International Journal on Interactive Design and Manufacturing (IJIDeM) (2019) 13:1061–1071
https://doi.org/10.1007/s12008-019-00567-6

ORIGINAL PAPER

An Augmented Reality approach to factory layout design embedding operation simulation

Andreas Kokkas[1] · **George-Christopher Vosniakos**[1]

Received: 16 January 2019 / Accepted: 16 April 2019 / Published online: 24 April 2019

Abstract
The subject of this work is layout planning of machinery, especially for Flexible Manufacturing Systems (FMS), using Augmented Reality tools. The goal is for the user to evaluate suggested layouts by taking into account non-measurable factors, such as operator experience, empirical or non-tacit knowledge and on-site impression. By reference to existing machinery (in the case study: CNC lathe and an industrial robot) the missing elements of a complete FMS cell (conveyor belts, automatic storage and retrieval system, other robots and CNC machines etc.) are super-imposed. The functional connection is enabled between the collaborating real-physical and the virtual machinery of the layout in the most convincing way, i.e., by simulation of the full production process involving movement, manipulation and processing of parts by real and virtual equipment in parallel and serial co-existence. This involves predefined scenarios that can be experienced exploiting alternative equipment in the same application and alternative layouts assessed in analogous applications. Static and dynamic simulation of the manufacturing cell offers the possibility of extensive analysis of the selected layout by walk-through navigation. The application is implemented in ARKit™ API tool and Unity3D™.

Keywords Augmented Reality · Facility layout · Dynamic simulation · Functional modelling · Machinery programming · Digital factory · Flexible manufacturing cell

1 Introduction

Facility layout of manufacturing systems addresses new factories as well as reconfiguration of existing factories. The latter results from product modifications resulting in process plan modifications, change of production volume, new product introduction, changes in batch sizes, replacement of old machinery etc. Traditional activities of facility layout planning involve, as a first step, collection of data relating to marketing, process plans, production plans and plant drawings. This is followed by material flow design, subject to safety, productivity and ergonomics rules, method study and associated workstation/workplace design, material handling and storage system design (technical logistics), and space allocation to workstations. Digital models and restricted physical prototypes often accompany these stages.

Facility layout is equally applicable to any type of manufacturing System, so much so to Flexible Manufacturing cells, since the latter require a closely thought out material handling system comprising conveyors, robots, Automated Guided Vehicles (AGVs), pallet changers etc. that needs to collaborate with computer numerically controlled (CNC) machine tools, other programmable devices, palletising stations, input/output and intermediate buffer stations, automated storage and retrieval systems (ASRS), as well as simpler automated or human-operated machines.

Quantitative approach to the problem involves mixed integer, quadratic assignment and other Operations Research based algorithms as well as, especially in case of elevated algorithmic complexity, Evolutionary Computation (genetic, simulated annealing) and other Artificial Intelligence based algorithms.

Virtual and, in particular, Augmented Reality (AR) techniques contribute to the solution of the problem in terms of capturing the real environment and allowing realistic

Electronic supplementary material The online version of this article (https://doi.org/10.1007/s12008-019-00567-6) contains supplementary material, which is available to authorized users.

✉ George-Christopher Vosniakos
vosniak@central.ntua.gr

[1] School of Mechanical Engineering, Manufacturing Technology Section, National Technical University of Athens, Heroon Polytehneiou 9, 15780 Athens, Greece

assessment of solutions, which have been obtained by either quantitative approaches or intuition and experience [1].

AR superimposes virtual models on the real world in real time through special devices such as tablets, mobile phones, head mounted displays etc. Its penetration is already high, but it is estimated that 1 bn devices will make use of it until 2020. In industry AR applications are becoming more widespread, having started with support of assembly, maintenance and training tasks [2]. In design of new production facilities AR is slowly starting to make an impact.

Until recently, development of AR applications especially for indoor use was mostly marker-based, exploiting the AR device's camera, whilst super-imposing virtual models was implemented by Software Development Kits (SDK) such as Vuforia™. However, in cases where the virtual model was substantially larger than the marker, the scene would have to be covered by a number of markers to prevent the tracking software from ignoring them due to either their size or their distance [3]. In fact, tracking was mediocre and discontinuous. Making use of objects as targets improved the situation, at the expense of the time required to scan each object in order to make it into a target and also the special equipment necessary. In any case, a substantial drawback was the necessity of the camera to view the target continuously and not move off it. The advent of new technology Visual Inertial Odometry (VIO), exploiting accelerometer, gyroscope and camera to compare consecutive frames, enabled independence from targets with only a planar surface being necessary to be viewed once initially and certainly not continuously as a reference [4].

VR-based simulation of a manufacturing process chain with an emphasis on the equipment, where all objects are virtual and no interaction with real equipment exists, is technologically quite mature, because the tools used in the past for videogame development have been easily extended to cover industrial design and simulation needs.

A Virtual Environment (VE) consists, in general terms, of a virtual world and its interaction with the user. Constructing such an environment involves modelling all pertinent objects, scenes and interaction functions. In recent years software platforms were used for these tasks, especially in the framework of CIM and their continuous improvement in functionality and reduction in cost led to virtual systems integration as a continuation of reality (Reality–Virtuality), often referred to as Virtual Manufacturing Systems [5]. Conventions and constraints applicable in traditional simulation are alleviated in VR whose use was further extended in the notion of Digital Factory [5]. Qualitative and quantitative analysis of material flows, collision risks and layout design belong to the pertinent functions supported [6]. Immersion into the Virtual Factory, e.g., by Head-Mounted Displays (HMD) was reported to be advantageous over non-immersive approaches and even more so are Augmented and Mixed Reality (AR/MR) [7, 8].

Using commercially available VR development software, industrial robots and machine tools have been modelled [9], emphasising on multimodal interaction [10, 11]. Inverse kinematics [12] and, more recently, dynamics analysis [13] have been linked to device programming approaches in AR/VR, [14]. Moving to design of manufacturing layouts as a whole, even small modifications may cause sizeable differences in terms of cost, time and flexibility [15–17]. Thus, AR/VR can enhance evaluation of suggested spatial layouts by taking into account non-measurable factors, such as operator experience, empirical or non-tacit knowledge and on-site impression [1, 8].

Overall, it seems that virtual models of factories involving machine tools and robots are useful in investigating their layout in 3D space, ergonomics of their use, their functionality and interaction of workers with them. Therefore, their modelling needs to be robust enough to allow credible simulation of their real counterparts. Interesting applications have been reported in designing and putting together manufacturing cells, in integrating and monitoring data in automotive industry, in studying alternative cell layouts in pointing out manufacturing system discrepancies etc. [18, 19]. In addition, pertinent frameworks have been proposed regarding machine tool design [20] factory design [21], product-process engineering [22] and evaluation of manufacturing process improvement [23].

Furthermore, increased challenges such as accuracy, production flow speed, remote controllability, reduced tolerance to errors prescribe a new era in production processes, which is currently dealt with by digitalisation in manufacturing termed Industry 4.0, VR/AR being a distinctive part of it [24].

Target-less AR (VIO) is used in this work in exploring different solutions for the layout of an FMS that is being developed by supplementing existing equipment, as necessary, in order to achieve the required functionality. Alternative solutions are generated and compared in this light. Whether obtained by some algorithm or numerical solver or not, the final layout needs to be assessed according to experience. This was so far achieved with CAD rough drawings or through 3D virtual environments. AR tools seem to be mature enough to allow live investigation of alternatives including constraints and issues that have not been taken into account in generating these alternatives.

Section 2 describes the FMS being designed. Section 3 introduces the development platform based on Unity3D™. Section 4 provides insight into the technicalities of developing the particular AR application, whereas Sect. 5 presents its use with the new VIO-based toolkit that has enabled it. Section 6 summarises the conclusions drawn from developing

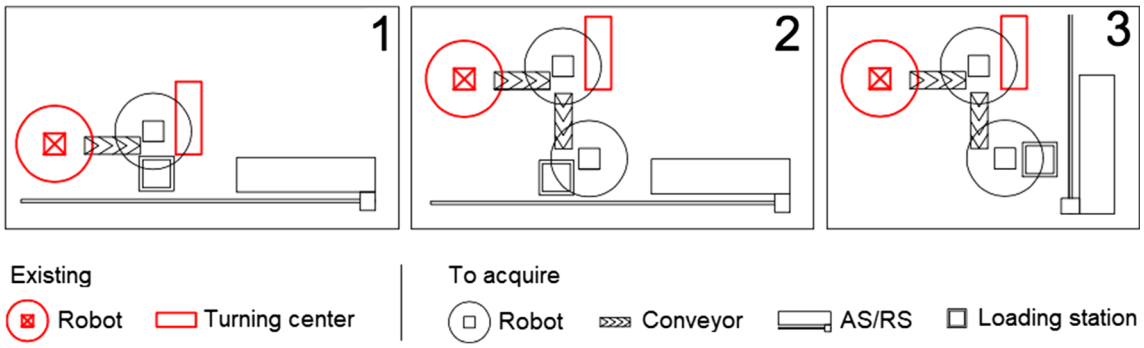

Fig. 1 Alternative layouts examined

and using this application and makes suggestions for future work.

2 FMC layouts

The flexible manufacturing cell being designed processes and handles axisymmetric parts. Parts are mainly processed on a 2-axis turning center, an EMCO COMPACT 5 CNC. This has been refurbished in terms of new stepping motors and new cnc controller, as well as an automatic tool post with a capacity of three tools.

The FMC also comprises an industrial robotic arm, namely a Staubli RX90L, possessing six axes, named as base, shoulder, arm, elbow, forearm and wrist, a reach of 1.1 m, a payload of 6 kg and a repeatability of 50 μm. Its end-effector is a pneumatically operated gripper. It can be programmed on- and off-line. On-line programming involves teaching of consecutive positions via a teaching pendant operated by the human user. Off-line programming involves writing a V+ program. A combination is often the best solution. In this way, the user can record the angle of each axis on the real setting and then import these values in the off-line generated program. Thus, the robot path is accurate and collision probability is eliminated. A particular program is written to handle a specific axisymmetric part.

The FMC needs an Automatic Storage and Retrieval System (AS/RS), a loading/unloading station and possibly a second robot and a conveyor belt as necessary in order to link the workstations together. These are to be tried as Virtual prototypes collaborating with the real turning center and robot that are available and that will keep their existing position.

The equipment selection and layout is based mainly on process and storage speed and not equipment cost. Factors that were taken into account are the ease of functioning of the rest of the equipment of the lab, ease of movement of handling equipment, comfort of workers, ease of monitoring and system extensibility. Mathematical modelling and optimisation was not pursued, given the constraints set in addition to the fact that there is only one machine tool. Thus, three alternative layouts were examined, see Fig. 1. These were assessed 'ad hoc' by two expert facility planning engineers, following factors such as obstruction of movement, overlap of the workspace of neighbouring machines etc., as in Table 1.

Table 1 *Ad hoc* assessment of alternative layouts (1: acceptable, 2: good)

Criterion	Layout 1	Layout 2	Layout 3
Work comfort	1	2	2
Personnel safety	1	2	1
Equipment cost	2	1	1

Thus, layout no 2 was selected in the first place and was set for further assessment. The scenario that was used to assess the preferred layouts refers to processing of a cylindrical part. The cylinder is grasped by the real robot and deposited onto the virtual conveyor. At the end of the conveyor the virtual robot grasps the part and inserts into the chuck of the real turning center, where it will be processed. After the end of processing, the same robot will pick the part from the turning center and deposit it to the second virtual conveyor at the end of which a third robot will pick it up and deposit it to a pallet. The AS/RS will collect the palletised part and accommodate it in a free storage place.

3 Development platform

Unity 3D has been being used for videogame development since 2005 and currently supports 27 platforms. The development environment possesses a graphical user interface, which enables project organisation, input of virtual models constructed by external software, writing code, adding sound and visual effects etc. through five basic windows: Toolbar, Project Window, Hierarchy Window, Inspector Window, SceneView. Unity3D™ supports plugins such as Apple's

ARKit™ and Vuforia™ SDK. Between the two languages that are available for constructing code, i.e., C# and Java, the former was used, due to its larger user community, steeper learning curve, full access to .NET Framework library, as well as ease of access to Unity libraries. Unity version 2017.2.0f3 was installed with "iOS Built Support" component allowing development of applications on iPhone and iTab [25].

ARKit™ is an Applications Programming Interface (API) for development platforms such as Unity 3D, which enables the deployment of AR applications through Xcode 9 (or more recent_ on Apple devices equipped with A9 processor and iOS 11 firmware (at least) using their cameras, graphics processor and motion sensors. ARKit exploits VIO technology, allowing the system to create 3D moving graphics that can record in real time in 6 degrees of freedom the movements of the device (mobile phone). The camera output is used for recognising optical milestones of the real world in order to define horizontal surfaces wich will be used as reference for superimposing 3D virtual models. The camera is also used for assessing the level of lighting before adapting the images to it appropriately to achieve photorealism in superimposing 3D models onto real world scenes, in particular following changes in device orientation–viewing angles [26].

4 Application development

4.1 Challenges

Co-existence and, in addition, 'collaboration' of real and virtual equipment is the main challenge of the application to be developed. This can be achieved in several ways, e.g., positioning targets on the real objects or using sensors that signal specific transition events. In this case time-based schedule programming was employed.

A second challenge might have been considered regarding positioning of virtual objects between the camera employed for third-person viewing and the virtual objects. A real object cannot intervene between the camera and the virtual objects without exploiting suitable software (e.g., for occlusion mask creation) or sensors (e.g., depth sensors, such as Kinect™. This process is known as occlusion and makes for more realistic and interactive applications. However, in this case occlusion was circumvented since from most viewing angles virtual content was 'on-top', i.e., between te camera and the real objects, due to the fact that real equipment was located perimetrically to the viewing point. Should this not have been the case, emerging devices supporting Mixed Reality, such as Hololens™ or Magic Leap™ would solve this problem.

A third challenge concerns computational power available with respect to the complexity of the modelled scenes and motion required. A MacBook Air™ was used possessing an Intel Core i5™ 1.6 GHz dual-core processor with 4 GB RAM and an Intel HD Graphics 6000™ card. Despite the descent graphics card, significant problems arose at the beginning, especially during movement of models, due to the simultaneous presence of a large number of High-Poly 3D models. Thus, decimation was pursued in order to clear details that were deemed unnecessary, at the same time trying to not degrade the final result.

4.2 Virtual models

Initially, the real manufacturing environment was modelled in Unity3D™, using primitives with correct dimensions, so that a good impression of the available space could be acquired, see Fig. 2. Virtual models of the equipment to be acquired were added following the asset importing process. Decimation was achieved using SolidWorks Composer™ aiming at reducing the number of triangles of the model's surface mesh by a typical 75%. Models were stored in 3D Studio™ format and transformed into Unity-compatible FBX format by being imported into 3ds Max™ software. In the conversion process all models lost their initial scaling, the scale factor being re-set to 0.0392. Note that Unity adopts 1 m as length unit.

4.2.1 Workpiece modelling

The workpiece starts as a cylinder with dimensions 40 mm × 130 mm ending up as a two-diameter rod, see Fig. 3a. Each unprocessed workpiece is transported on the first conveyor, following its linear movement, until it becomes child of the end effector of the robot that loads it into the turning center where it is processed. This is achieved by *Destroying* the *rawItem* and *Instantiating* the *readyItem* exactly in the same position on the chuck. In general, the child-parent relationship activation and deactivation is made use of widely, in order to alternate among the variety of loading/unloading states of the workpiece until this ends up in the AS/RS, see Fig. 3b.

4.2.2 Robotic arms

Different industrial robots have been modelled in order to be tried within the manufacturing system being designed. One of them is identical to the already available robot, a staubli RX90L, except that it possesses a vacuum gripper, see Fig. 4. The robot library is extensible, the available robots being accessed in the initial SetupState/Scene0.

In general, each robot link inherits the movement of its predecessors in the kinematic chain, whose members are related with parent–child relationship. Kinematics of each robot is defined according to the Denavit Hartenberg (DH) convention. All robots in the library possess rotary joints, hence

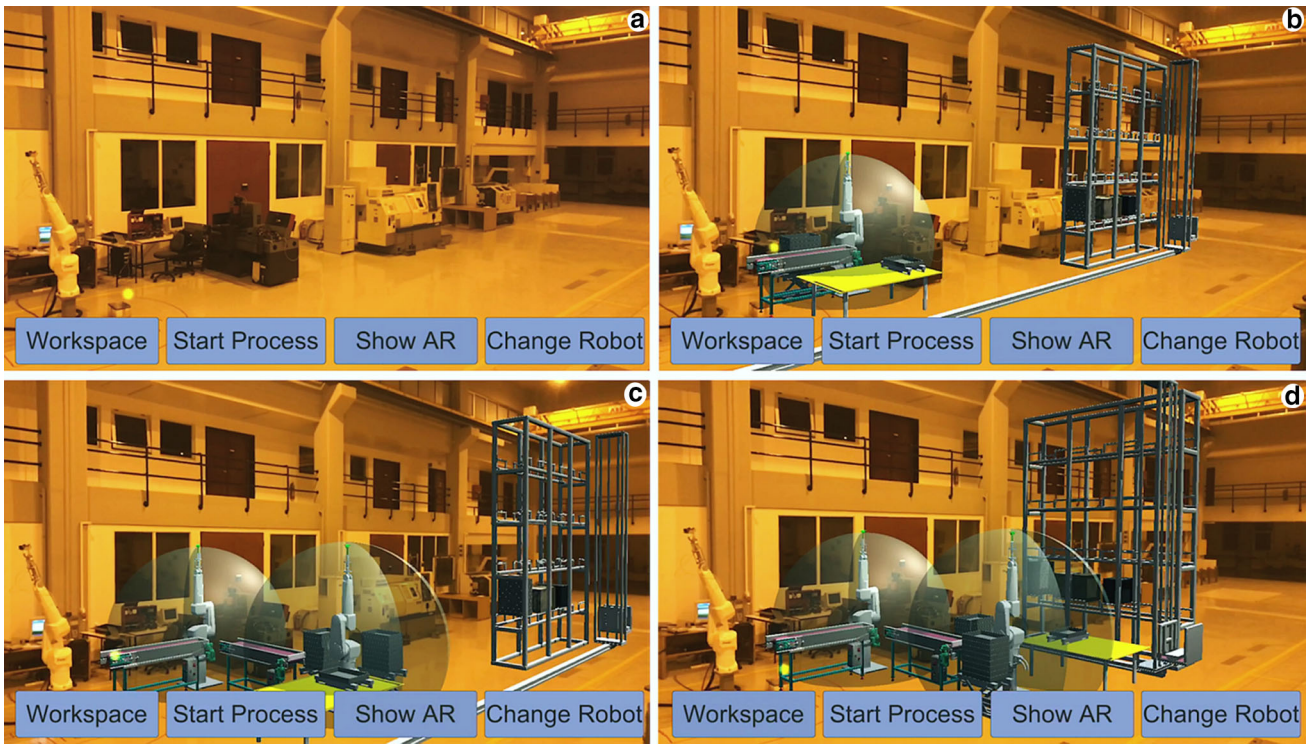

Fig. 2 Manufacturing environment **a** space available, **b** 1st layout, **c** 2nd layout, **d** 3rd layout

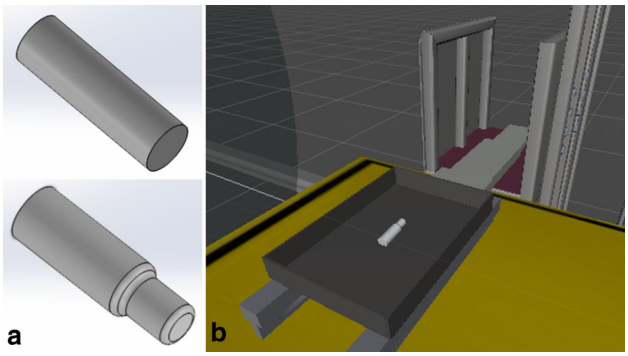

Fig. 3 Workpiece **a** raw and processed, **b** being transported on a tray

void game objects are defined in Unity corresponding to the respective links rotating about Z axis.

Forward kinematics was applied throughout as explained in detail in [27], i.e., for each link a 4×4 homogenous transformation matrix is defined incorporating the DH parameters to define the link's end position and orientation by reference to the previous link. Thus, multiplication of the successive matrices yields the position and orientation of the end effector of the robot with respect to the base (global coordinate system). The DH parameters of the robots employed can be found in [14, 27, 28].

For each robot i in the virtual scene the component script *FKMoveri* is attached to it, in order to interactively teach the necessary movements of the end effector in conjunction with the Play function of Unity3D™. The script involves 'coroutines'. A coroutine is a C# method that can interrupt its execution and return control to the system, in order to continue from where it did so, but in the next frame either immediately or after some definable delay. In our case, in order to move the robot from one point to the next, spherical interpolation was used through the function Quaternion.Slerp of the coroutine GoToTarget(). The coroutine takes as input the initial (rotationZiFrom) and final (rotationZiTo) angles of the six joints and the duration of the movement.

4.2.3 AS/RS

A 24 position ASRS is involved in this work, see Fig. 5a. The moving wagon slides horizontally and vertically, accommodating trays or pallets on a mechanism that can slide in- and out-wards to load and unload them to the respective shelf. Horizontal and vertical movements are accompanied by a characteristic alarm sound. Movement was implemented by asrsMotion script exploiting linear interpolation through function Lerp of coroutine MoveObject() (horizontal-vertical) and coroutine MoveCarrierOnAxisZ() (inwards-outwards). These coroutines take as input the initial and final positions for the respective axis of movement as well as its duration. Storage shelves were modelled as a 4×3 matrix, the actual points of interest, for loading and unloading, being assigned to empty game objects. Actual loading

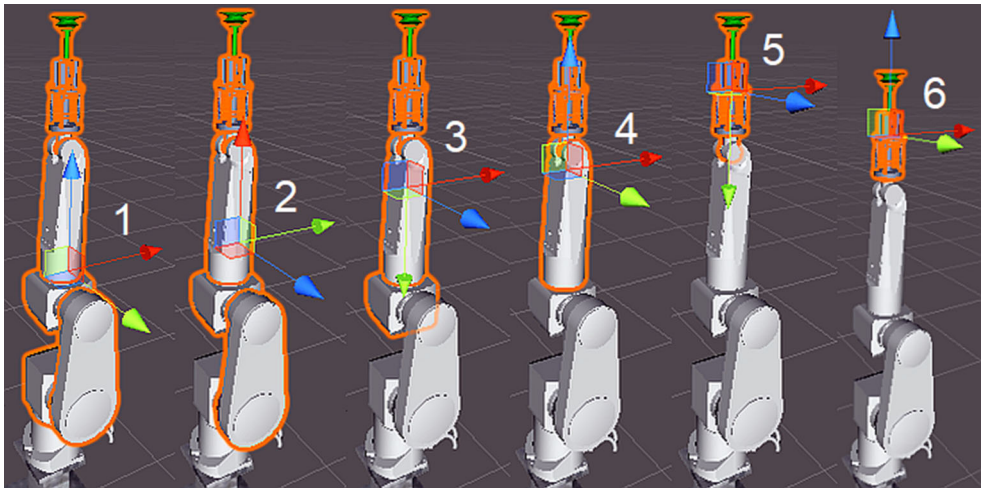

Fig. 4 Robot Staubli RX90L modelled with 6 links and respective local coordinate systems

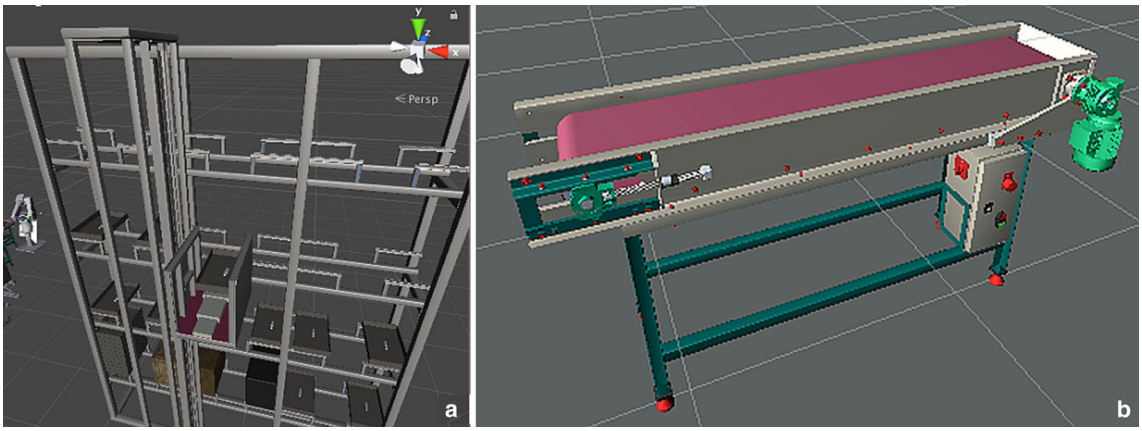

Fig. 5 **a** AS/RS, **b** conveyor module

and unloading are implemented by exploiting parent–child relationships. For instance, each tray that is ready for loading is assigned the tag ActiveTray; upon its loading method BoxLoading() untags it to allow a new tray to be loaded.

4.2.4 Conveyors

Conveyors consist of two rolls, one driving and one driven, with an endless belt around them, see Fig. 5b. Movement of virtual workpieces on conveyors is performed by linear interpolation through coroutine MoveItem in script components RawItemMotion και ReadyItemMotion for raw and processed workpieces, respectively, after its instantiation through method CreateItemPrefab(). In this way, the impression of conveyor band movement is created.

4.3 Application management

The Application consists of two scenes, namely Scene 0 and UnityARKitScene, and three States, namely Begin (application start), Setup (selection of robots) and Player (normal simulation), Fig. 6. Each state decides which scene is to appear in each frame. This is controlled by the StateManager script, which assigns control to the methods *StateUpdate*(), *ShowIt*() of the active State. Similarly, each state decides which camera will be activated in each Scene, selecting from a list of cameras stored in the script.

The overall management of the application at the User Interface level is performed by the *AnimationController* script. Four alternatives are offered, shown as buttons at the bottom of each image in Fig. 2. *ShowAR* button controls, on one hand, whether virtual content will appear in the scene or not, mainly in order to allow a general perspective of the empty space, whilst, on the other hand, enabling transition to different layouts of the machinery. Using *ChangeRobot* button different robots can be tried. *Workspace* button displays

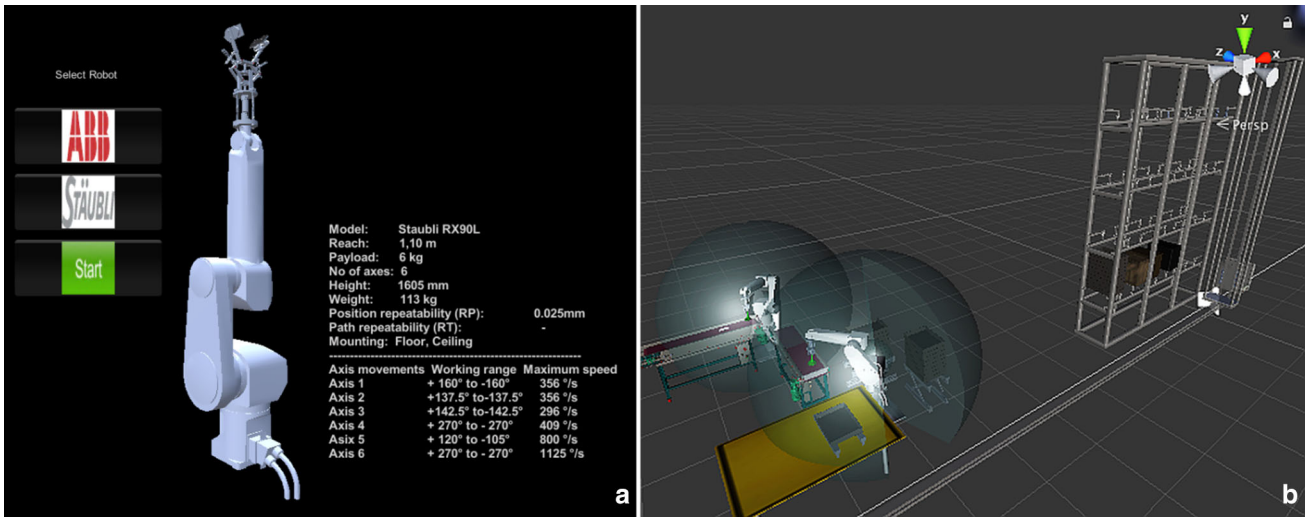

Fig. 6 **a** Setup state, Scene0, **b** Player state, UnityARKitScene

in Invisible texture the workspace of each robot, thereby giving an overview of the outer limits that the end-effector can reach. When *StartProcess* button is pressed neither *ChangeRobot* nor *ShowAR* button can be pressed.

The scripts *RawItemMotionManager*, *ReadyItemMotionManager* and *ASRSmotion* coordinate all movements in the Virtual Scene, making use of coroutines. Grouping the latter in 3 discrete scripts provides for better control in application development. In addition, by default, a coroutine needs to be completed before the next one starts. Therefore, managing all movement in one script would have resulted in a very specific movement sequence due to the inability of coroutines to overlap, i.e., two or more game objects would not be possible to move at the same moment in time.

RawItemMotionManager script manages the movement of the unprocessed workpiece, the first virtual robot and the processed workpiece until this is released onto the second conveyor. It starts upon pressing of the StartProcess button by making use of appropriate flag. Methods *ItemLoading*() and *ItemUnLoading*() make the workpiece a child of the robot's end effector and the turning center, respectively.

ReadyItemMotionManager script manages the movement of the processed workpiece, the second virtual robot and the loading of trays and workpieces at the pickup point (Table game object). This script starts only after *RawItemMotionManager* script has fetched the processed workpiece, as verified by Boolean variable *firstItemIsReadyAndPositioned*. Methods *ItemLoading*(), *ItemUnLoading*(), *BringTray*(), *ReleaseTray*(), *CreateNewTray*() exploit parent–child relationships, e.g., the first two manipulate the workpiece and the second virtual robot/the active tray, respectively.

Scripts *RawItemMotion, ReadyItemMotion, FKMover2* and *FKMover3* contain the coroutines involved in the *RawItemMotionManager* and *ReadyItemMotionManager* scripts. The aim was for the latter to contain only a specification of the manufacturing cell's programming and not the exact way or programming tools in which this was to be implemented. Note that coroutine *CreateItemPrefab* which creates the processed workpiece is 'overloaded' so that when called by script *RawItemMotionManager* the processed workpiece is created exactly at the same position and orientation with the unprocessed workpiece, as this is expected to happen in reality.

Communication of *ASRSmotion* script, with the other two high-level management scripts is done again through appropriate flags.

4.4 Implementation issues

Of the game objects that are available in the Hierarchy Panel after loading the ARKit™ plugin, the *HitCubeParent* and *ARCameraManager* are the most useful, whereas game objects *DirectionalLight* and *CameraParent* are activated by *StateManager* script at the same time with *UnityARKitScene* activation, in order to facilitate management of different cameras and light in each scene. *HitCubeParent* game object is parent of all objects appearing in *UnityARKitScene (Decimated* game objects). Script *UnityARHitTestExample* is component script of *HitCubeParent*, thus all its children game objects inherit the properties assigned by *UnityARHitTestExample* script.

UnityARHitTestExample script is responsible for moving virtual models onto horizontal surfaces which are determined by the device camera in ARKit™. *UnityARKitScene* will move the content of *Decimated* game objects to any point (*ARAnchor*) designated by the user's finger on the device's screen. To constrain placement of objects on a particular horizontal plane the script's Update method was modified to state

Table 2 Manufacturing cell functions and respective duration (*R* real, *V* virtual)

Element	Motion	Trigger	Duration (s)	Cycle (s)
Robot1-R	Workpiece from Grasp to Release		14	56
	From Release to next Release		42	
Conveyor1-V	Transport of unprocessed workpiece		10	56
Robot2-V	Workpiece grasping		1	
	Workpiece transport to turning center (part 1)		2	
	Workpiece transport to turning center (part 2)		3	
	Retract to allow processing		(2)	
	Approach		(2)	
	Waiting for loading		1	
	Workpiece grasping and transport to Conveyor-2 (part 1)		2	
	Return to waiting position		5	
	Workpiece transportation to Conveyor-2 (part 2)	Flag	3	
Turning center-R	Processing		29	
Conveyor2-V	Processed workpiece transportation		10	16
Robot3-V	Processed workpiece grasping	Flag	1	
	Processed workpiece transport to storage tray (part 1)		2	
	Processed workpiece transport to storage tray (part 2)		3	
AS/RS-V	Z move from home to loading position	Flag	2	14
	Z move from loading position to home		2	
	XY move from loading position to home		10	

interest only in surfaces determined by *GeneratePlanes* game object.

For iOS application development through the Unity 3D platform, having added the 3D models to *UnityARKitScene*, some modifications in the Build Settings tab should be make. So, Bundle Identifier (com.yourName.yourARapp), Camera Usage Description ('Augmented Reality') and the project name are set. The application is deployed for iOS through the Xcode environment.

5 Use of the application

The application was run on an Apple iPhone7™. It can be used to try a layout without constraints imposed by markers, observing at the same time its workings involving both real and virtual equipment. In this case, the real robot was programmed to pick and place the unprocessed workpiece and this programmed movement is directly observed though the device's camera. The user can select alternative virtual robots. Different layouts can be tried by re-programming the application, which is straightforward due to its modularity. However, the application follows a pre-planned time-schedule, as shown in Table 2.

Synchronising real and virtual models is based on the exact moment at which the workpiece is picked up by the real robot. At the very moment when the robot's gripper grasps the part the StartProcess button at the user interface screen is pressed. Up to that point all virtual models are on-wait. Note that the robot needs about 14 s to transfer the workpiece and deposits one workpiece every 56 s on the first conveyor.

The plane of reference on which the virtual content is to be displayed needs to be selected before UnityARKitScene starts. Initially, a plane is sought that coincides with the room floor. Planar patches are identified by ARKit™, see Fig. 7a, but the particular application needs a position and orientation reference attached to the patch. For that reason geometric features of the floor of the workshop were used to select the level of machinery augmentation and its orientation, see Fig. 7b. Thus, the point of that plane that is touched by the user's finger on the device screen will serve as the reference point. It only serves once at the beginning of the application and does not serve as a marker.

Characteristics snapshots of the simulation from a distance so as the—typically stationary—user obtains a general impression of the workings of the manufacturing cell are shown in Fig. 8. From a closer distance it is possible to view details of the equipment and of the process being executed, in a—typically-walk—through fashion, as in the characteristic snapshots presented in Fig. 9. It is also straightforward to try the same scenario using different robots, see Fig. 10.

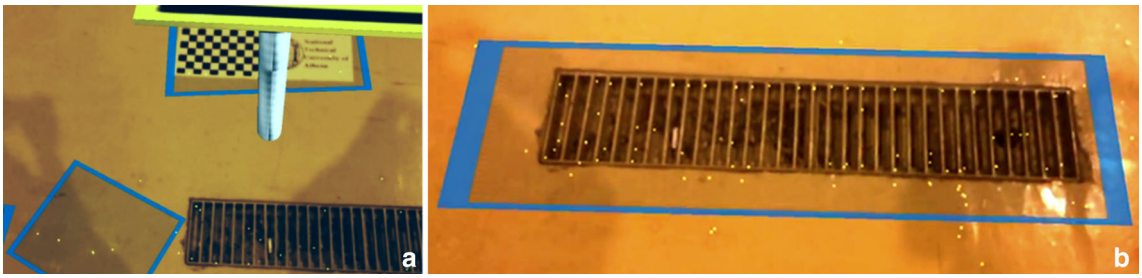

Fig. 7 ARKit™ plane determination on laboratory floor automated

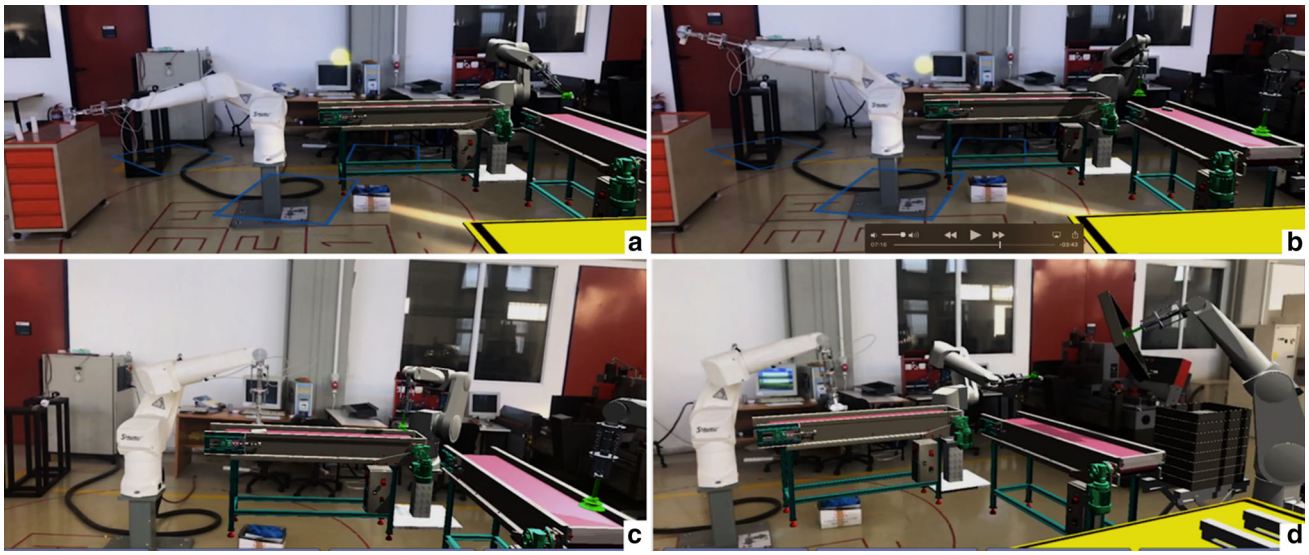

Fig. 8 Overall observation **a** Workpiece-R pickup (Robot1-R), **b** Workpiece-R transfer (Robot1-R) and preparation of Robot2-V, **c** release of workpiece-R on conveyor-V (Robot1-R), **d** transfer of workpiece-V from turning center to Conveyor2-V (Robot2-V) and loading preparation of tray-V (Robot3-V) (suffix legend: *R* real, *V* virtual)

A typical scenario execution is added as a video supplement to the text.

6 Conclusions

The main contribution of developed application to factory layout assessment consists in the following:

(a) The user can assess empirically/intuitively the alternative layouts by navigating through both existing equipment and equipment that is candidate for acquisition.
(b) The model is not static but functional offering the opportunity to simulate the collaboration of real and virtual models, i.e., functions close to the expected ones in reality are witnessed.
(c) Accuracy in layout design can be achieved, since the user/designer can have both an overview of the system and a detailed view of whichever parts of the system are deemed critical.

The interactive character of this approach relates to the capability of the user of the application to freely choose from a library of robots or other machines, place them on the real factory floor as desired and see them functioning in collaboration to other both real and virtual machines according to user-defined kinematic scenarios. Thus, the mix of real and virtual equipment is defined interactively as well as its layout and, to an appreciable extent, its collaborative operation. Interactive nature of the application encompasses intuitive appraisal of the alternative factory configurations that are tried, but quantitative appraisal using time-based indices is not precluded.

A number of future developments are suggested, some currently being pursued by the authors' group:

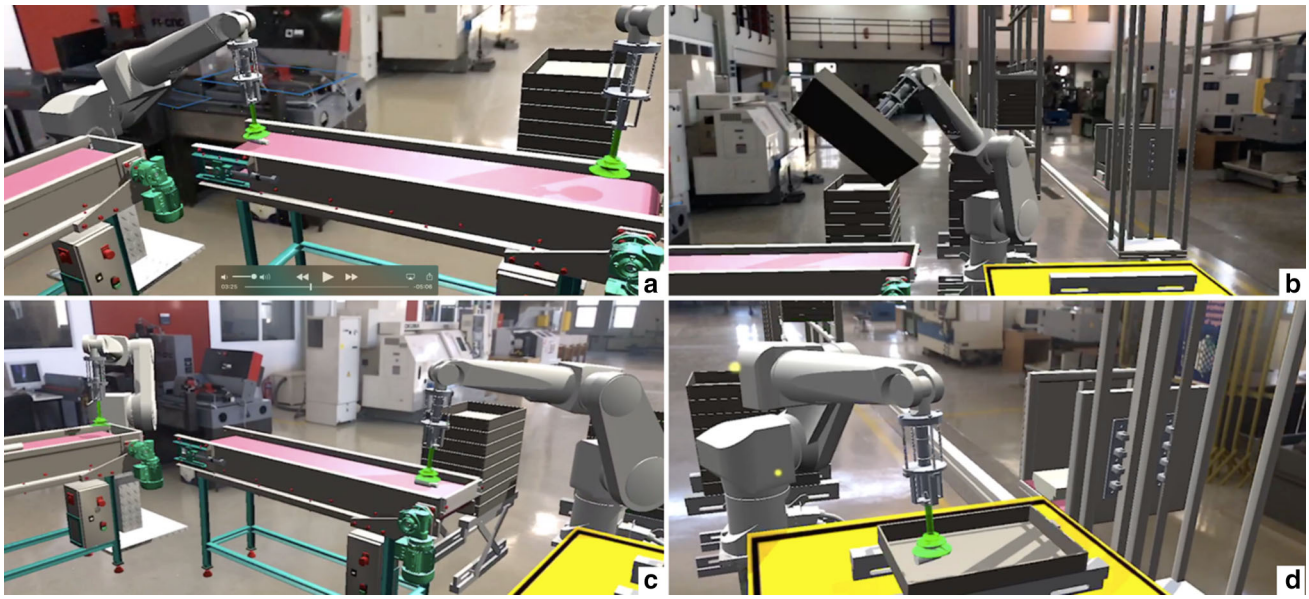

Fig. 9 Detailed observation **a** release and transfer of processed workpiece (Robot2-V), **b** loading of new tray to pickup position (Robot3-V), **c** transfer of processed workpiece to empty tray, **d** release of processed workpiece into empty tray and preparation for loading to ASRS (suffix legend: *R* real, *V* virtual)

Fig. 10 Substitution of different virtual robots

(a) A larger number of AR Scenes, corresponding to different alternative layout, would allow comparison in the same application.
(b) Use of inverse rather than direct kinematics for robot programming, which is favoured when alternative layouts need to be assessed, for which individual joint movement is much more difficult to program compared to end-effector programming.
(c) Extension to Mixed Reality to take occlusion into account. This is possible to achieve in ARKit by programming, i.e., scanning multiple planes, but it would be improved especially through emerging hardware, such as Magic Leap™ and Hololens™ accommodating SLAM technology for space mapping.
(d) Collaboration between real and virtual models can be achieved through sensors (including image recognition from cameras) as a replacement of time-base scheduled scenarios in order to achieve more interactive and flexible simulation mode.

References

1. Jiang, S., Nee, A.Y.C.: A novel facility layout planning and optimization methodology. CIRP Ann. Manuf. Technol. **62**, 483–486 (2013). https://doi.org/10.1016/j.cirp.2013.03.133
2. Paelke, V.: Augmented reality in the smart factory: supporting workers in an industry 4.0. environment. In: Proceedings of the 2014 IEEE Emerging Technology and Factory Automation (ETFA), pp. 1–4. IEEE (2014)
3. Billinghurst, M., Clark, A., Lee, G.: A survey of augmented reality. Found. Trends Hum. Comput. Interact. **8**, 73–272 (2015). https://doi.org/10.1561/1100000049
4. Jeong, B., Yoon, J.: Competitive intelligence analysis of augmented reality technology using patent information. Sustainability **9**, 497 (2017). https://doi.org/10.3390/su9040497
5. Chryssolouris, G., Mavrikios, D., Papakostas, N., Mourtzis, D., Michalos, G., Georgoulias, K.: Digital manufacturing: history, perspectives, and outlook. Proc. Inst. Mech. Eng. Part. B J. Eng. Manuf. **223**, 451–462 (2009). https://doi.org/10.1243/09544054JEM1241
6. Shariatzadeh, N., Sivard, G., Chen, D.: Software evaluation criteria for rapid factory layout planning, design and simulation. Proc. CIRP **3**, 299–304 (2012). https://doi.org/10.1016/j.procir.2012.07.052
7. Pentenrieder, K., Bade, C., Doil, F., Meier, P.: Augmented Reality-based factory planning—an application tailored to industrial needs. In: 6th IEEE and ACM International Symposium on Mixed and Augmented Reality, Nara, Japan, pp. 31–42
8. Lee, J., Han, S., Yang, J.: Construction of a computer-simulated mixed reality environment for virtual factory layout planning. Comput. Ind. **62**, 86–98 (2011). https://doi.org/10.1016/j.compind.2010.07.001
9. Gupta, O.K., Jarvis, R.A.: Using a virtual world to design a simulation platform for vision and robotic systems. In: Advances in Visual Computing, pp. 233–242. Springer (2009)
10. Reinhart, G., Munzert, U., Vogl, W.: A programming system for robot-based remote-laser-welding with conventional optics. CIRP Ann. Manuf. Technol. **57**, 37–40 (2008). https://doi.org/10.1016/j.cirp.2008.03.120
11. Akan, B., Ameri, A., Curuklu, B., Asplund, L.: Intuitive industrial robot programming through incremental multimodal language and augmented reality. In: IEEE International Conference on Robotics and Automation (ICRA), pp. 3934–3939 (2011)
12. Chong, J., Ong, S., Nee, A., Youcef-Youmi, K.: Robot programming using augmented reality: an interactive method for planning collision-free paths. Robot. Comput. Integr. Manuf. **25**, 689–701 (2009). https://doi.org/10.1016/j.rcim.2008.05.002
13. Fang, H., Ong, S., Nee, A.: Interactive robot trajectory planning and simulation using augmented reality. Robot. Comput. Integr. Manuf. **28**, 227–237 (2012). https://doi.org/10.1016/j.rcim.2011.09.003
14. Vosniakos, G.C., Levedianos, E., Gogouvitis, X.V.: Streamlining virtual manufacturing cell modelling by behaviour modules. Int. J. Manuf. Res. **10**, 17–43 (2015). https://doi.org/10.1504/IJMR.2015.067616
15. Menck, N., Yang, X., Weidig, C., Winkes, P., Lauer, C., Hagen, H., Hamann, B., Aurich, J.C.: Collaborative factory planning in virtual reality. Proc. CIRP **3**, 317–322 (2012). https://doi.org/10.1016/j.procir.2012.07.055
16. Luo, Y.-B.: A virtual layout system integrated with polar coordinates-based genetic algorithm. Int. J. Comput. Appl. Technol. **35**, 122–127 (2009). https://doi.org/10.1504/IJCAT.2009.026589
17. Nee, A.Y.C., Ong, S.K., Chryssolouris, G., Mourtzis, D.: Augmented reality applications in design and manufacturing. CIRP Ann. Manuf. Technol. **61**, 657–679 (2012). https://doi.org/10.1016/j.cirp.2012.05.010
18. Hibino, H., Inukai, T., Fukuda, Y.: Efficient manufacturing system implementation based on combination between real and virtual factory. Int. J. Prod. Res. **44**, 3897–3915 (2006). https://doi.org/10.1080/00207540600632224
19. Bal, M., Hashemipour, M.: Virtual factory approach for implementation of holonic control in industrial applications: a case study in die-casting industry. Robot. Comput. Integr. Manuf. **25**, 570–581 (2009). https://doi.org/10.1016/j.rcim.2008.03.020
20. Abdul Kadir, A., Xu, X., Haemmerle, E.: Virtual machine tools and virtual machining—a technological review. Robot. Comput. Integr. Manuf. **27**, 494–508 (2011). https://doi.org/10.1016/j.rcim.2010.10.003
21. Pedrazzoli, P., Sacco, M., Jonsson, A., Boer, C.R.: Virtual factory framework: key enabler for future manufacturing. In: Cunha, P., Maropoulos, P. (eds), Digital Enterprise Technology: Perspectives and Future Challenges, pp 83–90. Springer (2007)
22. Cecil, J., Kanchanapiboon, A.: Virtual engineering approaches in product and process design. Int. J. Adv. Manuf. Technol. **31**, 846–856 (2007). https://doi.org/10.1007/s00170-005-0267-7
23. Ostermayer, D., Aurich, J., Wagenknecht, C.: Improvement of manufacturing processes with virtual reality based CIP-workshops. Int. J. Prod. Res. **47**, 5297–5309 (2009). https://doi.org/10.1080/00207540701816569
24. Monostori, L.: Cyber-physical production systems: roots, expectations and R&D challenges. Proc. CIRP **17**, 9–13 (2014). https://doi.org/10.1016/j.procir.2014.03.115
25. Unity_Technologies (2017) Unity User Manual (5.5). https://docs.unity3d.com/Manual/index.html. Accessed 11 Feb 2017
26. Dilger, D.E.: Inside Apple's ARKit and Visual Inertial Odometry, new in iOS 11. https://appleinsider.com/articles/17/10/12/inside-apples-arkit-and-visual-inertial-odometry-new-in-ios-11. Accessed 11 Apr 2018
27. Gogouvitis, X.V., Vosniakos, G.-C.: Construction of a virtual reality environment for robotic manufacturing cells. Int. J. Comput. Appl. Technol. **51**, 173–184 (2015). https://doi.org/10.1504/IJCAT.2015.069331
28. Michas, S., Matsas, E., Vosniakos, G.-C.: Interactive programming of industrial robots for edge tracing using a virtual reality gaming environment. Int. J. Mech. Manuf. Syst. **10**, 237–259 (2017). https://doi.org/10.1504/IJMMS.2017.087548