phases are revisited until feasibility is achieved. Otherwise, if the design is not improvable further in a reasonable way (e.g., without excessive investment costs or substantial modifications in the workcell design), the planning process terminates with a fail. The verification steps are further detailed in Section 5.7.

The next step is the physical implementation. Up to this point, the development steps were performed offline, using the virtual model. However, development cannot proceed further in an offline manner due to the presence of task-specific uncertainties. In the following, the implemented physical system (together with the partial program code) needs to be checked in an online verification step. First, the overall feasibility of the (partial) workcell and operation is checked, then the twin closeness. Refinement and calibration are performed if necessary. These modifications can be realized in the virtual and/or in the physical workcell. In case the modifications are applied on the model, the previous offline phases are revisited, utilizing the newly gathered information about the physical workcell. If solely physical modifications are applied, only the verification step needs to be repeated.

After sufficient twin closeness is achieved, the next step is metrology. Here, the necessary information is collected using the sensors in the physical system to resolve the task-specific uncertainties. Then, with the resolved data, the initial models are updated. It is noted that although the update is realized within the DT, this step is performed online, as the measurements are taken in the physical workcell. Next, based on the prepared DT models, online planner tools and the measurement results, the autonomous online planning is realized in the physical workcell. In this phase, the task-specific plans are created for the actual scene, which are then post-processed to an executable program code.

Afterwards, the complete physical implementation (including the program code) is assessed for feasibility and twin closeness in a final verification step. It is noted that, depending on the task, this test should be sufficient to ensure feasible operation under any defined circumstances (i.e., not only in one particular scene of the flexible operation). This final feasibility assessment is carried out similarly as the previous online verification step. If the system is successfully verified, the program codes are executable and lastly, the commissioning step can be performed. Here, the program codes and planner modules are finalized on the cell controller, and the workcell is set ready for operation. With the commissioning being ready, the development process is finished.

4. Generalized pick-and-place operation

4.1. Pick-and-place workcell

First of all, the basic components of a flexible pick-and-place system are summarized. Such a workcell consists of a **manipulator**, and an **end-effector** mounted on its last link (e.g., vacuum or finger gripper). Furthermore, there are **workholding units** at picking and at placing locations (in short, picking/placing workholders), which are capable of restricting or limiting part motion in certain directions (e.g., fixture or workpiece pallet). To overcome task-specific uncertainties, a **metrology system** is present, which consists of sensors or sensor networks. The most common types of sensors are: touch, force, vision and proximity. And lastly, there is the object to be manipulated, namely the **workpiece**. For the sake of simplicity, manipulators are referred to as **robots**, and end-effectors as **grippers**.

In order to fully comprehend pick-and-place operations, the basic sequence of this, arguably simplest assembly task needs to be dissected and generalized. Both the picking and the placing sub-process can be described as a sequence of a manipulator movement and a gripper setting (or activating and deactivating) steps.

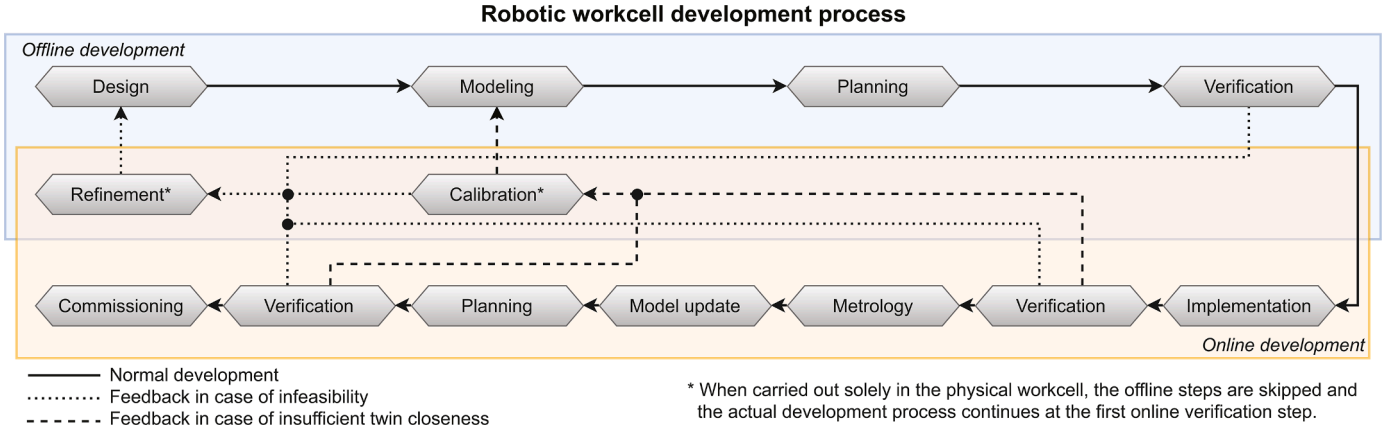

Fig. 2. Development process steps for flexible pick-and-place applications.

For clarification, poses and frames are defined in the robot task space as task space points, however, a pose corresponds to the reference of a rigid body (only one per rigid body), whereas frames correspond to a constant transformation w.r.t. rigid body references (can be multiple per rigid body). Also, configurations are defined in the robot configuration space as a robot joint vector.

The notable robot configurations (path segment starting and end-points) in the sequence of the pick-and-place operation are as follows (see an example scenario in Fig. 3):

(1) **initial robot configuration**
(2) **metrology configuration(s)**
(3) **seizing approach configuration**
(4) **seizing configuration** – before the gripper setting for workpiece seizing (a) and after (b)
(5) **seizing retreat configuration**
(6) **releasing approach configuration**
(7) **releasing configuration** – before the gripper setting for workpiece release (a) and after (b)
(8) **releasing retreat configuration**
(9) **finishing robot configuration**
(*) **intermediate configuration(s)**

Approach and retreat configurations aim to envelope the more intricate seizing and releasing phase, where the physical connection of the workpiece is transferred between the workholders and the gripper. In this way, the problem complexity can be reduced, as path planning is divided to shorter, separated segments. Meanwhile, the overall offline plannable path length can be increased as well. Approach configurations can also be starting points for servoing, which could allow lower tolerance in determining the said configurations. Intermediate configurations have the same purpose. These can be defined between any two subsequent configurations to divide the path segments. For each above listed configuration, there is also a corresponding frame in the task space.

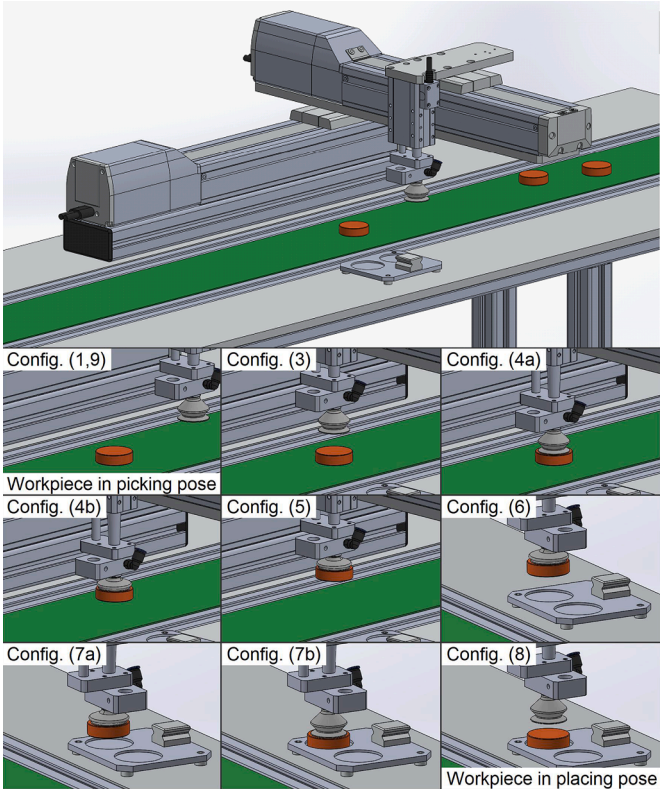

Fig. 3. Example pick-and-place scenario with basic configurations and workpiece poses, where the initial and finishing configurations coincide and there are no distinct metrology and intermediate configurations.

It is noted that these robot configurations are a priori identifiable, if, before seizing, the workpieces are static with precisely known poses or in motion (dynamic) with precisely predictable trajectories. Otherwise (e.g., in case of servoing), the configurations are only identifiable a posteriori.

Regarding the workpieces, there are two notable poses. A workpiece before seizing lays in the picking pose, and after releasing, it lays in the placing pose. Other notable frames are the grasp frames. Grasp frames are defined on the workpiece and on the gripper. If a workpiece is in a picking pose, and the robot is in the seizing configuration, the corresponding grasp frames are aligned and form a grasp frame pair (see Fig. 4).

Having prepared the key configurations, frames and poses, the pick-and-place operation can be described. It starts with a workpiece in picking pose and the manipulator in initial configuration, away from the seizing configuration. The gripper needs to be empty and set to be ready for seizing. Before starting the picking phase, if necessary, metrology is performed for which the manipulator is moved to the metrology configuration(s). Then, the picking phase is started. The first step in picking is positioning the manipulator to seizing approach configuration, then to seizing configuration. The next step is setting the gripper to seize the workpiece laying in picking pose. Once seized, the workpiece is manipulated away from the seizing frame into the seizing retreat frame, to reach a safety distance from the environment and other workpieces.

At this point, the picking phase ends and the placing phase can begin. Similarly to the picking phase, the first step in the placing phase is positioning the manipulator to the releasing approach configuration, then to releasing configuration, only this time with seized workpiece. Then, by setting the gripper to release, the workpiece is released into placing pose. Finally, the manipulator is positioned into releasing retreat configuration. Depending on whether or not additional pickable workpieces are available, the process is started over with the next workpiece, or the manipulator returns into the initial or metrology configuration to start a new cycle. It is noted that metrology and gripper setting can not only be realized in discrete configurations, but also continuously along a path segment. This is determined by the state machine described in Section 4.2.

The corresponding task-specific uncertainties, and thus workcell flexibility can occur in many forms. These include, among others, the different number of workpieces and workpiece types, their dynamic state, knowledge about their pose and geometry, the interaction capabilities with the workpieces, as well as the knowledge about the picking and placing poses and environment.

4.2. Generalized cell control

Similarly to numeric controlled machines, there are in general two types of commands during robotic cell operation, namely motion and switching commands. Motion commands are generally handled by the motion controller while switching commands are handled by the programmable logic controller (PLC). This setup is usually present in each

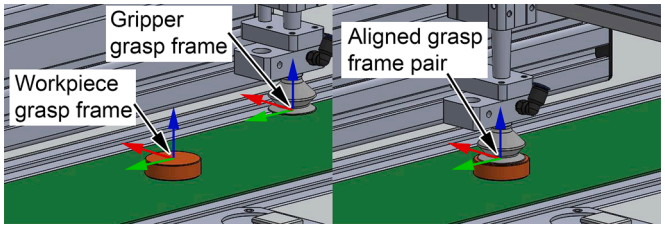

Fig. 4. Workpiece-gripper relationship with the grasp frame pair in case of seizing.

manipulator, however, most of the time another controller (PLC) is present on the cell level as well. The motion controller includes an interpolator, while the PLC realizes a state machine, which ensures the synchronized motion of the different motion controllers as well. As these two are connected, switching commands can be executed before, during or after a motion is executed. Encoding the mentioned commands, a main result of the development process is the corresponding program code.

Although the included commands are generalized, these need to be post-processed to become compatible with the cell controller(s). The switching commands and the corresponding state machines can be described using various standardized PLC programming languages [43]. Due to the standardization, the transformation of a pseudo-code to a PLC code is mostly straight-forward. On the other hand, handling of motion commands are entirely robot brand and type dependent. Motion commands may differ, e.g., in possible motion interpolation types or spaces, typically configuration or task space, in which the target is defined. Therefore, although the commands can be generalized on the planning side, significant post-processing might be necessary.

During the path planning process, multiple spaces can be utilized, depending on the particular scenario. For example, in case of 6 DoF (degrees of freedom) robot arms, planning linear motions is much more convenient in the robot task space. Not only is the calculation of the corresponding robot configuration interpolation simpler, but also is the whole workcell design (i.e., the CAD model and presumably the path endpoints), defined in the robot task space.

Ultimately, the planned paths need to be checked for being within the robot reach, obeying the joint limits and being collision-free. These on the other hand, need to be checked continuously (or densely sampled) along the paths, for which a continuous, unambiguous representation is much more suitable. For serial robots, reachability is assessed by checking if a real robot configuration exists within the joint limits for a task space point. For the evaluation of collisions, the manipulator and corresponding geometries need to be moved in the workcell environment along the path, which again, can only be realized if the path is known in the configuration space. Consequently, in general, both the forward and inverse kinematic models of the manipulator are necessary during planning and feasibility assessment. In addition, for feasibility assessment, a representation needs to be applied that describes the manipulator motion control unambiguously.

That being said, in this methodology, configuration space representation is chosen for general description. Motion commands are generalized as time parametric robot configuration interpolations from the actual robot configuration to the target configuration (referred to as path segments). Although explicitly not required, target configurations (i.e., path endpoints) are also regarded, as these and the interpolations are handled by separate models. The time parametrization encodes the joint velocities and accelerations which can be also transformed into the task space via post-processing, if necessary.

With the above considerations, the selected representation can be post-processed depending on the workcell equipment. It is noted that in a particular development scenario, the DT needs to mimic the controller behavior and the corresponding program code needs to be suitable for the physical controller for proper operation.

4.3. Pick-and-place specific commands

Considering the specification described above, the specific commands can be summarized as robot motion commands (*move_robot*) and switching commands (*set_gripper, measure, process_data* and *plan*). The former need to be supported with time parametrized robot configuration interpolation, which end with the target configuration. The switching commands need to be supported with parameter arguments. Gripper setting needs grasping-related parameters (e.g., gripping force), while measurement needs measurement parameters, and so on. In addition, most of the above parameters (or data necessary for their computation) can be considered task-specific depending on the scenario. To resolve these, the specification is summarized simply as measurement and data processing parameters, while any obtained task-specific data is represented as resolved data. Finally, the commands need to be organized into a state machine and the sequence, branching, cycles and corresponding conditions need to be defined depending on the particular scenario.

Pseudo-code 1
General pick and place operation example.

```
procedure pick_and_place_operation
  initialization
    while ∃ free placing workholder slot do
    move_robot (initial_robot_configuration)
    set_gripper (metrology_parameters)
    move_robot (metrology_configuration)
    set measured_data = measure (measurement_parameters)
    set resolved_data = process_data (measured_data, processing_parameters)
    plan (resolved_data)
    while ∃ pickable workpiece for a free placing workholder slot do
      set_gripper (seizing_approach_parameters)
      move_robot (seizing_approach_configuration)
      move_robot (seizing_configuration)
      set_gripper (seizing_parameters)
      move_robot (seizing_retreat_configuration)
      move_robot (intermediate_1_configuration)
      move_robot (releasing_approach_configuration)
      move_robot (releasing_configuration)
      set_gripper (releasing_parameters)
      move_robot (releasing_retreat_configuration)
      if ∃ pickable workpiece for a free placing workholder slot then
      move_robot (intermediate_2_configuration)
      end if
    end while
    refill and/or rearrange workpieces
  end while
  move_robot (finishing_robot_configuration)
end procedure
```

For a general case, the program code is shown in Pseudo-code 1 (to keep the code more concise, move commands are parametrized with target configurations instead of interpolations). While the main operation sequence is to be followed, there can be minor deviations from this sequence in case of different scenarios.

5. Functional description of the Digital Twin

To create all the necessary data mentioned in Section 4.3, functional blocks of the DT models are defined. These blocks are organized based on the task steps and also, considering the related, already available tools (e.g., libraries or algorithms). In the following, the purpose, the necessary inputs for the preparation, as well as the included data and methods are presented for each model. It is noted that the order of the model preparation is not necessarily the same as the order in which they are discussed. In fact, following the concurrent engineering paradigm, design, modeling and planning phases can be parallelized to a certain extent. The extended development process, including the generated planning-related data in each step, is shown in Fig. 5.

5.1. Kinematic modeling

The kinematic model is a fundamental part in the DT, providing the underlying kinematic structure and essential functionalities. When selecting and preparing the kinematic model and its modules, the basic task assumptions need to be considered, like the number of necessary spatial dimensions or static/dynamic environment.

Kinematic modeling needs to be prepared offline, as it is the basis of both offline and online computations. In general, when preparing the kinematic model, the following tools need to be prepared: **layout planner**, for analyzing and adjusting the model's geometric parameters in component relations to achieve a feasible layout; **manipulator inverse** and **forward kinematics**, to define key frames and robot

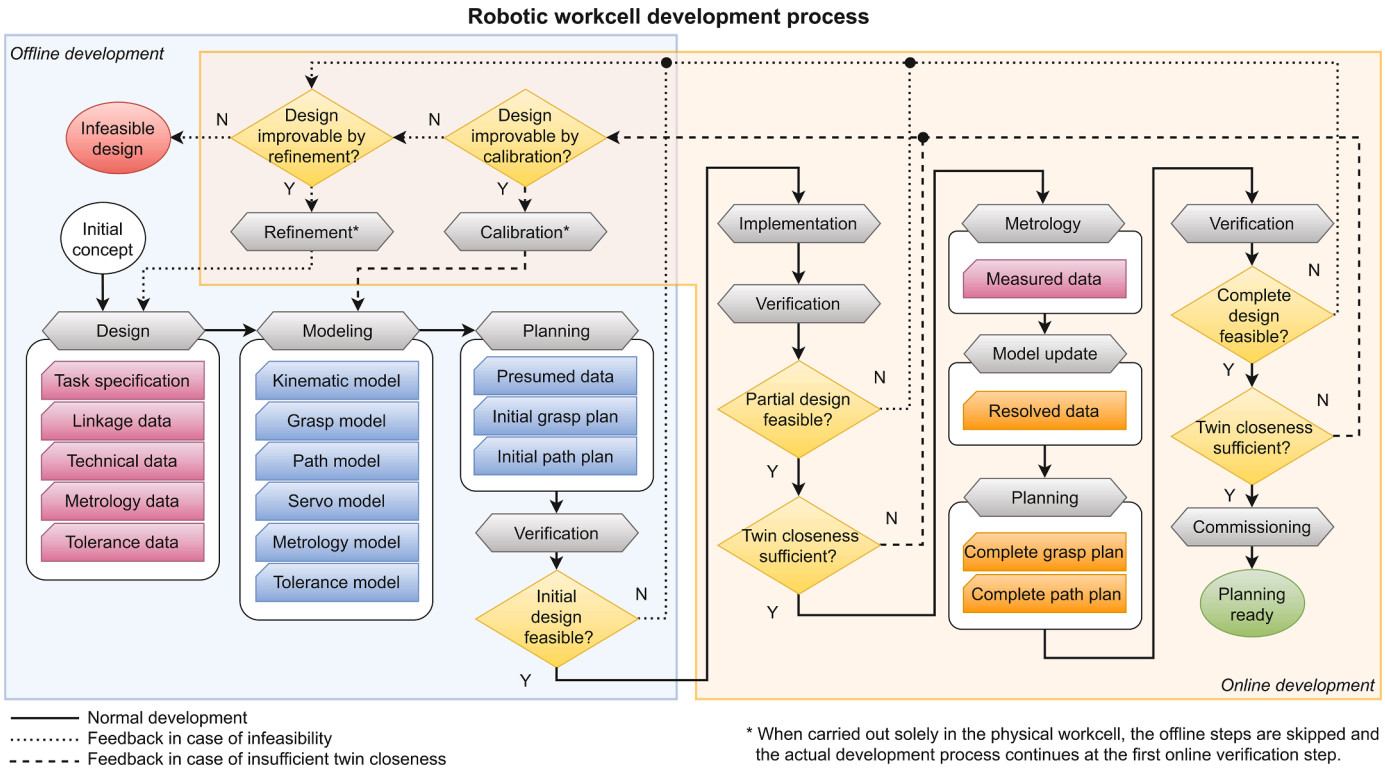

Fig. 5. Digital twin-based development workflow.

configurations (see Section 4.1) and to switch between the configuration and task space; **rigid body kinematics**, for moving components by changing the component relations to modify the kinematic model scene; and **collision query**, for checking the collision status of the kinematic model in a defined scene.

Various solution approaches exist for each above defined functionality. For basic manipulator kinematic description and conventions, the reader is referred to [44]. For modeling multi-body systems, the Unified Robot Description Format (URDF) [45] is commonly used, especially in the ROS community [25]. There exist modeling and simulations tools, such as the Robotics Toolbox [28] for Matlab or the LinkageDesigner module for Wolfram Mathematica [46]. Moreover, various collision engines are presented in [47].

The required input data include the task description with assumptions, picking and placing poses. In addition, geometric data (linkage data), including CAD or mesh geometries of the workcell components and the workpiece, geometric relations between the components and the description of the robot mechanism are also necessary. Furthermore, in case of development iterations, feedback from other models needs to be considered (such as grasp or path plan). It is noted that the latter feedback data is necessary for optimization and modifications in case of infeasibility induced design refinement iterations. This statement holds for every model with feedback data. The object-oriented representation of the kinematic model is shown in Fig. 6.

5.2. Grasp modeling

The grasp model is responsible for determining the workpiece-gripper relationship [48], and corresponding tool path points. Grasp synthesis includes the definition of a grasp frame pair, in which one frame is attached to the workpiece and one to the gripper separately. In addition, for such a pair, the gripper setting needs to be determined. When the frames of the grasp frame pair are aligned, the manipulator is capable of grasping the workpiece with the corresponding gripper setting (see Fig. 4). There can be multiple candidates for grasp frame pairs (even for specific error cases), as well as for seizing and releasing configurations.

The grasp planning can only be performed entirely offline, if there are no grasping-related task-specific uncertainties. Otherwise, the initial grasp plan is prepared offline, based on the initial, presumed data, along

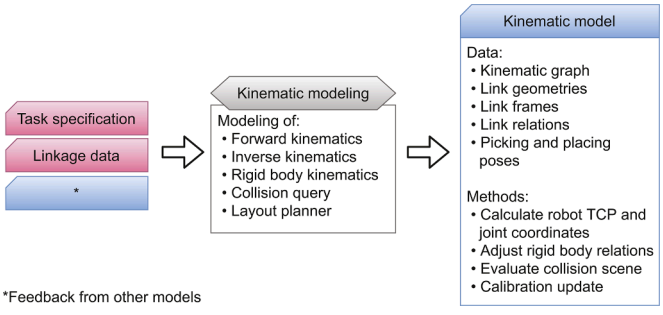

Fig. 6. Preparation of the kinematic model functional block.

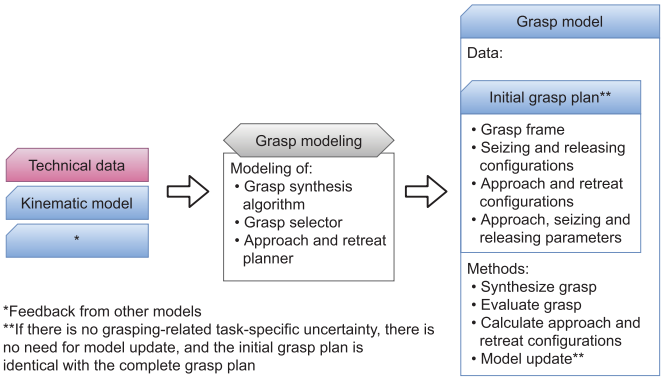

Fig. 7. Preparation of the grasp model functional block.

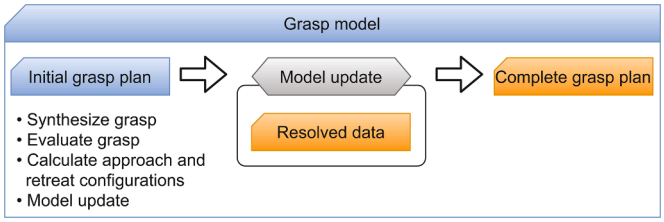

Fig. 8. Grasp model update in the online phase.

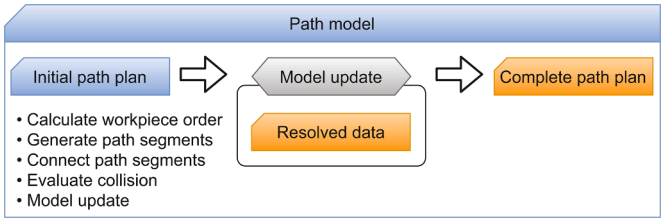

Fig. 10. Path model update in the online phase.

with the necessary methods. This reduces the online development work to parameter tuning and model update. To achieve a feasible grasp plan, the following tools are necessary: **grasp synthesis algorithm**, to generate the grasp frame pair, seizing and releasing configuration candidates and corresponding parameters; **grasp selector**, to select a suitable combination of grasp frame pair and corresponding configurations; and **approach** and **retreat planner**, to prepare approach and retreat configurations for seizing and releasing configurations and also the gripper setting parameters before seizing and after releasing.

Grasp planning has a broad literature. Bicchi and Kumar give a summary on analytical methods [49], while Bohg et al. on data driven methods [50]. Many approaches are presented for grasping of novel workpieces in [17]. Also, complete tools are available, such as the corresponding part of OpenRAVE [26], or GraspIt! [51] for grasp planning and evaluation, along with planning workflows for flexible robotic cells [52].

The necessary input data are technical data, including the gripper characteristics (operation range, max. loading, etc.), and the kinematic model with the workpiece, gripper and workholder geometries and the picking and placing poses. The object-oriented representation of the grasp model is shown in Fig. 7.

If online grasp planning is inevitable, then the initial grasp plan is completed in the online phase by revisiting the grasp model, after resolving the uncertainties. The model data is updated based on the metrology results (see Section 5.5), and the grasp planning is re-evaluated to create the complete grasp plan for the actual cycle (see Fig. 8).

5.3. Path modeling

The path model is responsible for determining the order of picked workpieces and path points, as well as the transitions between the path points. Similarly to the grasp model, path planning can possibly be performed entirely offline depending on the application. In any case, at least the initial path plan and the methods are established offline to reduce the online development work. To achieve a feasible path plan, the following tools are required: **sequence planner**, to return the order of pickable workpieces (if multiple are present); **motion planner**, to determine the order of path points and generate and connect tool path segments in the task and joint space; and **collision query**, for checking the collision status of the scene (not necessarily the same as in the kinematic model).

For robotic pick-and-place operations, the sequence planning problem is typically an extension of the well-studied Traveling Salesman Problem (TSP) [53] or the Generalized Traveling Salesman Problem (GTSP) [54], with domain-specific constraints. The combined solution of robot configuration selection and task sequence is often advantageous, which is presented as the Robotic Task Sequencing Problem (RTSP) in [55].

Robot path planning also has a broad literature with numerous existing solutions. The most commonly applied heuristic methods are probabilistic ones, namely rapidly exploring random trees (RRT) [56] and probabilistic roadmap (PRM) [57]. Other classical and heuristic methods are summarized in [58]. Additionally, there are offline robot programming tools including robot motion simulation capabilities, such as RoboDK [59], Gazebo [60], MoveIt! [61], or the corresponding part of OpenRAVE [26].

The necessary input data includes technical data in form of robot velocities and accelerations, the kinematic and grasp model with the key robot configurations (path segment endpoints), as well as the servo model (see Section 5.4). The object-oriented representation of the path model is shown in Fig. 9.

In case of path-related task-specific uncertainties, the path planner needs to be applied both in the offline and online phases, similarly to grasp planning. However, in this case the model data is updated based on both the servo and metrology results (see Section 5.4 and 5.5). This update is shown in Fig. 10.

5.4. Servo modeling

In case of task-specific uncertainties, two approaches are applicable, namely servoing and metrology. Servoing can handle robot motions where the destination (be it seizing, releasing or any other point) cannot be defined by geometric coordinates. Instead, the target point is defined as the fulfilment of a measurement-based condition. Using a closed loop control cycle, the missing endpoint can be reached with continuous measurement and actuation of the manipulator, until the defined target condition is reached. In addition, servoing is also a tolerance enhancement technique (without the need for changing the equipment), and can be effectively used for error scenarios as well (e.g., when the workpiece lays significantly away from the expected picking pose). The most common servo techniques are visual, force and tactile servoing.

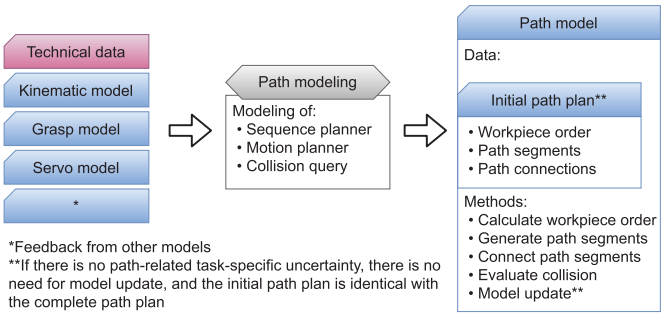

Fig. 9. Preparation of the path model functional block.

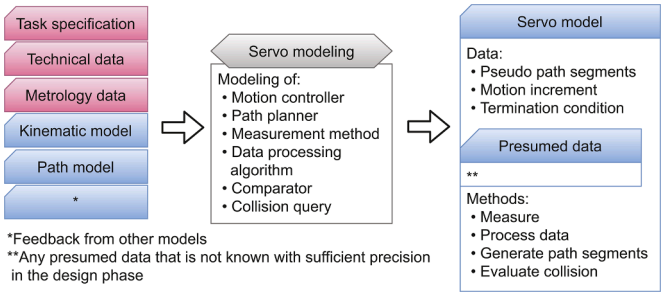

Fig. 11. Preparation of the servo model functional block.

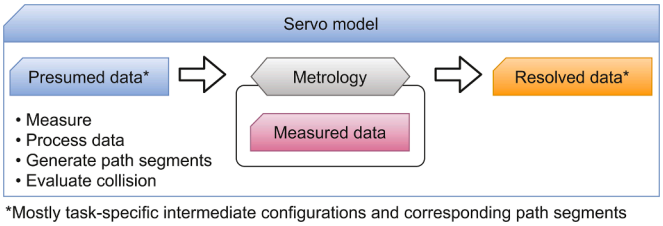

Fig. 12. Role of metrology in the servo model in the online phase.

Although most of the times the servo model cannot be finalized entirely offline, its modeling is recommended to be started offline. This model is essentially the combination of a metrology model and a path planner (usually in a simpler form), therefore, the modeling includes the following tools: **measurement method** together with the **data processing algorithm**, to acquire and process the information necessary for servoing; **comparator**, to compare the processed data with the target condition; **path planner** and **motion controller**, to generate and execute the motion increment (or velocity) for servoing, based on the comparison results; and **collision query**, similarly as in the path model. The path planner can be the same as in the path model, or different.

For visual servoing, active perception is summarized in [62] and [63]. The basic and advanced visual servoing techniques are presented by Chaumette and Hutchinson in [64,65]. A coupled vision-force servo approach is presented in [66] and a framework for tactile servoing is presented in [67]. Path planning for servoing is summarized in [68].

The servo technique is greatly dependent on the applied sensors and strategy, thus the realization of these tools can be various. The necessary input data includes the task specification (i.e., which data is uncertain), technical data in form of sensor and manipulator characteristics (measurement range, resolution, calibration parameters, etc.), metrology-related data and the kinematic and path models. Metrology-related data can contain device-specific data, such as region of interest, gain and filtering, and data processing-related data, such as training set, thresholds and segmentation parameters. The object-oriented representation of the servo model is shown in Fig. 11.

As the servo model depends on the observation of the physical environment, it is inevitably applied online. During the online phase, the originally presumed data is updated with the actual data from the resolved uncertainties. This is then provided to the models performing online planning (see Fig. 12).

5.5. Metrology modeling

The other possibility for resolving task-specific uncertainties is the application of metrology. Unlike the servo model, metrology resembles an open control loop by obtaining and processing the required information for the models realizing online planning. It can also be responsible for identifying error cases. Technically, the servo model corresponds to robot motion commands, while metrology model corresponds to sensor activation commands. Metrology is most commonly vision- or tactile sensing-based.

Similarly to servoing, the complete modeling of metrology is usually

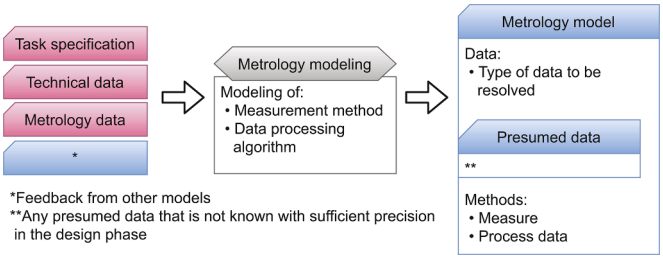

Fig. 13. Preparation of the metrology model functional block.

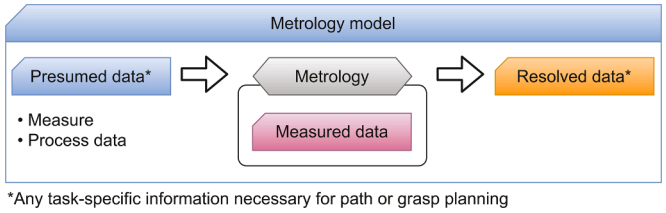

Fig. 14. Role of metrology in the metrology model in the online phase.

not reasonable due to external factors (e.g., lighting scene or material properties in case of vision-based metrology). Instead, model refinement is carried out iteratively, based on experiments in the real workcell. Thus, the metrology model is generally finalized in the online phase, where every environmental factor is present, and model tuning can be performed. The tools to be prepared are the **measurement method**, together with the **data processing algorithm**, to acquire and process the information necessary for resolving the uncertainties.

A great amount of research has been done in the field of computer vision. Conventional methods for robotic applications are presented in [69], whereas deep learning methods are summarized in [70]. Moreover, vision-based methods for flexible robotic applications are presented in [71]. Besides the application of vision-based metrology, tactile sensors are also applicable. Corresponding approaches are summarized in [72].

The metrology model is also greatly dependent on the applied sensors, therefore, the realization of these tools can be rather diverse. The necessary input data are technical data in form of sensor characteristics, metrology-related data, as well as the uncertainties to be resolved. The object-oriented representation of the metrology model is shown in Fig. 13.

Metrology is also applied in the online phase. Similarly to the servo model, during the online phase, the presumed data is updated using the resolved uncertainties, which is then provided to the models performing online planning (see Fig. 14).

5.6. Tolerance modeling

Lastly, the tolerance model needs to be set up. This model is necessary for evaluating the precision aspect of the operation. Its purposes include the dimensional chain setup, tolerance stack-up calculation along the dimensional chain and stack-up comparison with the precision requirements (task feasibility evaluation). The tolerance model is also applied for determining the deviations in the physical workcell, and to prepare the feasible tolerance criteria (lower and upper bounds) for twin closeness assessment. Additionally, the tolerance model needs to suggest whether growing the feasible tolerance region (see Fig. 1(c)) is necessary or not, based on the design specification. The tool to be created offline by modeling is the **tolerance analysis method**, which evaluates the tolerance stack-up.

The literature of process-related tolerance modeling for robotic applications is rather scarce. Research can be found on such topics for robotic machining [73], however to the best of the authors' knowledge, there is no tolerance model specifically representing robotic assembly applications, like the present pick-and-place operation, from the

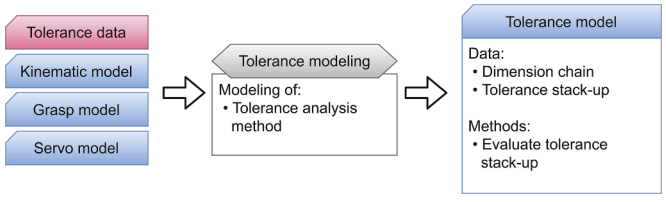

Fig. 15. Preparation of the tolerance model functional block.

operational point of view. The authors are convinced that by considering workholding, grasping, servoing and metrology characteristics, a generalized tolerance stack-up can be defined for pick-and-place, which could be a useful tool for workcell design and development. However, this is in the scope of a future paper. Nonetheless, tolerance representations and tolerance analysis approaches are summarized in [74,75].

As currently adequate pick-and-place specific tolerance analyzing tools cannot be found in the literature, a manually set up, transformation-based evaluation is recommended. The necessary input data include tolerance data in form of manipulation, seizing, releasing, manufacturing and metrology tolerances. In addition, tolerance-specific assumptions, the kinematic and grasp model with the seizing, releasing and grasp frames, and the servo model are necessary. The object-oriented representation of the tolerance model is shown in Fig. 15.

5.7. Verification

Having established the basic functionalities of the DT, the only major point left for further detailing is the verification steps of the development workflow.

5.7.1. Feasibility assessment

Feasibility is assessed in three stages as shown in Fig. 5. As part of the offline verification process, feasibility assessment is first carried out in the model space after offline planning. The digital models of the DT need to meet the constraints posed by task requirements. In general, feasibility assessment involves the workcell layout, the digital task execution (including grasp plan, path plan and metrology model) and the tolerance model. The workcell layout needs to be collision-free and constructible. The digital task needs to be collision-free and within the capabilities of the manipulator and gripper, such as range, resolution or force limits. In addition, the metrology system needs to be suitable including measurement range, resolution and occlusion, among others. Moreover, task constraints (e.g., cycle time) need to be satisfied, and the tolerance requirements need to be fulfilled by the tolerance model.

In case of any infeasibility, the digital workcell design needs to be refined. Depending on the occurring infeasibilities, modifications can be made in the design parameters (e.g., workcell layout), component geometries, used equipment or even in the task definition. These modifications are carried out in the model space. In each iteration, the affected models need to be actualized and recomputed, and the feasibility needs to be re-evaluated until feasibility on the model level is ensured.

In case of the online verification processes, feasibility is assessed in the physical space before and after online planning. First, the original modeling assumptions need to be checked. Qualitative differences can occur due to insufficient implementation. These can be in form of additional obstacles in the workcell environment risking collision, ill-considered robot joint limits, gripper or sensor capabilities, etc. Such deviations often cause infeasible task execution in which case the design needs to be refined.

If the design is refined in the model space, the development process needs to be restarted from the design phase, considering the latest lessons learned. If the modifications are carried out solely in the physical space, then the iteration process is simply the re-assessment of feasibility, bypassing the offline steps. This can happen for example, when the cabling restricts the robot motion or occludes the region of interest of the vision system. Here, leading the cables in a different way in the physical workcell can possibly solve the above problems, without any modifications in the digital model.

During this assessment, also the automated planner tools need to be investigated. Each generated plan needs to meet the task requirements. Due to the factors, the modeling of which is troublesome or pointless, online experimentation is usually inevitable. Typically, it is the servo or metrology models that require refinement, as their complete offline preparation is rather complicated.

5.7.2. Twin closeness assessment

The assessment of the DT closeness, i.e., checking if the digital and physical counterparts are within a feasible tolerance region, is carried out during online development in two stages: after implementation and after online planning (i.e., after the finalization of the physical workcell). Although the tolerance model is assessed for feasibility in the offline phase, the actual realization of the workcell (manufacturing, construction, device precision, etc.) is checked during twin closeness assessment. If the tolerance model covers every relevant scene and dimension of the twin closeness, the physical workcell can be checked for each of these elements considering the corresponding worst tolerance case, which bound the feasible tolerance region. On the other hand, in more complex scenarios, the application of the tolerance model might not cover each element sufficiently. In these cases, for the assessment of the overall twin closeness the operation needs physical testing as well.

For example, relevant scenes can include the component arrangement before, during or after seizing or releasing, or at certain points during robot motion, which are near collision. Also, relevant artefacts can be contacting faces, edges or points on the workpiece, gripper or fixture, or near colliding areas on any components.

If the closeness is insufficient, the deviation needs to be improved with any of the methods presented in Fig. 1, but preferably via calibration. If sufficient closeness cannot be reached in this way (possibly due to non-geometric effects), the feasible tolerance region needs growing through design refinement. For example, if the releasing configuration is slightly inaccurate, which prevents the successful placing of the part into the placing workholder, then the task requirements cannot be fulfilled. Therefore, the tolerance stack-up for the releasing configuration, as well as for the workholder needs to be checked. As the digital model was prepared so that the releasing configuration is suitable, the physical construction of the workcell or manufacturing of the components might not meet the specified design requirements of the CAD models or drawings. Also, the robot controller might use different kinematic parameters than used in the digital model, or possibly, the feasible tolerance region was set up by neglecting a significant inaccuracy source. By checking the inaccuracy sources in the workcell the calibration can aim specific parameters of the system. If need be, tolerance enhancement techniques, such as visual servoing, can also be applied through design refinement, even for uncalibrated systems.

6. Implementation of the methodology

In the following, two implementation scenarios are presented. First, the detailed development process and experiments are shown for a flexible pick-and-place scenario. Then, the reconfiguration of the same scenario is introduced briefly, focusing on the main steps of a feasible reconfiguration. Finally, further applications are listed, which are well suitable for the methodology.

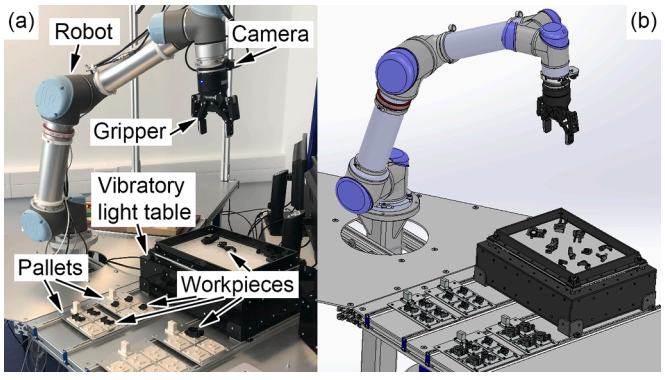

Fig. 16. The physical workcell (a) and its CAD model (b).

6.1. Overview of the example scenario

The task of the experimental workcell is to manipulate and sort workpieces from a semi-structured picking arrangement onto the corresponding workpiece pallets. The physical workcell is shown in Fig. 16 (a). The scenario involves three types of workpieces that are injection molded RC (remote control) car components with different geometries. These are produced simultaneously on the same mold runner and are separated from the mold runner into a common, mixed bulk. This mixed bulk is fed onto a vibratory light table in the robotic workcell. Using a 2D camera mounted on the robot, together with the homogeneous background provided by the light table, the workpieces are detected, identified and localized. For capturing the image, the robot moves to the corresponding metrology configuration.

The manipulation process takes advantage of the fact that the workpieces lay on a flat surface in one of their stable poses. Constrained by the workpiece slots on the pallets, the workpieces can be picked only from a single stable pose without re-gripping. Consequently, the workpiece stable poses are considered as task-specific uncertainties and need to be determined from the camera image. Those workpieces that lay in the suitable stable pose are referred to as graspable workpieces. Also, those graspable workpieces that can be picked up without the robot and gripper colliding with the environment or any other workpiece are referred to as pickable workpieces. With the above consideration, after identifying and localizing the pickable workpieces, they can be manipulated onto the corresponding workpiece pallet. With the separate pallets, arranging into a structured layout and sorting can be realized simultaneously. One of the workpiece pallets is shown in Fig. 17(a).

The workpieces may be tangled or lay too close to each other or the walls of the light table. Also, the workpieces may lay in a stable pose, not suiting the picking strategy. In such case, (i.e., if there are no pickable workpieces), the workpieces are rearranged on the picking area by activating the vibratory table, and the identification process starts over. The table vibration and the feed onto the table is configured so that the probability of getting a pickable workpiece is sufficiently high for reliable workcell operation. Further details of the feeding and rearrangement mechanism are not in the scope of the paper.

The applied picking strategy is selected so that the robot approaches the workpieces with the last joint axis normal to the top surface of the light table, i.e., the axis is considered vertical. This approach is possible due to the pallet geometries, as all workpieces can be placed with the last joint axis being close to vertical. This strategy facilitates the grasp planning, collision checking and path planning steps, as the picking phase essentially becomes a 2,5D task with no change in the robot configuration setup. During the placing phase, additional last joint axis orientations are introduced, however, these can be planned offline.

The task-specific uncertainties can be summarized as actual workpiece type and stable pose, and workpiece position and orientation on the top of the light table. These change cycle by cycle with the help of the vibratory function of the table. From the three workpiece types, one is selected to present the proposed methodology. The selected RC1 workpiece is shown in Fig. 17(b). Being an experimental scenario, no strict cycle time requirement was set up. Nonetheless, cycle time was kept reasonably low, and was evaluated during the experiments in Section 6.9.

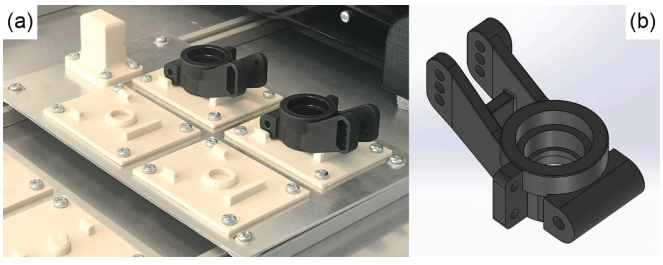

Fig. 17. The selected RC1 workpiece on the corresponding four slot workpiece pallet (a) and the workpiece CAD model (b).

6.2. Initial concept

The DT set up starts with an initial concept. The initial workcell design was given in the form of a UR5 robot on a robot stand, surrounded by three identical worktables. The robot is equipped with a Robotiq 2F-85 finger gripper and an IDS XS camera. The purpose-built vibratory light table and the part pallets were fixed on one of the worktables well within the robot reach (see Fig. 16). As the rough cell layout was known and all components were pre-selected and available beforehand, a detailed (in terms of component geometries) CAD model was prepared. However, in terms of component relations, the CAD model was preliminary. Using this CAD model as initial design, the design phase can be started.

6.3. Design

During the design phase, the necessary data is collected and/or defined for the building of the different models within the DT. Most importantly, the CAD model is refined by determining component relations and the pose of each equipment. The CAD model, which was prepared during the design phase, is shown in Fig. 16(b). Further design specification details of the workcell will be listed along with the models, and/or in the experimental results, wherever relevant. As mentioned earlier, the design, modeling and planning phases are not completely separated, these can also be parallelized to a certain extent. Therefore, it is not expected to provide every information without getting feedback from the subsequent phases.

6.4. Modeling and offline planning

6.4.1. Kinematic modeling

The kinematic model of the DT was prepared in Wolfram Mathematica, using the LinkageDesigner module [46]. In this module, the linkage model is built up with the parametric kinematic graph of the robot and its environment. The components are connected using parametric mechanism joints with which a functional, adjustable digital model can be created. The linkage model is shown in Fig. 18(a) with the corresponding kinematic graph in Fig. 18(b).

The linkage model contains the geometric relations in form of homogeneous transformation matrices together with the mesh geometries of the links. These mesh geometries could be generated from the CAD model. The inverse and forward kinematic model of the robot is also prepared here. Together with the PQP (Proximity Query Package) and V-Collide collision engine libraries (presented in [47]), this model representation allows collision checking between the mesh geometries of the workcell.

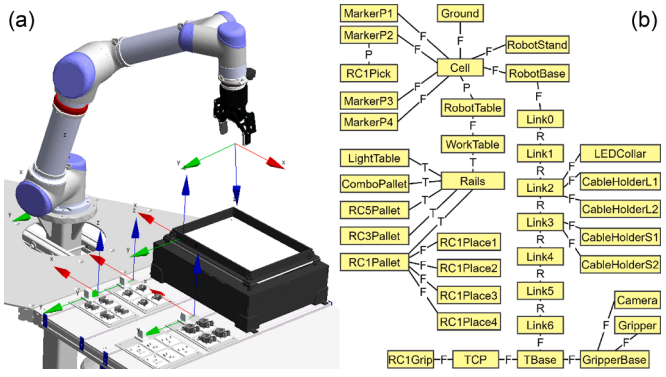

Fig. 18. Linkage model (a) and the corresponding simplified kinematic graph (b).

Using the kinematic model, the estimated releasing frames and a number of uniformly sampled (approximate) seizing frames were checked for reachability and collision. For the metrology frame, the optical axis of the camera was aligned with the center normal of the light table top surface. Its vertical location was set up in a presumed height. Using the inverse kinematic model, the frames are transformed to configurations, and a common configuration setup was selected suitable for all of them. With these considerations, the workcell layout was accepted. Nevertheless, the presumed configurations will need to be refined.

6.4.2. Grasp modeling and offline planning

During grasp modeling, two constraints were considered. First, the workpieces lay in one of their stable poses. This significantly reduces the space of possible pickable workpiece poses, which can be exploited during grasp planning. Consequently, an algorithm is necessary for calculating object stable poses based on their mesh geometry.

Second, the applied gripper is a parallel finger gripper with flat contact surfaces on the fingers. This constraint can be applied during grasp synthesis. To utilize this condition, a mesh analysis algorithm is required, which is able to detect opposing parallel surfaces on the workpiece mesh geometry. These can then be used as contact surface pairs combined with the contact surfaces of the gripper fingers. Finally, the finger distance and a sufficient grasping force will be necessary for gripper setting. Based on the contact surface pair candidates, grasp frame pairs can be generated. From here, the determination of the releasing frames, and corresponding approach and retreat frames can be realized using the kinematic model.

Up to this part, grasp synthesis can be pre-computed offline. The only remaining part for online planning is the determination of the seizing and corresponding approach and retreat frames. Since the grasp frame will be determined w.r.t. the workpiece reference, the online planning of the said frames is linked with the localization of the workpiece reference frame.

Finally, a collision checking algorithm will be necessary for online planning. Due to the uncertain location of the workpieces on the light table, for each graspable workpiece, the collision state of the gripper needs to be addressed. This needs to be checked between the gripper and the light table walls, and between the gripper and the other workpieces. Overall, the relevant design data are the characteristics of the Robotiq 2F-85 gripper listed in its specification (max. finger distance, max. gripping force, etc.).

Based on the above considerations, at first the workpiece stable pose generating algorithm is required. Stable poses can be calculated for any mesh model with known center of gravity (CoG). In case of concave geometries, the convex hull needs to be determined. Then the stable triangles are found by projecting the CoG onto each triangle using their face normal. A triangle is considered stable if the projection pierces its

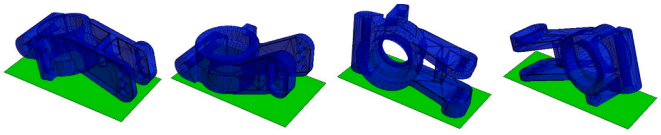

Fig. 19. Possible stable poses of the RC1 workpiece.

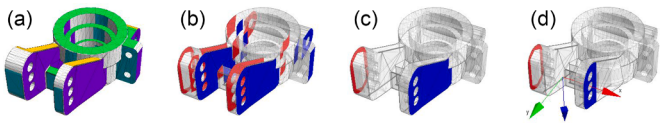

Fig. 20. Steps of the selection of the contact surfaces: faces corresponding to characteristic directions (a), the group of faces with opposing face normals in the selected direction (b), the topmost and outermost pair of surfaces (c) and the overlapping part of the selected surfaces with a grasping frame candidate (d).

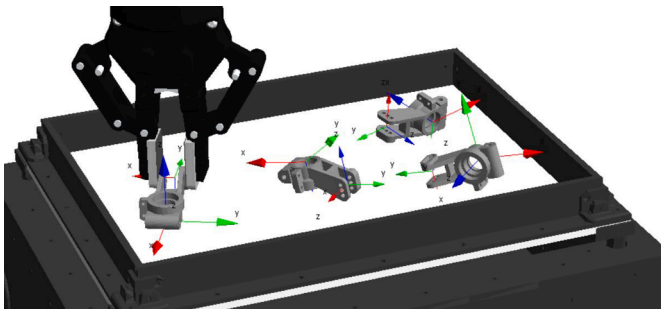

Fig. 21. Final result of the grasp synthesis for the RC1 workpiece.

area. The vertices, edges or faces of the original mesh model corresponding to the stable triangles of the convex hull need to be identified by remapping. The found elements are then grouped by connectivity, face normal direction and distance from the CoG. Finally, the homogeneous transformation matrices can be calculated that transform the original mesh model into the corresponding stable pose [76].

Using the above method, the stable poses of the current workpiece can be generated (see Fig. 19). In this case, one of the selected workpiece's stable poses is the same as the one defined by the workpiece pallet. The pallet slots align the lower part of the workpieces. For the selected workpiece, the center hole is aligned with a chamfered pin and the end faces of the bar on the back are aligned with chamfered side walls (see Fig. 17(a)). Consequently, the workpiece grasp frame determination reduces to finding an upper, sufficiently graspable feature (with opposing parallel surfaces) on the workpiece in the corresponding stable pose.

For parallel surface pair detection, a face normal clustering method was developed. The characteristic face normals are identified by filtering the face normal directions with the total area of the corresponding triangles (see Fig. 20(a)). Next, the opposing faces are grouped to form the graspable feature candidates. From these, the normal direction with the topmost graspable surfaces was selected (see Fig. 20 (b)). Using the outmost faces and calculating their intersecting projections, the contacting surfaces with the grasping width can be precomputed (see Fig. 20(c)). By calculating the geometrical centerpoint of the bounding box of the contact surfaces, a grasp frame candidate can be generated (see Fig. 20(d)).

There remain 3 DoF for the determination of the grasp frame w.r.t. the gripper, since the connection between the flat gripper finger and workpiece surfaces can be described with a planar joint. For the rotational DoF, vertical direction was selected, following the picking strategy. Whereas the two translational parameters were selected so that the gripper fingers sufficiently cover the grasped faces, providing a firm grasp. By fixing these parameters, and considering the fixed configurational setup, there are only two possible grasp candidates, due to the cyclic joint coordinates of the last axis. The grasp selector only needs to select whether a 360° rotation of the last axis is to be applied or not.

Next, the grasping parameters were set up. The finger distance for grasping was determined during grasp synthesis based on the workpiece geometry. The grasping force was tested manually, using the real workpiece and gripper. The alignment of the grasp frame pairs is shown in Fig. 21. The corresponding seizing approach and releasing grasping parameters were set to a slightly wider finger distance, than the one used for seizing. Whereas, for the metrology, the gripper setting was defined to be fully open, to avoid the occlusion of the scene during image capture.

Lastly, the collision checker was prepared. Exploiting the 2,5D nature of the pick-and-place process and the corresponding assumptions discussed earlier, collision avoidance can be also reduced to a 2,5D problem. The main idea is to project the geometry of the gripper fingers onto the top surface of the light table for the graspable workpieces. Then, considering a safety distance due to the perspective distortion (see

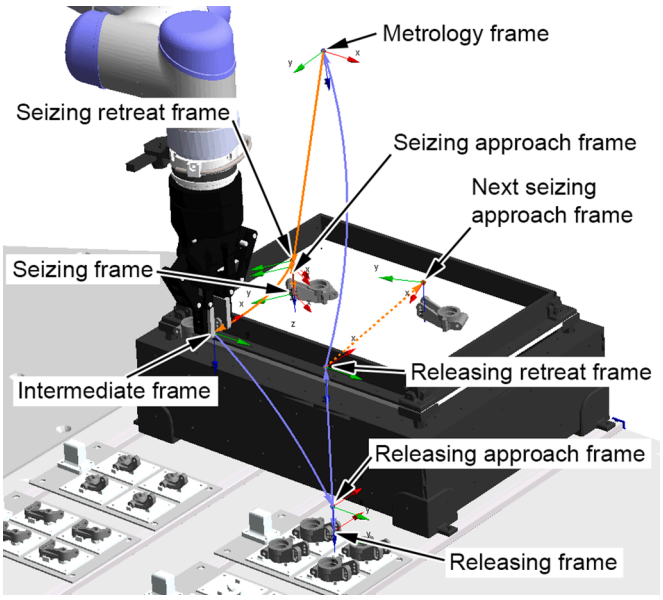

Fig. 22. The complete tool path, the pre-computable segments (blue) and the online planned segments (orange) with the corresponding endpoint frames. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

Section 6.4.5), the projected polygons are checked for intersection with other polygons, representing the workpieces and the light table walls. Additionally, the height of the objects is considered. The potentially colliding polygons are filtered based on the presumed height of the corresponding object, and the vertical position of the gripper fingers. The captured image and corresponding image processing steps are shown in Section 6.6 and in Fig. 25.

6.4.3. Path modeling and offline planning

In this scenario, the tool path could be separated to offline and online computable path segments due to the fixed metrology frame and workpiece placing poses. Since the picking phase is uncertain in terms of workpiece picking poses, online path planning is inevitable. The tool path was split by introducing an intermediate path point, which separates the picking phase from the placing phase. Path planning from the intermediate frame to the releasing approach, releasing and releasing retreat frames can be planned offline. Furthermore, the segment from the releasing retreat frame to metrology frame can also be pre-computed. The complete tool path in a general case is shown in Fig. 22.

The main criteria for the tool path are to obey joint limits and be collision-free in every points of the path. Reachability was ensured with the prepared workcell layout. On the other hand, collision avoidance still needs to be regarded. Collision might occur between the robot and the environment, as well as between the robot links including the mounted equipment.

The intermediate frame was defined above the edge of the light table closest to the placing pose. Also, a safety distance was considered between the light table and the gripper with the seized workpiece. Since the robot remains in the same configuration and maintains a close to vertical last joint axis, the planning process is significantly simplified. The pre-computable path segments were realized using joint interpolation with an additional vertical linear interpolation (in the task space) immediately before and after part placement. The latter correspond to the releasing approach and retreat frames, respectively. The partial tool path result was checked for collision using the corresponding robot configuration interpolation, by densely sampling the path segments. For the evaluation of collision between the mesh geometries, the kinematic model was used.

The remaining path segments for online planning were the ones corresponding to seizing, which occur in two forms. First, from the metrology frame to seizing approach, seizing and then to the seizing retreat frame. Or, alternatively, if multiple workpieces are manipulated based on a single metrology step, from the intermediate frame to seizing approach, seizing and then to the seizing retreat frame. In both cases, the last missing path segment is the one from seizing retreat to the intermediate frame.

These were considered as a combination of an approach phase with joint interpolation and a linear interpolation in vertical direction. The linear segments, similarly to the releasing path, were added immediately before and after seizing the workpiece, for the seizing approach and retreat frames. The online planned path segments were prepared by sampling seizing frames on the light table and checking the corresponding path segments for collision. Based on the simulation results, in the selected robot configuration setup, there are no observable collisions (not even near collisions). Therefore, it is assumed that regular joint and linear interpolation can be applied in the online planning phase. Here, the relevant design data are the characteristics of the UR5 robot listed in its specification, including max. joint and TCP (tool center point) velocities and accelerations.

In this particular scenario, sequence planning has little impact on the cycle time, as the placing poses are relatively close to each other and the robot remains in the same configuration setup. Therefore, a basic sequence planner is prepared, addressing only the workpiece pickability aspect. The workpiece sequence is initially set up based on the order of workpiece detection. Then the sequence is refined considering the pickability constraints, i.e., pickable workpieces that block other potentially pickable workpieces are manipulated first.

6.4.4. Servo modeling

Due to the relative simplicity of the task and the precision of the system, servo techniques were presumed to be unnecessary. However, in case the tolerance requirements would not fulfil, the mounted camera could be used for visual servoing and online compensation could be realized based on the identified geometric relation between the workpiece and the gripper fingers.

6.4.5. Metrology modeling

To resolve the uncertainties originating from the semi-structured workpiece arrangement, a 2D camera-based metrology system with an image processing method was developed. A 2D approach was suitable due to the consistent background provided by the light table.

There are three types of task-specific uncertainties in this system, namely the workpiece type, workpiece stable pose and workpiece location. For the first two task, a comparison-based method was developed. The idea is to match the workpiece shapes on the captured image, with pre-computed virtual images of the virtual workpiece. With a sufficient virtual image database, the workpiece types and stable poses can be identified based on the best match. The virtual workpiece images are

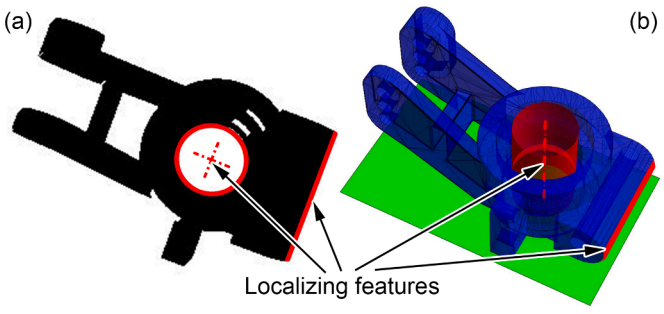

Fig. 23. A sample reference image of the database for matching (a) and the 3D workpiece (b) with highlighted localizing features (red). (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

captured from different viewpoints and with different orientations with uniform sampling. For better representation of the physical metrology, the images are generated perspectively using a virtual camera. The results were binarized for faster evaluation. Furthermore, the relative camera location and bounding box dimensions were also stored for filtering purposes. A sample reference image is shown in Fig. 23(a). It is noted that the actual reference images do not include the highlighted localizing features. The corresponding design specification contains parameters of the reference database setup, such as lighting scene, material properties, virtual camera parameters and camera-workpiece geometric relations. For matching, the design specification contains the image processing parameters, e.g., for thresholding and convolution.

The matching method is applicable for separated workpieces laying in one of their stable poses after thresholding and segmentation. In cases where the workpieces are not separated, the identification process fails. Here, the 2D convex hull of the identified shapes are computed for the online collision detection for grasping.

Although the matching method could be used for workpiece localization, its accuracy is not sufficient. Therefore, a separate method was developed to achieve the required precision. Firstly, physical marker points were placed on the light table (on the picking surface). As the physical location of the marker points are known, homography transformation (presented in [68]) can be applied. In this way, the location of any point on the image can be determined w.r.t. the markers, assuming the height of the corresponding physical entity is known. With such considerations, accurate localization can be realized, despite the perspective distortion of the captured images. The marker points were detected using simple thresholding and filtered segmentation.

For the second part of the localization task, a feature-based approach was chosen. A set of characteristic features were selected for the component that are easy to detect, yet provide sufficient information of the workpiece position and orientation (see Fig. 23). Using the above described marker points, the identified feature points can be calculated in the task space as the workpiece picking pose. Finally, to determine the seizing frames, the identified feature points need to be transformed using the workpiece grasp frame. Here, the design specification contains feature parameters (feature types, measures) and homography parameters (heights and offsets). Both components of the metrology model were set up and tuned with the help of sample images captured in the physical workcell.

6.4.6. Tolerance modeling

The dimensional chain of the pick-and-place operation was set up as a transformation chain in a plane parallel to the top surface of the light table. This chain included the workpiece manufacturing tolerances, the pallet location tolerances, robot positioning repeatability, metrology precision, as well as seizing and releasing tolerances. The tolerance stack-up was set up, and to get a comprehensible result, a worst-case analysis was performed.

The transformation chain formula was evaluated with each chain element in both the lower and the upper tolerance limit. The 2D projection of the geometry was transformed with each result. The transformed shapes were overlaid, and finally, their union was computed. This results in a boundary within which the actual workpiece will lay according to the tolerance data. Since there are two different aligning features on the workpiece pallet, two different 2D projection boundaries of the workpiece were calculated. The results for the chamfered orienting columns are shown in Fig. 24(a) and for the chamfered positioning pin in Fig. 24(b).

Based on these results, the tolerance requirement for releasing is theoretically fulfilled. For the twin closeness to be sufficient in the releasing configuration, the workpiece needs to be within the above worst-case boundary after releasing. The closeness in the seizing configuration can be elaborated similarly. However, other workcell scenes and artefacts, such as robot links along the trajectory risking collision were not covered by the tolerance model, as these do not have strict feasibility criteria. The distances between the potentially colliding components are in general much greater than the tolerances. These remaining parts of twin closeness were evaluated simply through physical testing of the operation.

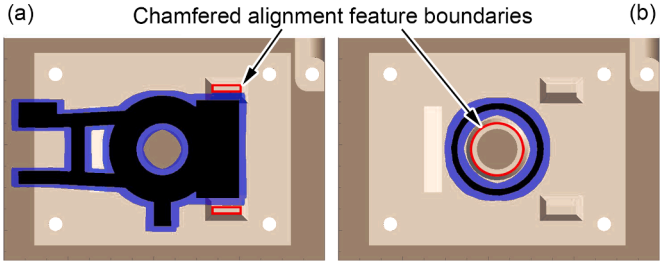

Fig. 24. The top view of the workpiece pallet with the workpiece projection (black), worst-case boundary region (blue) and geometric boundaries of the aligning features (red outline) for the workpiece back-chamfered columns (a) and the workpiece hole-chamfered positioning pin relationships (b). (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

6.5. Physical implementation

As the physical workcell was partly prepared already, its implementation did not include cell construction. Instead, the workcell components were positioned based on the workcell layout from the kinematic model of the DT. Moreover, the state machine and image processing tools were moved to the PC (personal computer) responsible for cell control. This PC was connected to each device in the workcell, including the robot, gripper, camera and light table. The verification steps and development iterations are summarized in Section 6.8 and the controller interfaces are presented in Section 6.9.

6.6. Metrology and model update

With the capability of capturing real images in the physical workcell from the prepared metrology configuration, the online metrology can be performed. Using the developed image processing algorithm, the

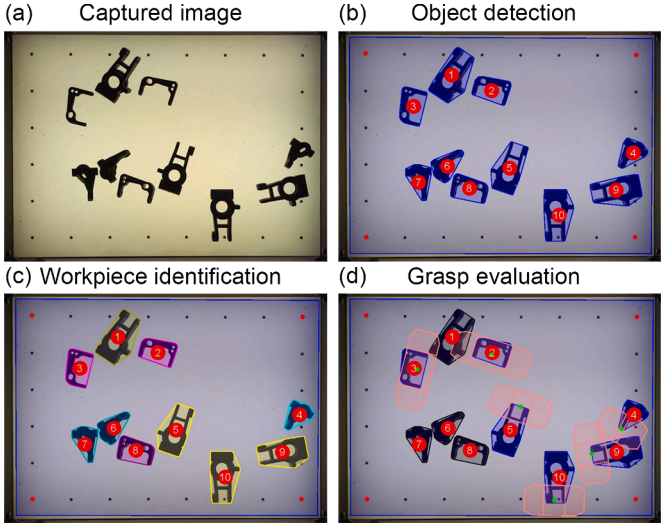

Fig. 25. Steps of resolving the task-specific uncertainties, the captured image (a), object detection (b), object identification (c), determination of the seizing frame and collision check for the projection of the gripper fingers (pink) (d). (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

identification and localization of the workpieces could be solved.

The first step is to detect the marker points on the light table and identify the cornerpoints. Next, the objects laying on the light table were detected by thresholding and segmentation. The detected objects were compared (on a binary basis) to the images stored in the reference database, filtered by the bounding box dimension and approximate location. For the identified graspable workpieces, the feature detection algorithm was executed to calculate their accurate position and orientation. These steps are shown in Fig. 25(b–d). Finally, with the resolved data, the grasp and path models are updated.

6.7. Online planning

During the online planning step, the complete tool path is prepared. This can be realized only, if the models are updated with the resolved data representing the workpiece arrangement in the actual instance. First, the seizing configurations need to be determined with the help of the grasp model. As the metrology returns the position and orientation of the workpiece reference frame in the task space, the seizing frame can be computed using the grasp frame pair. Next, for each seizing frame, gripper collision needs to be checked. This can be evaluated based on the captured and processed 2D image (see Fig. 25(d)) using the collision checker prepared in the grasp model. If a workpiece lays in a suitable stable pose, and the corresponding finger projection is intersection-free (i.e., the workpiece is graspable and pickable), then the corresponding seizing frame is applicable for path planning.

For each captured image, every graspable workpiece is checked for collision, and each collision-free instance is put in a queue for manipulation. The sequence of the picked workpieces depends on whether the grasping of a workpiece is blocked by another pickable workpiece. Otherwise, the sequence is determined by the queue, using a first-in-first-out strategy. The filtered seizing frames can then be used for the path planning up to the intermediate frame. This can be realized with regular linear and joint interpolations, as discussed earlier. The complete path is then formed by connecting the path segments.

The pseudo-code matches the one presented in Pseudo-code 1, except for *intermediate_2_configuration*, which coincides with the releasing retreat frame in this case. Therefore, lines 21-23 are omitted. The adjusted pseudo-code can be executed by the cell controller after post-processing.

6.8. Verification

In the current scenario, there were no offline feasibility issues in the digital workcell. After determining the workcell layout and robot configuration setup, the whole digital operation was assessed feasible. Furthermore, the tolerance analysis also returned feasible operation precision for the seizing and releasing frames as well as for the placing pose.

However, after implementation, the feasibility assessment highlighted that an additional joint limitation needed to be introduced. This was caused by the cables running along the robot arm that were stretching during testing. The problem was resolved by adjusting the grasp selection to consider this limitation, and select the grasp which does not cause cable stretching.

Moreover, twin closeness assessment highlighted that the light table and the pallet positions were inaccurately set up in the physical space compared to the kinematic model. As the robot was already calibrated, it could be used as a measuring device for further cell calibration. Using an impromptu tool as a touch probe, the light table marker points and pallet cornerpoints were measured. With manual robot jogging, the corresponding TCP poses (considering the tool offset) could be determined from the robot controller. The actualized dimensional parameters (in the DT) provided sufficient precision in the second iteration and the model became ready for the online planning phase.

After the first online computations, other feasibility issues were identified. On the zeroth iteration, the metrology frame was not far enough from the table to fit the whole picking surface within the captured image. Therefore, the metrology configuration was adjusted in the DT. Furthermore, the metrology model needed to be tuned based on captured images in the physical workcell. The feature detection parameters also needed tuning for consistent recognition and precise localization, as the original sample images did not sufficiently cover the variety of flexible scenarios. Moreover, velocity inconsistencies were found due to inaccurate post-processing. Here, the robot joint and linear motion velocity parameters needed tuning.

On the other hand, twin closeness was appropriate after the earlier cell calibration. With the infeasibilities resolved in the DT, consistent and precise workcell operation was achieved. After the final verification step, the development was ready, and the physical workcell was prepared for operation.

6.9. Experimental results

The overall cell control is managed by a PC, running a code in Wolfram Mathematica 11.3. This PC is connected to the cell devices. The simplified data exchange is the following. Motion commands are sent to the UR robot via TCP/IP protocol on a TCP socket. The commands are post-processed into URScript, and are either joint or linear interpolations (movej and movel, respectively). Gripper setting commands are also sent to the UR robot as the gripper is connected to the robot controller. The commands are sent together with the gripping distance and force. The IDS XS camera is interfaced through the uEye application programming interface (API) via serial communication on a USB (Universal Serial Bus) port. Similarly, the Arduino controller of the vibratory light table is accessed and controlled through serial communication on a USB port.

For metrology, the IDS XS camera resolution is set to 1280×720. The camera was calibrated, its intrinsic parameters (camera matrix and distortion parameters) are available. The camera settings are default, except for the autofocus and auto gain/shutter functions, which were turned off.

A series of experiments were run to evaluate the workcell operation capabilities. Specifically, four pieces from each of the three workpiece types were sorted and manipulated onto the corresponding four slot pallets. The task was ready, when all 12 workpieces were palletized. Then, the workpieces were fed back onto the light table, and the task was repeated. Altogether 10 tasks were performed in sequence, meaning 120 workpiece manipulation instances. For this, 129 image capturing were required, which means that the average picked workpiece per captured image is less than 1. This is caused by the lack of workpiece refilling, as the last few workpieces were less likely to get in a pickable pose, needing several rearrangement cycles. Overall, the development of the workcell was successful, and no failure occurred during operation. The experimental results are summarized in Table 1.

6.10. Reconfiguration

At this point, as the sorting and palletizing cell was developed together with the underlying DT, the application of the workcell for

Table 1
Experimental results with average online computation times on an Intel I7-6700 CPU @ 3.40GHz PC under Windows 10.

Manipulation success rate	(120/120) 100%
Image capture time	0.52 s
Marker detection and segmentation time per image	1.55 s
Workpiece and stable pose identification time per image	0.48 s
Workpiece localization time per image	0.25 s
Collision detection time per image	0.04 s
Total online planning time per image	2.32 s
Robot motion time per workpiece	11.72 s

different scenarios can be achieved through reconfiguration. This can include modifications such as re-tasking, introducing new workpiece types, changing equipment or modifying key poses. To implement such changes, depending on the scenario, model tools might need to be replaced, and the models and plans need to be updated along with the changing input data.

The workcell was reconfigured for a cableshoe clamping application, where cableshoes are fed into the fixture of an imitated press machine, to be joined with a cable. The reconfiguration was carried out in two steps, first the workpiece type was replaced, then the placing pose and cell environment was modified.

6.10.1. Workpiece change

The cableshoe has a complex geometry and material, but it is suitable for the same gripper, vibratory light table and metrology equipment. Reconfiguring for such a workpiece type involves little manual interaction. Information from the new workpiece is required in form of a digital 3D model, which was created via 3D scanning. The manual modeling includes the selection and setup of the grasp frame pair, after automated stable pose and graspable surface pair generation. Considering a similar 2,5D approach as for RC1, the path modeling only needs to be updated with the releasing configurations. The final notable manual modeling task is in case of the metrology model. Although the object identification database can be automatically created, the localizing features need to be selected and prepared manually. Finally, the tolerance model is updated with the corresponding dimensions.

As no infeasibilities were identified in the offline phase, the grasp plans and metrology models were tested in the physical workcell. Here, one feasibility issue occurred, as two stable poses were often indistinguishable from each other on the light table. The resolution of the reference database was improved, and thresholding parameters were adjusted to maximize identification reliability. Although in this case feasibility could be achieved with the present toolset, for other, less suitable workpieces additional equipment and tools can also be introduced for identification and localization purposes, such as machine learning based identification, or 3D imaging with point cloud-based fitting.

6.10.2. Placing workholder change

In the following step, the workcell environment and equipment was modified. The main change occurred in the placing workholder, as the pallets were replaced to a press and corresponding fixture. However, due to an external request, the UR5 robot was replaced with a UR10, additional obstacles were installed and the layout was modified as well. Unlike in the previous step, here the reconfiguration started with the update of the kinematic model and the placing pose. Due to the equipment change, the earlier 2,5D approach (and thus a single robot configuration) was no longer applicable, as the press does not allow vertical access to the placing workholder. This means that the grasp synthesis was needed to be recomputed based on both the picking and placing workholders and the path model was needed to be replaced, as simple joint interpolation was no longer suitable.

By combining the models of the picking and placing environment, workpiece and gripper, the grasp frame parameters were set so, that the manipulation could be performed in a feasible way with tilted last robot link axis. This eliminated the possibility of using a single robot configuration setup. Hence, due to the multiple possible seizing configurations, workpiece pickability was determined by checking each corresponding configuration candidate for the graspable workpieces. During path modeling, the order of candidates was determined with a GTSP based sequence planner, furthermore a PRM based model was set up with continuous 3D collision detection [77]. Using the kinematic model of the workcell, a PRM map was generated for the later online planning of path segments between the metrology and seizing approach, between the seizing retreat and releasing approach as well as between the releasing retreat and seizing approach configurations.

During the offline verification, no infeasibilities were identified. However, during the first online verification, feasibility issues occurred; due to the configuration changes, cable tangling caused infeasible robot motions (typically in the last robot joint). This was tackled by excluding critical configurations from the path model, reducing the possible key configurations at the path endpoints. Then, the twin closeness also proved to be insufficient, due to imprecise physical rearrangement of the cell; the robot motions risked collision, and the releasing configuration was inaccurate. Using the earlier calibration method, this time the whole workcell was calibrated in the kinematic model, including every obstacle present. Finally, the PRM map was updated based on the calibrated kinematic model, and thus sufficient twin closeness was achieved. The reconfigured system can be seen during operation in https://youtu.be/9novNg8slN4.

6.11. Other application scenarios

To represent how the proposed methodology applies in further cases, different pick-and-place scenarios are collected here. Table 2 summarizes the offline and partly online applied tools and models, when utilizing DT based development in the selected scenarios.

This table is intended to give an impression about the tasks of the offline and online planning phases, as well as regarding the selection of model and tool types. In the mentioned cases, by following the proposed development and verification steps, the on-site and overall workload can be reduced while achieving feasible solutions.

Table 2
Summary of tools in the offline and online planning phases in case of pick-and-place scenarios.

Scenario	Main task-specific uncertainties	Grasp synthesis algorithm	Grasp selector	Approach-retreat planner	Sequence planner	Motion planner	Collision query	Servo model	Metrology model
Standard pick-and-place with reconfigurations	-	○	○	○	○	○	○	-	-
Picking from conveyor [11]	seizing config. (2D)	○	○	○	-[2]	●	○	-	●
Semi-structured bin-picking [3]	seizing config. (2,5D)	○	●	○	●	●	●	-	●
Unstructured bin-picking [10]	seizing config. (3D)	○	●	●	-[2]	●	●	-	●
Precise assembly in motion [12]	releasing config. (2,5D)	○	○	○[1]	-	●	○	●	●
Standard peg-in-hole assembly	releasing config. (3D)	○	○	○[1]	-	○	○	●	-
Sorting in warehouse [19]	grasp frame, seizing config. (3D)	●	●	●	-[2]	●	●	-	●

○ offline applied, ● partly online applied, - not present.
[1] applied for seizing only, releasing is solved with the servo model.
[2] sequence planner is not required, as in each online phase, the operation is calculated for a single workpiece.

7. Conclusion and future work

In this paper, a comprehensive, generic development methodology was proposed for robotic pick-and-place scenarios to achieve feasible and accurate applications. The approach aims to facilitate the development process and reduce the on-site workload, and is built upon the Digital Twin (DT) concept.

The methodology is applicable for the majority of conventional and flexible industrial robotic pick-and-place workcells. The proposed workflow guides the involved experts along the development phases, from design to commissioning, and also in case of later adjustment, maintenance or reconfiguration steps. The methodology covers basic and advanced tools and approaches required for most industrial pick-and-place applications. The applied functional blocks are the kinematic, grasp, path, servo, metrology and tolerance models. The workflow includes iterative workcell refinement, driven by the verification steps realized on multiple occasions. The DT and the results of the offline planning are assessed for feasibility, whereas the physical workcell is tested for both feasibility and twin closeness. Infeasibilities are resolved through workcell refinement, and twin closeness is improved via calibration or refinement. When feasibility and sufficient twin closeness are achieved, the digital pick-and-place operation can be accurately realized in the physical workcell.

The methodology targets applications for which digital model based offline planning and autonomous online planning is required, and on-site, intuitive robot programming methods are inapplicable or inefficient. These include the original design or re-design of flexible pick-and-place cells, frequently modified conventional or flexible cells and advanced operations requiring offline planning or optimization.

The methodology was demonstrated through the development of a semi-structured pick-and-place task, which was realized in an experimental workcell. To present the practicality of the workflow, its comparison to industrial practices in robotic workcell development, along with further use cases, will be part of a following study. Future work will focus on the extension of the present methodology with currently missing modules, particularly with a pick-and-place specific tolerance model. On the other hand, there are various robotic assembly types, in which a similar methodology could be beneficially used. Therefore, a further research direction is the generalization of the approach to different assembly types, such as screwing or spot welding.

Authors' statement

Gábor Erdős: Conceptualization, Software (LinkageDesigner), Writing & Review, Supervision

Bence Tipary: Conceptualization, Software (integration and implementation), Writing - Original draft preparation & Editing, Methodology, Investigation, Visualization

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgment

The authors thank András Kovács and László Zahorán for their contributions to the implementation of the collision detection, and path planning algorithms used in the paper. Work in this paper has been in part funded under project number ED_18-22018-0006, supported by the National Research, Development and Innovation Fund of Hungary, financed under the (publicly funded) funding scheme according to Section 13. §(2) of the Scientific Research, Development and Innovation Act, and in part by the Ministry for Innovation and Technology and the National Research, Development and Innovation Office within the framework of the National Lab for Autonomous Systems.

References

[1] J.W.S. Chong, S.K. Ong, A.Y.C. Nee, K. Youcef-Youmi, Robot programming using augmented reality: an interactive method for planning collision-free paths, Robot. Comput.-Integr. Manuf. 25 (2009) 689–701, https://doi.org/10.1016/j.rcim.2008.05.002.

[2] G. Erdős, I. Paniti, B. Tipary, Transformation of robotic workcells to Digital Twins, CIRP Ann. 69 (2020) 149–152, https://doi.org/10.1016/j.cirp.2020.03.003.

[3] D. Holz, A. Topalidou-Kyniazopoulou, J. Stuckler, S. Behnke, Real-time object detection, localization and verification for fast robotic depalletizing, in: 2015 IEEERSJ Int. Conf. Intell. Robots Syst. IROS, IEEE, Hamburg, Germany, 2015, pp. 1459–1466, https://doi.org/10.1109/IROS.2015.7353560.

[4] D. Buchholz, Bin-picking—5 decades of research, in: D. Buchholz (Ed.), Bin-Pick. New Approaches Class. Probl. 44, Springer, Cham, 2016, pp. 3–12, https://doi.org/10.1007/978-3-319-26500-1_2, editor.

[5] G. Sohlenius, Concurrent engineering, CIRP Ann. 41 (1992) 645–655, https://doi.org/10.1016/S0007-8506(07)63251-X.

[6] R. Rosen, G. von Wichert, G. Lo, K.D. Bettenhausen, About the importance of autonomy and Digital Twins for the future of manufacturing, IFAC-PapersOnLine 48 (2015) 567–572, https://doi.org/10.1016/j.ifacol.2015.06.141.

[7] Y. Lu, C. Liu, K.I.-K. Wang, H. Huang, X. Xu, Digital Twin-driven smart manufacturing: connotation, reference model, applications and research issues, Robot. Comput.-Integr. Manuf. 61 (2020), 101837, https://doi.org/10.1016/j.rcim.2019.101837.

[8] E. Negri, L. Fumagalli, M. Macchi, A review of the roles of Digital Twin in CPS-based production systems, Proc. Manuf. 11 (2017) 939–948, https://doi.org/10.1016/j.promfg.2017.07.198.

[9] B. Schleich, N. Anwer, L. Mathieu, S. Wartzack, Shaping the Digital Twin for design and production engineering, CIRP Ann. 66 (2017) 141–144, https://doi.org/10.1016/j.cirp.2017.04.040.

[10] L.-P. Ellekilde, H.G. Petersen, Motion planning efficient trajectories for industrial bin-picking, Int. J. Robot. Res. 32 (2013) 991–1004, https://doi.org/10.1177/0278364913487237.

[11] Q. Chen, C. Zhang, H. Ni, X. Liang, H. Wang, T. Hu, Trajectory planning method of robot sorting system based on S-shaped acceleration/deceleration algorithm, Int. J. Adv. Robot. Syst. 15 (2018) 1–13, https://doi.org/10.1177/1729881418813805.

[12] G. Reinhart, J. Werner, Flexible automation for the assembly in motion, CIP Ann. 56 (2007) 25–28, https://doi.org/10.1016/j.cirp.2007.05.008.

[13] M. Berger, G. Bachler, S. Scherer, Vision guided bin picking and mounting in a flexible assembly cell, in: R. Loganantharaj, G. Palm, M. Ali (Eds.), Intell. Probl. Solving Methodol. Approaches, IEA/AIE, Lecture Notes in Computer Science, 1821, Springer, Berlin, Heidelberg, 2000, pp. 109–117, https://doi.org/10.1007/3-540-45049-1_14.

[14] C.R. Tudorie, Different approaches in feeding of a flexible manufacturing cell, in: N. Ando, S. Balakirsky, T. Hemker, M. Reggiani, O. von Stryk (Eds.), Simul. Model. Program. Auton. Robots, SIMPAR, Lecture Notes in Computer Science, 6472, Springer Berlin Heidelberg, Berlin, Heidelberg, 2010, pp. 509–520, https://doi.org/10.1007/978-3-642-17319-6_46.

[15] J.-K. Oh, S. Lee, C.-H. Lee, Stereo vision based automation for a bin-picking solution, Int. J. Control Autom. Syst. 10 (2012) 362–373, https://doi.org/10.1007/s12555-012-0216-9.

[16] N.B. Kumbla, S. Thakar, K.N. Kaipa, J. Marvel, S.K. Gupta, Simulation based on-line evaluation of singulation plans to handle perception uncertainty in robotic bin picking. 2017 ASME 12th Int. Manuf. Sci. Eng. Conf., in: Manuf. Equip. Syst., 3, ASME, Los Angeles, California, USA, 2017, pp. 1–12, https://doi.org/10.1115/MSEC2017-2955.

[17] N. Correll, K.E. Bekris, D. Berenson, O. Brock, A. Causo, K. Hauser, et al., Analysis and observations from the first amazon picking challenge, IEEE Trans. Autom. Sci. Eng. 15 (2018) 172–188, https://doi.org/10.1109/TASE.2016.2600527.

[18] A. Cowley, B. Cohen, W. Marshall, C.J. Taylor, M. Likhachev, Perception and motion planning for pick-and-place of dynamic objects, in: 2013 IEEERSJ Int. Conf. Intell. Robots Syst. IROS, IEEE, Tokyo, 2013, pp. 816–823, https://doi.org/10.1109/IROS.2013.6696445.

[19] M. Fujita, Y. Domae, A. Noda, G.A. Garcia Ricardez, T. Nagatani, A. Zeng, et al., What are the important technologies for bin picking? Technology analysis of robots in competitions based on a set of performance metrics, Adv. Robot. 34 (2020) 560–574.

[20] S. D'Avella, P. Tripicchio, C.A. Avizzano, A study on picking objects in cluttered environments: exploiting depth features for a custom low-cost universal jamming gripper, Robot. Comput.-Integr. Manuf. 63 (2020), 101888, https://doi.org/10.1016/j.rcim.2019.101888.

[21] K. Harada, T. Tsuji, K. Nagata, N. Yamanobe, H. Onda, Validating an object placement planner for robotic pick-and-place tasks, Robot. Auton. Syst. 62 (2014) 1463–1477, https://doi.org/10.1016/j.robot.2014.05.014.

[22] P.Y. Chua, T. Ilschner, D.G. Caldwell, Robotic manipulation of food products – a review, Ind. Robot. Int. J. 30 (2003) 345–354, https://doi.org/10.1108/01439910310479612.

[23] A. Downs, Z. Kootbally, W. Harrison, P. Pilliptchak, B. Antonishek, M. Aksu, et al., Assessing industrial robot agility through international competitions, Robot. Comput.-Integr. Manuf. 70 (2021), 102113, https://doi.org/10.1016/j.rcim.2020.102113.

[24] V. Krueger, F. Rovida, B. Grossmann, R. Petrick, M. Crosby, A. Charzoule, et al., Testing the vertical and cyber-physical integration of cognitive robots in

manufacturing, Robot. Comput.-Integr. Manuf. 57 (2019) 213–229, https://doi.org/10.1016/j.rcim.2018.11.011.
[25] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, et al.. ICRA Workshop on Open Source Software, ROS: an Open-Source Robot Operating System, 3, IEEE, Kobe, Japan, 2009, p. 5.
[26] R. Diankov, Automated Construction of Robotic Manipulation Programs, PhD Thesis, Carnegie Mellon University, Pittsburgh, PA, 2010.
[27] P. Fritzson, Principles of Object-Oriented Modeling and Simulation with Modelica 2.1, IEEE, 2010, https://doi.org/10.1109/9780470545669.
[28] P.I. Corke, A robotics toolbox for MATLAB, IEEE Robot. Autom. Mag. 3 (1996) 24–32, https://doi.org/10.1109/100.486658.
[29] A.C. Kak, K.L. Boyer, C.H. Chen, R.J. Safranek, H.S. Yang, A knowledge-based robotic assembly cell, IEEE Expert 1 (1986) 63–83, https://doi.org/10.1109/MEX.1986.5006501.
[30] P. Freedman, C. Michaud, G. Carayannis, A. Malowany, A database design for the runtime environment of a robotic workcell, Robot. Comput.-Integr. Manuf. 5 (1989) 21–31, https://doi.org/10.1016/0736-5845(89)90027-6.
[31] R. Bernardo, R. Farinha, P.J.S. Gonçalves, Knowledge and tasks representation for an industrial robotic application, In: Ollero A, Sanfeliu A, Montano L, Lau N, Cardeira C, editors, in: ROBOT 2017 Third Iber. Robot. Conf. 693, Springer International Publishing, Cham, 2018, pp. 441–451, https://doi.org/10.1007/978-3-319-70833-1_36.
[32] L. Wang, S. Keshavarzmanesh, H.-Y. Feng, A function block based approach for increasing adaptability of assembly planning and control, Int. J. Prod. Res. 49 (2011) 4903–4924, https://doi.org/10.1080/00207543.2010.501827.
[33] N.F. Edmondson, A.H. Redford, Generic flexible assembly system design, Assem. Autom. 22 (2002) 139–152, https://doi.org/10.1108/01445150210423189.
[34] V. Villani, F. Pini, F. Leali, C. Secchi, C. Fantuzzi, Survey on human-robot interaction for robot programming in industrial applications, IFAC-PapersOnLine 51 (2018) 66–71, https://doi.org/10.1016/j.ifacol.2018.08.236.
[35] A. Nubiola, Contribution to Improving the Accuracy of Serial Robots, PhD Thesis, École de Technologie Supérieure, 2014.
[36] D. Jones, C. Snider, A. Nassehi, J. Yon, B. Hicks, Characterising the Digital Twin: a systematic literature Review, CIRP J. Manuf. Sci. Technol. 29 (2020) 36–52, https://doi.org/10.1016/j.cirpj.2020.02.002.
[37] S. Boschert, R. Rosen, Digital Twin-the simulation aspect, in: P. Hehenberger, D. Bradley (Eds.), Mechatron. Futur., Springer, Cham, 2016, pp. 59–74, https://doi.org/10.1007/978-3-319-32156-1_5.
[38] L. Pérez, S. Rodríguez-Jiménez, N. Rodríguez, R. Usamentiaga, D.F. García, Digital Twin and virtual reality based methodology for multi-robot manufacturing cell commissioning, Appl. Sci. 10 (2020) 3633, https://doi.org/10.3390/app10103633.
[39] P. Aivaliotis, K. Georgoulias, Z. Arkouli, S. Makris, Methodology for enabling Digital Twin using advanced physics-based modeling in predictive maintenance, Proc. CIRP 81 (2019) 417–422, https://doi.org/10.1016/j.procir.2019.03.072.
[40] J. Leng, Q. Liu, S. Ye, J. Jing, Y. Wang, C. Zhang, et al., Digital Twin-driven rapid reconfiguration of the automated manufacturing system via an open architecture model, Robot. Comput.-Integr. Manuf. 63 (2020), 101895, https://doi.org/10.1016/j.rcim.2019.101895.
[41] M. Priggemeyer, D. Losch, J. Roßmann, Interactive calibration and visual programming of reconfigurable robotic workcells, in: IEEEASME Int. Conf. Adv. Intell. Mechatron, AIM, IEEE, Auckland, New Zealand, 2018, pp. 1396–1401, https://doi.org/10.1109/AIM.2018.8452707.
[42] T.Y. Melesse, V.D. Pasquale, S. Riemma, Digital Twin models in industrial operations: a systematic literature review, Proc. Manuf. 42 (2020) 267–272, https://doi.org/10.1016/j.promfg.2020.02.084.
[43] K.-H. John, M. Tiegelkamp, IEC 61131-3: Programming Industrial Automation Systems, Springer, Berlin, Heidelberg, 2001, https://doi.org/10.1007/978-3-662-07847-1.
[44] B. Siciliano, O. Khatib, Springer Handbook of Robotics, Springer, 2016.
[45] Garage, W., XML robot description format (URDF). https://wiki.ros.org/urdf/XML (accessed July 15, 2020).
[46] Erdős, G., Linkagedesigner, the mechanism prototyping system website, 2005. http://www.linkagedesigner.com/ (accessed July 19, 2020).
[47] S. Trenkel, R. Weller, G. Zachmann, A benchmarking suite for static collision detection algorithms, in: Proc. Int. Conf. Cent. Eur. Comput. Graph. Vis. Comput. Vis., Václav Skala - UNION Agency, 2007, pp. 265–270.
[48] A. Sahbani, S. El-Khoury, P. Bidaud, An overview of 3D object grasp synthesis algorithms, Robot. Auton. Syst. 60 (2012) 326–336, https://doi.org/10.1016/j.robot.2011.07.016.
[49] A. Bicchi, V. Kumar, Robotic grasping and contact: a review, in: Proc. 2000 ICRA Millenn. Conf. IEEE Int. Conf. Robot. Autom. Symp. Proc. Cat No00CH37065 1, IEEE, San Francisco, CA, USA, 2000, pp. 348–353, https://doi.org/10.1109/ROBOT.2000.844081.
[50] J. Bohg, A. Morales, T. Asfour, D. Kragic, Data-driven grasp synthesis—a survey, IEEE Trans. Robot. 30 (2014) 289–309, https://doi.org/10.1109/TRO.2013.2289018.
[51] A.T. Miller, P.K. Allen, GraspIt! a Versatile simulator for robotic grasping, IEEE Robot. Autom. Mag. 11 (2004) 110–122, https://doi.org/10.1109/MRA.2004.1371616.
[52] J.P. Carvalho de Souza, C.M. Costa, L.F. Rocha, R. Arrais, A.P. Moreira, E.J.S. Pires, et al., Reconfigurable grasp planning pipeline with grasp synthesis and selection applied to picking operations in aerospace factories, Robot. Comput.-Integr. Manuf. 67 (2021), 102032, https://doi.org/10.1016/j.rcim.2020.102032.
[53] S. Alatartsev, S. Stellmacher, F. Ortmeier, Robotic task sequencing problem: a survey, J. Intell. Robot. Syst. 80 (2015) 279–298, https://doi.org/10.1007/s10846-015-0190-6.
[54] G. Laporte, H. Mercure, Y. Nobert, Generalized travelling salesman problem through N sets of nodes: the asymmetrical case, Discrete Appl. Math. 18 (1987) 185–197, https://doi.org/10.1016/0166-218X(87)90020-5.
[55] F. Suárez-Ruiz, T.S. Lembono, Q.-C. Pham, RoboTSP–a fast solution to the robotic task sequencing problem, in: 2018 IEEE Int. Conf. Robot. Autom. ICRA, IEEE, Brisbane, QLD, 2018, pp. 1611–1616, https://doi.org/10.1109/ICRA.2018.8460581.
[56] S.M. LaValle, J.J. Kuffner, Rapidly-exploring random trees: progress and prospects, Algorithm. Comput. Robot. New Dir. 5 (2001) 293–308.
[57] L.E. Kavraki, P. Svestka, J.-C. Latombe, M.H. Overmars, Probabilistic roadmaps for path planning in high-dimensional configuration spaces, IEEE Trans. Robot. Autom. 12 (1996) 566–580, https://doi.org/10.1109/70.508439.
[58] E. Masehian, D. Sedighizadeh, Classic and heuristic approaches in robot motion planning-a chronological review, World Acad. Sci. Eng. Technol. 23 (2007) 101–106.
[59] RoboDK: Simulator for industrial robots and offline programming. https://robodk.com/ (accessed July 15, 2020).
[60] Koenig, N., Howard, A., Gazebo-3D multiple robot simulator with dynamics. http://gazebosim.org/ (accessed July 20, 2020).
[61] S. Chitta, I. Sucan, S. Cousins, MoveIt! [ROS Topics], IEEE Robot. Autom. Mag. 19 (2012) 18–19, https://doi.org/10.1109/MRA.2011.2181749.
[62] S. Chen, Y. Li, N.M. Kwok, Active vision in robotic systems: a survey of recent developments, Int. J. Robot. Res. 30 (2011) 1343–1377, https://doi.org/10.1177/0278364911410755.
[63] R. Bajcsy, Y. Aloimonos, J.K. Tsotsos, Revisiting active perception, Auton. Robots 42 (2018) 177–196, https://doi.org/10.1007/s10514-017-9615-3.
[64] F. Chaumette, S. Hutchinson, Visual servo control. II. Advanced approaches [tutorial], IEEE Robot. Autom. Mag. 14 (2007) 109–118, https://doi.org/10.1109/MRA.2007.339609.
[65] F. Chaumette, S. Hutchinson, Visual servo control. I. Basic approaches, IEEE Robot. Autom. Mag. 13 (2006) 82–90, https://doi.org/10.1109/MRA.2006.250573.
[66] J. Baeten, H. Bruyninckx, J. De Schutter, Integrated vision/force robotic servoing in the task frame formalism, Int. J. Robot. Res. 22 (2003) 941–954, https://doi.org/10.1177/027836490302210010.
[67] Q. Li, C. Schürmann, R. Haschke, H. Ritter, A control framework for tactile servoing, Robot. Sci. Syst. IX (2013) 1–8, https://doi.org/10.15607/RSS.2013.IX.045.
[68] M. Kazemi, K. Gupta, M. Mehrandezh, Path-planning for visual servoing: a review and issues, in: G. Chesi, K. Hashimoto (Eds.), Vis. Servoing Adv. Numer. Methods, Lecture Notes in Control and Information Sciences, 401, Springer London, London, 2010, pp. 189–207, https://doi.org/10.1007/978-1-84996-089-2_11.
[69] L. Pérez, Í. Rodríguez, N. Rodríguez, R. Usamentiaga, D.F. García, Robot guidance using machine vision techniques in industrial environments: a comparative review, Sensors 16 (2016) 335.
[70] Ruiz-del-Solar, J., Loncomilla, P., Soto, N. A survey on deep learning methods for robot vision. ArXiv180310862 Cs 2018:1–43.
[71] Du, G., Wang, K., Lian, S., Zhao, K. Vision-based robotic grasp detection from object localization, object pose estimation to grasp estimation: a review. ArXiv190506658 Cs 2020:1–24.
[72] Z. Kappassov, J.-A. Corrales, V. Perdereau, Tactile sensing in dexterous robot hands–review, Robot. Auton. Syst. 74 (2015) 195–220, https://doi.org/10.1016/j.robot.2015.07.015.
[73] E. Ferreras-Higuero, E. Leal-Muñoz, J. García de Jalón, E. Chacón, A. Vizán, Robot-process precision modeling for the improvement of productivity in flexible manufacturing cells, Robot. Comput.-Integr. Manuf. 65 (2020), 101966, https://doi.org/10.1016/j.rcim.2020.101966.
[74] H. Chen, S. Jin, Z. Li, X. Lai, A comprehensive study of three dimensional tolerance analysis methods, Comput.-Aided De.s 53 (2014) 1–13, https://doi.org/10.1016/j.cad.2014.02.014.
[75] B. Schleich, S. Wartzack, A quantitative comparison of tolerance analysis approaches for rigid mechanical assemblies, Proc. CIRP 43 (2016) 172–177, https://doi.org/10.1016/j.procir.2016.02.013.
[76] M. Fekula, G. Horváth, Determining stable equilibria of spatial objects and validating the results with drop simulation, Proc. CIRP 81 (2019) 316–321, https://doi.org/10.1016/j.procir.2019.03.055.
[77] L. Zahorán, A. Kovács, Efficient collision detection for path planning for industrial robots, Fister Jr. I, Brodnik A, Krnc M, in: Proc. 6th Stud. Comput. Sci. Res. Conf. StuCoSReC, University of Primorska Press, 2019, pp. 19–22, https://doi.org/10.26493/978-961-7055-82-5.19-22.