CIRP Annals - Manufacturing Technology 72 (2023) 101–104

Contents lists available at ScienceDirect

CIRP Annals - Manufacturing Technology

journal homepage: https://www.editorialmanager.com/CIRP/default.aspx

Design framework for managing the life cycle of Kinematic Digital Twins

Gábor Erdős (2)[a,b,*], Gergely Horváth[a,b]

[a] EPIC Centre of Excellence in Production Informatics and Control, Institute for Computer Science and Control (SZTAKI), Budapest, Hungary
[b] Department of Manufacturing Science and Engineering, Budapest University of Technology and Economics, Budapest, Hungary

ARTICLE INFO

Article history:
Available online 24 April 2023

Keywords:
Digital twin
Design method
Robot

ABSTRACT

Mass customisation calls for new innovative solutions such as the Digital Twin (DT) concept. DT enhances the classical sequential design-manufacture-usage life cycle into a more flexible concept that can handle the changes stemming from the physical realization of the nominal design. One of the greatest challenges of realizing a Kinematic Digital Twin (KinDT), where movement modelling is of upmost importance, is the synchronisation of the as-built model with the as-designed model. Consequently, this paper proposes a design framework to support the synchronisation and feedback information within the DT. The usage of the framework is demonstrated in two robotic applications.

1. Introduction

With the dawn of Industry 4.0 the need for higher autonomy in the manufacturing scene has surfaced, bringing forth the next generation of smart and autonomous systems. Mass customisation and the ever-growing need for flexible production systems call for new innovative solutions such as the Digital Twin (DT) concept. In the authors' interpretation, Digital Twin enhances the classical, sequential design-manufacture-usage life cycle of a product or process into a more flexible concept that can handle the changes stemming from the physical realization of the nominal design concepts. Stark et al. [1] define an 8-dimensional model where the *Digital Master* model defines the nominal design, while the *Digital Shadow* allows the integration of the physical product's characteristics. This multi-model concept is emphasized by Erkoyuncu et al. [2] and it defines DT as a set of models with varying granularity, describing various aspects across the life cycle. Also, for specific requirements, specialized Digital Twins exist. One such example is *System Digital Twin* [3], which is specialized for the Markovian representation of the system resources and their interactions.

The difference between the design and its realized physical artefact—the so called *as-built model*—was always known to engineers. Schleich et al. [4] introduced the *Skin Model concept* to capture the deviations resulting from manufacturing and assembly processes. Tolerancing, verification and validation of the nominal design are the traditional tools that ensure the product, service or system accomplishes its intended functional requirements [5]. The *Geometric Design and Tolerancing* (GD&T) approach ensures that even though the as-built model diverges from the designed model, it can be measured, verified and validated. If the as-built model becomes valid, then it meets its respective specification and fulfils its intended purpose [5]. In case it fails validation, it is considered as scrap product.

This GD&T approach is also utilized for classical, offline robotic process planning, at the workcell level. In this process planning approach, the as-designed cell components and movement trajectories of the robots are paired with their real counterparts in the realized, as-built cell. At the workcell level GD&T as-built cell can be measured and validated, but unlike the standard GD&T approach, the process failing validation is not wasted. Either a tedious iterative process is started to adjust the as-designed model to the as-built one or switching to on-line programming of the as-built cell happens. In both cases, process planning is re-executed completely for the as-built cell. Furthermore, this approach is not prepared for dynamic obstacles, autonomous real time planning of the robot motions or any features of the cell unknown at design time [6].

These new requirements call for a more flexible process planning and implementation approach, which could be supported by the so-called *Kinematic Digital Twin* (KinDT) model. KinDT is a specialized DT for system, where movement and geometric modelling is of utmost importance. One of the greatest challenges of realizing DT concepts is synchronising the as-built model with the as-designed model. This synchronisation is more than validation, because it requires more elaborate feedback information from the measurements than a single go/no-go that is present in case of validation. As Tomiyama et al. [7] recognized, the feedback mechanism between the as-built model and the as-designed model—in the context of smart product development—is not yet sufficiently understood and in that paper it is concerned as a future research topic.

To support the development of KinDT and to arm the next generation of robots with smart and advanced functions, a design framework is developed that supports the synchronisation and information feedback between the as-designed and as-built models. The as-designed model is principally the digitised design intent, while the as-built model is derived from the physical artefact through

* Corresponding author.
E-mail address: erdos.gabor@sztaki.mta.hu (G. Erdős).

https://doi.org/10.1016/j.cirp.2023.04.062

measurement. The two-branched, complexity based hierarchical framework, with multiple levels of geometrical representations is displayed in Fig. 1, where the hierarchical geometry representation addresses problems of varying complexity, while its multiple levels offer the necessary granularity to choose just the right tool for the particular synchronisation goal.

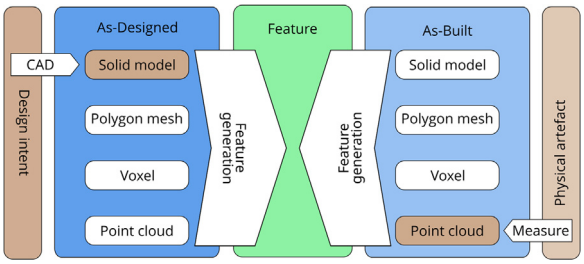

Fig. 1. The proposed framework.

2. Framework

The digital twin is a digital representation of a physical workcell that is periodically updated based on the constant flow of sensor data. However, the way sensor data fits into the digital twin is not always evident, since raw sensor data in itself is not necessarily valuable. What makes it valuable is interpretation that requires domain-specific knowledge about the task at hand. Design intent is the manifestation of such domain-specific knowledge. Formalization of this interpretation is carried out through a newly elaborated design framework that is visualized in Fig. 1. This framework establishes a connection between the functional requirements—the design intent—and the physical artefact, through the definition of *features*. Established connection is utilized for validation during design, construction and all along maintenance, repair and overhaul tasks, however, it is also used while operating a workcell.

The newly developed framework creates a link between measured data and design information or between the *as-designed* and *as-built* models of various objects. The distinction between as-built and as-designed models have been present in the engineering practice for a long time. Although these models are often deemed equal to either the design intent or the physical object, they are just simplified digital models of the infinitely more complex vision of the designing engineer or the real instance in question. In general, the as-designed model is created by applying CAD technology to the design intent, while the as-built model is generated by measuring the real world.

The proposed framework is built up of geometrical object *representations*, with *operations* carried out over the multiple levels of representations. Geometrical representation levels are mostly worked out in the literature of reverse engineering. The framework consists of two distinct branches with hierarchies of multiple levels of geometrical representations. Design usually creates solid models by utilizing CAD technology, while measurement—according to [8,9]—primarily produces some kind of point cloud. The hierarchical levels from highest complexity to lowest complexity are the following:

- Solid model representation.
- Polygonal mesh representation.
- Voxel representation.
- Point cloud representation.

3. Operations on the framework

The usefulness of the framework is tightly coupled with the corresponding operations that are available within its scope. Operations can be classified into the following distinct operation classes: *transformation, morphing, feature generation, feature distance, feature comparison* and *feature dictionary*. From now on, the phrases *As-Designed* and *As-Built*—as seen in Fig. 1, and later in Fig. 4 and Fig. 6—will appear capitalized as well. While as-designed and as-built refer to the abstract models, As-Designed and As-Built refer to the two branches of the framework.

3.1. Transformation

The first type of operations is transforming a geometric object from one level of the hierarchy of geometry representations to another, e.g., transforming from point cloud to voxel, or from solid model to polygon mesh. What is crucial in this case, is that levels are on the same representation branch, either on As-Designed or As-Built.

The literature of such operations is vast. Typical direction of transformation is from top-to-bottom on the As-Designed branch (from more complex to less complex), while bottom-to-top on the As-Built branch. On the As-Designed branch, decomposition of the original solid geometry into lower complexity levels of the hierarchy is straightforward in most situations (e.g., creating a polygon mesh from a solid geometry). This attribute is expected, as the higher complexity levels include much more detail than the lower levels, which means the decomposition and retrieval of lower complexity level data is possible. The other direction (creating higher complexity level representations from lower complexity level data) during design is not a common use-case, however, practices can be borrowed from the reverse engineering literature.

On the other branch—the As-Built branch—the general operation direction, opposed to the As-Designed branch, is from lower to higher complexity. As a consequence of digitisation, during measurement data point acquisition, the result of most measurements is some kind of point cloud [8,9]. Accordingly, on the As-Built branch, it is natural that the most common transformations are starting from point cloud data to create higher complexity level representations. The technical details of such operations are out of scope for this paper, however [8] gives a detailed summary on the topic.

3.2. Morphing

Operations that change the geometric representation, while not transforming it to another hierarchical representation level are called morphing operations.

An example of a morphing operation could be the merging of point cloud measurements created from different angles; filtering and removing voxels based on some criteria; simplifying or smoothing a polygon mesh by removing polygons, applying boolean operations on solid models. The input level of all these operations are the same as their output level.

3.3. Feature operations

The primary operations, however, are related to the features. Features form a bridge between the As-Designed and As-Built branches of the proposed framework. Relationship can be established between identical features. This means that the same feature should be generated on both branches of the framework and these features can be derived even from different geometric representations. In Section 4, examples are shown of how features are generated and used to bridge the gap between As-Designed and As-Built framework branches.

Feature distance is a quantitative or in certain cases a qualitative measure, representing the dissimilarity of two features. As features can be multi-dimensional vectors, it is not a hard requirement for the feature distance to be a single numeric value. Also, feature distance calculations have to be defined on a feature-by-feature basis.

Although a general definition for feature distance cannot be given, once properly defined, it can be used to carry out *feature comparison*, which is just a special use of the calculated distance. Generally, equality can be defined based on the distance between the two compared objects, with a predefined threshold, under which the equality of the two features is realized.

Feature comparison acts as the basis of classification against a *feature dictionary*. Feature dictionary is generated from the As-Designed branch. Then classification can be executed by calculating the distance between the feature generated from the As-Built branch and the features in the dictionary.

4. Applications

In this section, two application examples—workcell calibration and gesture control of a robot arm—are shown. These examples are used to demonstrate how to utilize the framework to generate the features in both branches and by comparing these features, how to extract meaningful information to operate the KinDT. The steps of the applications are marked with the symbols that can be seen in Fig. 2. Blunt arrow heads mark the start of the algorithm in either branch. The short but wide arrows, pointing up or down between the different geometrical representations, mark transformation operations. There are curved, forked-tail arrows, starting and finishing by pointing on the same level box, representing morphing operations, which change the underlying representation, but do not move it to another level. The three consecutive forked-tail arrows—pointing either from the As-Designed or the As-Built box to the Feature box—are feature generation operations. To ease the visual reception of the images, the white arrows, labelled "Feature generation", are removed. Generated feature names are printed inside the Feature box.

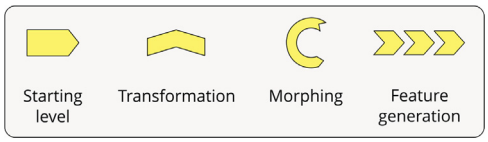

Fig. 2. Operation symbols.

4.1. Workcell calibration

Workcell calibration is the calculation of the transformation between a reference coordinate frame and the local coordinate frame of the workcell to be calibrated. To that end, the transformation must be established between the homogeneous transformation matrix representing the reference frame of the workcell and the one representing the local reference frame of a workcell component. This workcell component was a UR5 robot arm in that case. The whole process is demonstrated in detail in [10], while in this paper only the essentials are mentioned. Fig. 3 visualizes the main steps of the original algorithm, while Fig. 4 visualizes the same steps utilizing the framework.

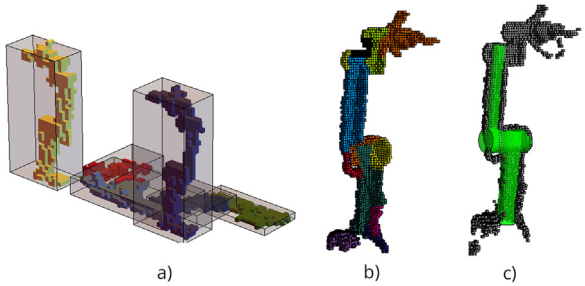

Fig. 3. Workcell calibration: a) bounding box fitting, b) segmentation, c) cylinder fitting.

A brief summary of the calibration process is the following. A Microsoft Kinect v2 sensor, which is a time-of-flight camera, is installed in the workcell and its detector origin is used as the workcell origin. During calibration, point cloud data is acquired from the workcell by the Kinect, containing the UR5 robot, among other objects in the cell. This point cloud is transformed into voxels from which the voxel connectivity graph (VCG) is built that is used to segment the cloud. The segmentation is based on the size of the axis aligned bounding box sides of the robot, as it can be seen in Fig. 3a.

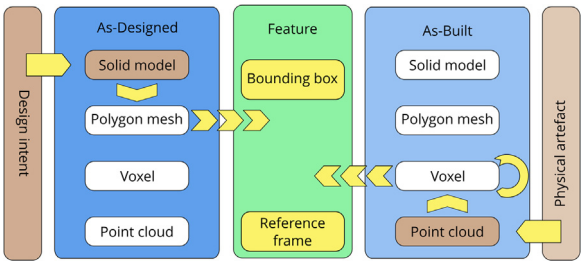

Fig. 4. Application of the design framework to workcell calibration.

After the points corresponding to the robot are found, a reference coordinate frame has to be fit on the robot. As the axes of the frame are tied to particular joints, these have to be identified first. To that end, another segmentation method is applied to find the various parts of the robot as seen in Fig. 3b. This segmentation is applied onto the voxels, throwing away ones that are not part of any links. Finally, cylinders are fit onto the joint voxels, laying out the direction of the reference frame, which can be compared to the reference frame of the STL model of the robot. This step can be seen in Fig. 3c.

In Fig. 4 the following operations are present:

- First, the operations to identify the UR5 robot in the scene.
- Second, the joints of the robot have to be recognized, as the frames are aligned with the direction of the joints. Once the robot and its joints are found, the correct transformation matrix can be calculated.

Point cloud data acquisition is visualized with the blunt arrow next to the Point cloud box on the As-Built branch. Converting the point cloud representation to voxel representation is a transformation operation—marked with the short but wide arrow—while the VCG generation, segmentation and box size calculation is generating the feature, which is the axis aligned bounding box. On the As-Designed branch, the bounding box is calculated, based on the STL model, generated from the original CAD model of the UR5 robot. So, the As-Designed branch starts at the solid model level (CAD model), utilizes a transformation to move to the polygon mesh layer (corresponding to the STL generation), and calculates the axis aligned bounding box, which is feature generation.

As both the input and the output of voxel segmentation are voxels, this is a morphing operation, carried out on the As-Built branch (see the markers, in Fig. 4 next to the Voxel level, in the As-Built box). Fitting cylinder on the joints and calculating the predefined frame from these cylinders is a feature generation on the As-Designed branch. The comparison is a feature distance calculation which—in this case—is defined as calculating the homogeneous transformation matrix transforming one reference frame to another.

4.2. Gesture control

The second application is gesture control of a collaborative robot. This application has been chosen to demonstrate the flexibility of the framework. The task has been carried out by capturing the pose of the human operator, interpreting it and controlling the robot. Central for such undertaking is the dictionary which translates gestures to robot commands. A solution for the task has been implemented in [11], visualization by the applied framework can be seen in Fig. 6.

To achieve gesture control, a feature dictionary, where human gestures are accumulated, has been built and each feature is linked to robot commands. Raising the right arm to stop or holding out a straight left arm to the side to signal "move to the left" is considered as the design intent of the gesture control system. Gestures are defined through the relative position of the arm of the human operator compared to the operator's body.

The general skeleton of the human body (see Fig. 5) is taken as a reference. The 26 joint skeleton model is given as a CAD model, so the As-Designed branch starts from a solid model. Joints are described by their spatial coordinate, so limb positions can be expressed by the normalized vector between the two corresponding joints. Calculating these limb vectors in the various gesture poses creates the basis of the dictionary. For every gesture the corresponding limb vector list—i.e., the feature—is derived and labelled. As this list incorporates every limb of the human body, complex gestures involving multiple body parts, e.g. two-handed gestures could also be handled.

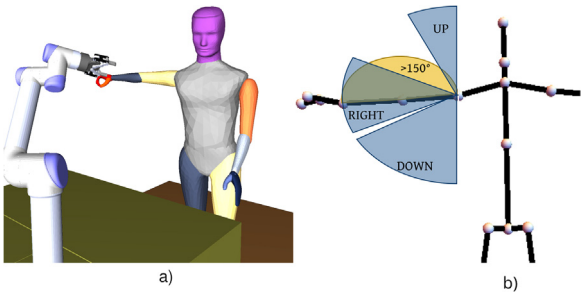

Fig. 5. Gesture control: a) example gesture on solid, As-Designed model, b) feature classification on the skeleton model.

The current position of the operator is captured using a Microsoft Kinect v2 sensor, generating point clouds (measurement in Fig. 6). From that point cloud (As-Built branch), the proprietary Microsoft Kinect SDK can extract the labelled position of 26 joints. The vectors between the respective joints create the skeleton information of the different body parts. Combination of the Kinect SDK joint position estimation and the vector subtraction is the feature generation operation, leading to a vector list feature in the same structure as the As-Designed branch.

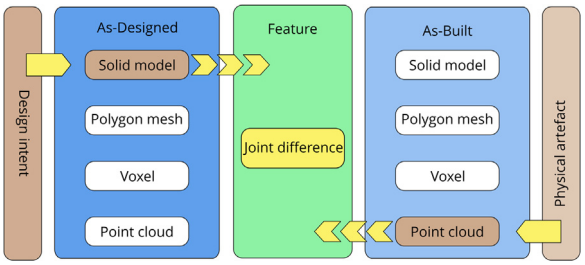

Fig. 6. Application of the design framework to gesture control.

The last step in gesture control is using feature distance and feature comparison operations to classify the feature (and the respective pose) from the As-Built branch, based on the dictionary created on the As-Designed branch. Fig. 5b shows example classification ranges of the relative joint positions to label the current arm pose.

5. Conclusion

In this paper, a generic design framework was proposed to synchronise the as-designed and as-build digital model of an artefact in a Kinematic Digital Twin. The as-designed and as-built models are defined by their digitisation processes. Digitising the design intent results in the as-designed model, while the realized physical artefact is digitised by measurements results in the as-built model. The proposed framework establishes a connections between the functional requirements—the design intent—and the realized part through the definition of features. The features form a bridge between the As-Designed and As-Built branch of the developed framework, that can be generated even from different geometrical representations, while feature comparison utilizes an abstract distance function. The framework was successfully applied to two different process planning applications, to prove its usability.

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.

Acknowledgements

This research has been supported by the ED_18-2-2018-0006 grant on an "Research on prime exploitation of the potential provided by the industrial digitalisation". The research is part of project no. TKP2021-NKTA-48 implemented with the support provided by the Ministry of Innovation and Technology of Hungary from the National Research, Development and Innovation Fund, financed under the TKP2021-NKTA funding scheme.

References

[1] Stark R, Fresemann C, Lindow K (2019) Development and Operation of Digital Twins for Technical Systems and Services. *CIRP Annals* 68:129–132.
[2] Erkoyuncu JA, del Amo IF, Ariansyah D, Bulka D, Vrabič R, Roy R (2020) A Design Framework for Adaptive Digital Twins. *CIRP Annals* 69:145–148.
[3] Magnanini MC, Tolio TAM (2021) A Model-Based Digital Twin to Support Responsive Manufacturing Systems. *CIRP Annals* 70:353–356.
[4] Schleich B, Anwer N, Mathieu L, Wartzack S (2014) Skin Model Shapes: A New Paradigm Shift for Geometric Variations Modelling in Mechanical Engineering. *Computer-Aided Design* 50:1–15.
[5] Maropoulos PG, Ceglarek D (2010) Design Verification and Validation in Product Lifecycle. *CIRP Annals* 59:740–759.
[6] Ceglarek D, Colledani M, Váncza J, Kim D-Y, Marine C, Kogel-Hollacher M, Mistry A, Bolognese L (2015) Rapid Deployment of Remote Laser Welding Processes in Automotive Assembly Systems. *CIRP Annals* 64:389–394.
[7] Tomiyama T, Lutters E, Stark R, Abramovici M (2019) Development Capabilities for Smart Products. *CIRP Annals* 68:727–750.
[8] Geng Z, Bidanda B (2017) Review of Reverse Engineering Systems – Current State of the Art. *Virtual Phys Prototyp* 12:161–172.
[9] Weckenmann A, Jiang X, Sommer K-D, Neuschaefer-Rube U, Seewig J, Shaw L, Estler T (2009) Multisensor Data Fusion in Dimensional Metrology. *CIRP Annals* 58:701–721.
[10] Horváth G, Erdős G (2017) Point Cloud Based Robot Cell Calibration. *CIRP Annals* 66:145–148.
[11] Horváth G, Erdős G (2017) Gesture Control of Cyber Physical Systems. *Procedia CIRP* 63:184–188.