CEIG - Spanish Computer Graphics Conference (2017)
F. J. Melero and N. Pelechano (Editors)

Extending Industrial Digital Twins with Optical Object Tracking

A. Tammaro[1], A. Segura[1], A. Moreno[1] and J. R. Sánchez[1]

[1]Vicomtech-IK4, Mikeletegi Pasealekua, 57,Donostia - San Sebastián. Spain

Abstract

In the last year, the concept of Industry 4.0 and smart factories has increasingly gained more importance. One of the central aspects of this innovation is the coupling of physical systems with a corresponding virtual representation, known as the Digital Twin. This technology enables new powerful applications, such as real-time production optimization or advanced cloud services. To ensure the real-virtual equivalence it is necessary to implement multimodal data acquisition frameworks for each production system using their sensing capabilities, as well as appropriate communication and control architectures. In this paper we extend the concept of the digital twin of a production system adding a virtual representation of its operational environment. In this way the paper describes a proof of concept using an industrial robot, where the objects inside its working volume are captured by an optical tracking system. Detected objects are added to the digital twin model of the cell along with the robot, having in this way a synchronized virtual representation of the complete system that is updated in real time. The paper describes this tracking system as well as the integration of the digital twin in a Web3D based virtual environment that can be accessed from any compatible devices such as PCs, tablets and smartphones.

CCS Concepts
•***Computing methodologies*** → ***Virtual reality;*** •***Applied computing*** → *Industry and manufacturing;*

1. Introduction

The new Industry 4.0 landscape introduces novel concepts that impact in different industrial areas. The addressed technologies behind the Industry 4.0 concept offer new market opportunities regarding the products and services around the factories of the future. These technologies comprise several disciplines like knowledge management, big data analysis, cyber-security, robotics, computer vision, human-computer interaction or simulation.

A subset of those technologies are enclosed under the Visual Computing concept: a key enabling technology for the realization of the Industry 4.0 vision [PTB*15] that mixes different and complementary technologies, such as Computer Graphics, Computer Vision, Human Machine Interaction and Multimodal and Multimedia Technologies.

Following the Industry 4.0 strategy, the introduction of the Digital Twin concept allowed to construct virtual entities connected with their real counterparts. This relationship goes beyond the mere 3D appearance, as the ultimate goal of a digital twin would be to mimick the behavior of a given physical system and its relationships with other components, the environment and the operators.

In this paper the concept and application of Digital twins are extended including a representation of the operational environment of a production system. The work is presented as a proof of concept using a simple setup in an academical environment. Through the use of simple and affordable image-based sensors, like *RGB-D* cameras or plain image cameras, we want to extend the virtualization to the environment that surrounds the machine and to every object that reside in the observed space. Our approach contemplates the use of computer vision algorithms and three-dimensional geometry on real pieces, images and CAD models, to recognize known shapes or detect the volume occupied by unknown objects. Our extension from a Digital Twin to a *Digital Environment* could help to deploy innovative industrial applications, allowing reasoning algorithms to operate with a far wider perspective on what is actually happening.

This paper is organized as follows, the next section describes our algorithms and the results that we achieved. In Section 3 we discuss our results and the improvements that we are currently developing, as well as the most problematic points of the digital environment representation, while in Section 4 the conclusions of this paper are made.

2. Method

Our Environment Digital Twin system is composed of two stages:

1. A sensing apparatus composed of a camera, and the robot controller that records information about the subjects in the scene. The sensors are controlled with a Qt based desktop application.

DOI: 10.2312/ceig.20171204

2. A web-based, three-dimensional visual interface in which all the information are displayed

Data comes both from external sensors, a *Point Grey Flea3* GigE camera, or internal, like the joint encoders of a robot manipulator. We used a *Universal Robots UR10* robotic arm as an actuator and its controller for acquiring the end-effector position and orientation in real time. To attach different tools to the robot end-effector, such as feature boards or tips, we used a custom-designed steel plate and other 3D-printed plastic supports. Alternatively, a two or three-finger gripper can be attached to the arm for object manipulation. The challenge is to link the different coordinate systems so that objects in camera image space, as well as objects attached to the robot end effector, can be located in world space. This requires an initial calibration step that is described in Section 2.1.

For the virtual 3D representation of the Digital Twin we used Web3D technology. This enables ubiquitous access from any desktop or mobile device with a web browser supporting WebGL [Khr]. The 3D view is contained in a web page that contains user interface elements to control the robot. The 3D view implementation has been implemented using the X3DOM [BJK*10] library.

Communication between the web interface, the robot and the optical tracker is based on HTTP REST web services. A server program connects to the robot via its TCP socket protocol to read the robot state and send commands to it, and exposes an HTTP service. This service can be used by the tracking system to post the latest shape detected as well as by the web visualization to retrieve the latest robot pose and detected shape data.

2.1. Camera calibration

The goal of this stage is to bring every source into a common world reference frame expressed as W. To achieve this, a checkerboard is attached to the robot end-effector as depicted in Fig. 1 and several images are captured with the camera (Fig. 3), usually about 15 – 20. For each image, the corresponding robot pose, as provided by the robot controller, is also stored. In the following explanation, several reference frames are mentioned. As shown in Fig. 2, they are the world reference frame W, the robot end-effector frame (or robot pose), the chessboard coordinate system, and the camera view coordinate system.

We use the *OpenCV* library [Its15] to handle the image stream of the camera, operate on matrices and for some basic computer vision algorithms. In our implementation we work with a two-dimensional projection of the subject onto the working table because of the nature of the single camera used. The working table is the focal point of the Digital Twin. We represent it with a 3D plane called Π, which contains the origin of W, its main axes $\mathbf{x}$ (W_x) and $\mathbf{y}$ (W_y) and the piece. Following a right-hand representation the $\mathbf{z}$ axis of W will be directed upwards and therefore the z component of the end effector is the distance between the tip of the robot and the table. We developed an algorithm that localizes any camera capable of recording images in the environment, and it's based only on the pictures that the camera produces. It is important to notice that this calibration procedure is needed only once, when the camera is placed, or every time it is moved. In an industrial environment this is very convenient because the cameras are installed and calibrated only at the

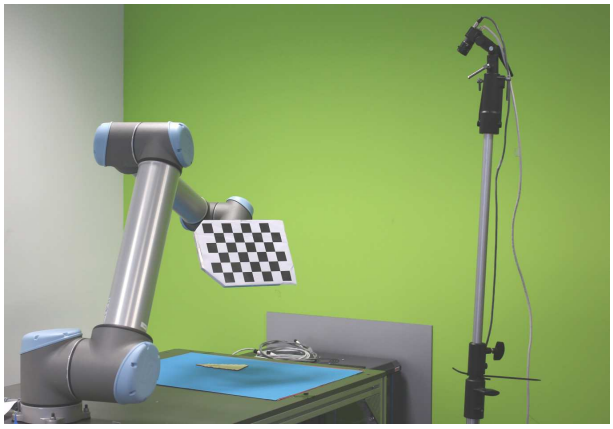

Figure 1: *To compute the extrinsic parameters of the camera in the world reference frame, a checkerboard is attached to the robot to create a link between the pose of the end effector and the checkerboard points detected in the camera image.*

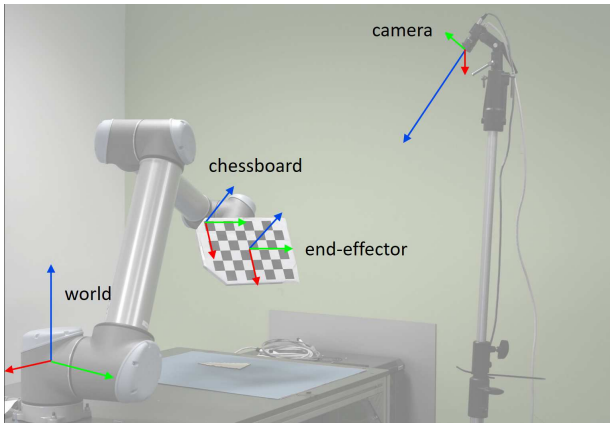

Figure 2: *The various reference frames depicted on the setup (the end-effector is under the chessboard plate)*

first use, and while their position is fixed, the system does not need further calibrations.

The following step is to use the POSIT algorithm [DD95] from *OpenCV* to have a rough estimation of the camera pose for every frame with respect to the identified checkerboard. We represent this pose as a transformation matrix between the chessboard and the camera itself called $T_{C,S}$. The final pose of the camera will be a composition of several transformations, the first is the one from the world reference frame to the end effector, which we will call T_{EE}. The second is the transformation between the end effector and the chessboard called $T_{EE,C}$. The transformation that transforms points from W to the camera coordinate system is:

$$T_S = T_{C,S} T_{EE,C} T_{EE} \tag{1}$$

To estimate the transformation $T_{EE,C}$ and to correct introduced by

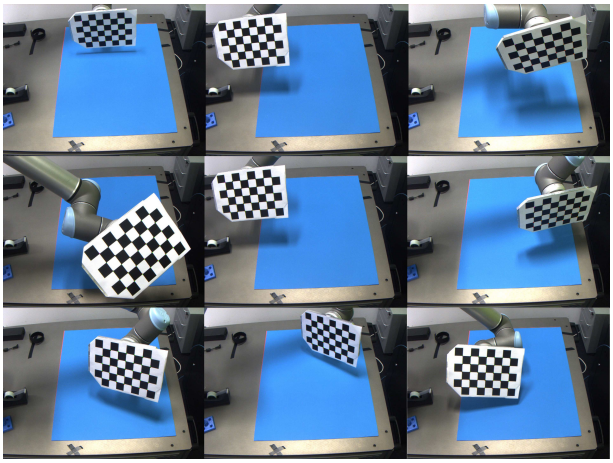

Figure 3: *Several shots of the calibration procedure taken from the camera. A checkerboard of known dimensions is attached to the robot end-effector, then it's positioned at different positions and orientations and an image is recorded with the camera.*

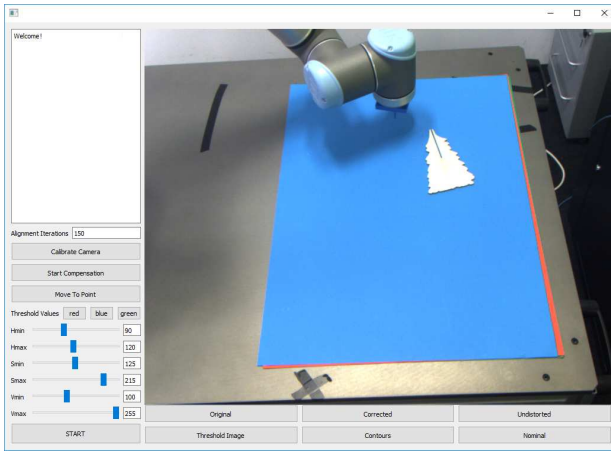

Figure 4: *The desktop application to extract information from the sensors and move the robot in specific points chosen directly on the image. The yellow object on the blue sheet is the piece that we are interested in.*

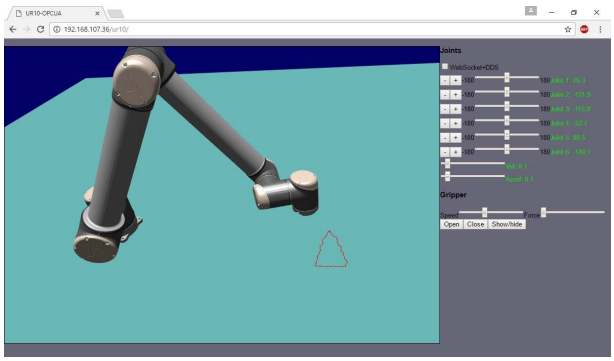

Figure 5: *The Web application displaying the 3D model of the robot, the contour of the piece (red), the table (green) and several command buttons to control the actuator.*

the POSIT algorithm we utilize a two-step *Levenberg-Marquardt* based minimization, that firstly compute $T_{EE,C}$ and then compute the final pose of the sensor while minimizing the detected points reprojection error in the images [HZ04]. The last step of the calibration is to close the loop between the camera pose and the world reference frame and to correct the error accumulated during the previous computations, we move the robot in the four corners of the image projected on the Π plane. We measure the deviation between the picked position on the image and the real end-effector position. Then we perform a bilinear interpolation of the correction vector on the Π plane so every time that an image point is projected on the plane, the correction is applied.

A blue sheet of paper is utilized to create more contrast between the piece and the background, this allows us to use a thresholding algorithm to efficiently extract the contour of the subject in the image. In a real application more complex background subtraction algorithms may be necessary.

Finally, we backproject the contour points of the object and intersect the generated rays with the Π plane, obtaining the three-dimensional points in the world reference frame [HZ04].

2.2. The Digital Environment Representation

The control of sensors and actuators along with the data visualization is made with two applications, the first is a Qt based desktop application, Fig. 4, that abstracts the specific sensor drivers and provides all the computer vision functionalities, the camera calibration, lens and perspective distortion correction, and robot movements. As a matter of fact through this interface is possible to pick points on the image, project them onto the Π plane and move the robot end effector in that particular plane position. The second application is used for the three-dimensional representation in a web page, Fig. 5. It is implemented as an X3D [Web] scene embedded in the HTML page DOM thanks to the X3DOM JavaScript library.

The robot is defined as a set of meshes stored as separate X3D files added to a set of nested Transform nodes representing the six robot joints. X3D is a declarative format for describing 3D content usually written as XML documents. X3DOM allows access to all elements of the scene by accessing its XML elements. Thus, the visual scene can be updated to match the state of the physical setup just by modifying its nodes and attributes.

As explained before, communications between this page and the physical setup is mediated by an HTTP REST service. The web interface uses this service to query the current robot pose. Additionally, the service allows the optical tracker to post information about the tracked shape which is stored in the server's memory. All data is exchanged in the form of JSON-formatted messages.

This communication architecture is based on continuous polling: the web page is constantly sending requests for new robot pose data and new shape contour data, and the tracker is constantly sending shape contour data in the form of post requests.

The page queries the server at a given frequency for new robot data and when a new pose is received (the values of the 6 angles of the joints) its angles are applied to the rotations of the 6 corresponding transform nodes.

The tracked part shape visualization is defined as a closed polyline (a LineSet node in X3D). Similarly to the robot pose, the page queries the server for newly detected shapes. These shapes are defined as a sequence of x, y, z point coordinates which the application uses to update the vertices of the polyline. In our experiments, robot position is queried at 10 Hz and shape contour data at 6 Hz. These are conservative frequencies in order to limit network traffic. Nevertheless, perceived latency is accaptably low for the slow movements involved. A more efficient system could employ the WebSocket protocol for direct transmission of new data to the visual interface with lower latency.

Additionally, the web page has several user interface elements that allow a user to manually control the robot and its gripper. Currently each joint angle can be increased or decreased in fixed steps, and motion speed and acceleration can also be controlled. User actions to these user interface elements trigger corresponding requests to the server that in turn translates them to the robot control protocol.

3. Discussion and future developments

Being this a work in progress, some problems still need to be addressed. Our main concern is to reduce at minimum the occlusions that the robot generates on the observed subject. The solution that we are currently developing is to increase the number of sensors in place in order to acquire data from different angles. The desired configuration is an array of three RGB cameras one for every side of the table at different heights and a stereo 3D camera positioned perpendicularly on top of the table. We are working on adding an *Ensenso n35* stereo camera for extracting 3D points of the subject and to overcome the current planar part limitation. With this arrangement the potential subject occlusions are minimized, and the scanned area is maximized. All the cameras will follow the same calibration process that is explained in Section 2.1, with an additional bundle adjustment [TMHF00] to refine the 3D coordinates of the subject and the relative positions between the cameras.

Another important question that affects the performance of the system is the segmentation of the subjects in the scene and the detection of known production pieces. We aim to improve the segmentation algorithm in order to remove the dependency from a colored sheet. In addition, due to the static nature of the environment, it is possible to use a background subtraction algorithm to highlight all the non-static objects. This method could also be utilized for extracting data from the production line, but as well information about eventual persons in the scene.

A relevant aspect of the digital environment is the differentiation of the data sources. One of the main concepts in the Industry 4.0 is the enormous amount of data that we are now able to collect and process. Therefore one of the major goals of this paper is to integrate as much data as possible, to finally have the most complete and deep representation of the environment. We already developed a library for 3D geometric algorithms, mesh and point cloud comparisons and object detection called *Geomlib* [SSB*15]. With the tools implemented in our library, the data coming from sensors and some prior knowledge about the pieces, e.g. CAD models, we are working right now on matching these sources of data, namely the point cloud coming from the Ensenso RGB-D camera and CAD models of the pieces in order to integrate the latter in the web visualizer.

Finally, the communication between the different modules (robot control, optical tracking and web visualization) can be improved using a more efficient protocol. The current implementation uses the HTTP protocol which require the client web page to continuously request new data. We intend to investigate a WebSocket-based protocol which can allow new data to be pushed to the client, reducing latency and network traffic.

4. Conclusions

In this paper we presented an extension of the environment to the common digital twin representation. We described a method to compute the extrinsic parameters of every sensor capable of capturing images regardless of its nature. We developed two applications, the first is designed to extract information from the pictures, e.g. the subject position and contour, and command the robot to move in specific image points. the second application is web-based and therefore accessible from every device. It shows the 3D virtual environment with the table, the actuator and the subject. Through this interface is also possible to move the robot and send specific commands to its controller.

References

[BJK*10] BEHR J., JUNG Y., KEIL J., DREVENSEK T., ZOELLNER M., ESCHLER P., FELLNER D.: A Scalable Architecture for the HTML5/X3D Integration Model X3DOM. In *Proceedings of the 15th International Conference on Web 3D Technology* (New York, NY, USA, 2010), Web3D '10, ACM, pp. 185–194. 2

[DD95] DEMENTHON D. F., DAVIS L. S.: Model-based object pose in 25 lines of code. *International Journal of Computer Vision 15* (1995), 123–141. 2

[HZ04] HARTLEY R. I., ZISSERMAN A.: *Multiple View Geometry in Computer Vision*, second ed. Cambridge University Press, ISBN: 0521540518, 2004. 3

[Its15] ITSEEZ: OpenCV, Open Source Computer Vision Library. http://opencv.org/, 2015. 2

[Khr] KHRONOS GROUP: WebGL, OpenGL ES for the Web. https://github.com/KhronosGroup/WebGL. 2

[PTB*15] POSADA J., TORO C., BARANDIARAN I., OYARZUN D., STRICKER D., DE AMICIS R., PINTO E., EISERT P., DOLLNER J., VALLARINO I.: Visual Computing as a Key Enabling Technology for Industrie 4.0 and Industrial Internet. *IEEE Comput. Graph. 35*, 2 (Mar 2015), 26–40. 1

[SSB*15] SÁNCHEZ J. R., SEGURA A., BARANDIARAN I., MUÑOZ J., LARREA J. A.: Dimensional Inspection of Manufactured Parts with Web-Based 3D Visualization. In *Virtual Concept International Workshop on Industry 4.0* (2015). 4

[TMHF00] TRIGGS B., MCLAUCHLAN P. F., HARTLEY R. I., FITZGIBBON A. W.: Bundle adjustment - a modern synthesis. In *Proceedings of the International Workshop on Vision Algorithms: Theory and Practice* (London, UK, UK, 2000), ICCV '99, Springer-Verlag, pp. 298–372. 4

[Web] WEB3D CONSORTIUM: ISO/IEC 19775-1: Extensible 3D (X3D). http://www.web3d.org/standards/number/19775-1. 3