AKADÉMIAI KIADÓ

UNIVERSITY of DEBRECEN

International Review of Applied Sciences and Engineering

DOI:
10.1556/1848.2023.00660

ORIGINAL RESEARCH PAPER

Cyber-physical recreation of six DOF industrial robot arm

Timotei István Erdei[1*], Dávid Péter Nusser[2] and Géza Husi[1]

[1] Department of Air- & Road Vehicles, University of Debrecen, Hungary

[2] Faculty of Engineering, University of Debrecen, Debrecen, Hungary

Received: April 13, 2023 • Accepted: September 25, 2023

ABSTRACT

The Cyber-Physical and Vehicle Manufacturing Laboratory, a model Industry 4.0 laboratory, is applying new innovative solutions to improve the quality of education. As part of this, a digital twin of the lab was designed and built, where users can practice. In the virtual space, it is possible to apply the known robot motion types, and the tool centre and wrist speed have been measured virtually. Robot control tasks can be performed "offline" using parameters. This information can then be transferred to the actual physical robot unit. The stable diffusion 1.5 deep learning model generates 2D geometric shapes for trajectory, allowing users to perform unique tasks during education. The Google Colab cloud-based service was used to teach our rendered-type dataset. For the 3D simulation frame, we used V-REP, which was developed on a desktop PC equipped with an Intel Core i5 7600K processor, Nvidia GTX1070 VGA with 8 GB of DDR5 VRAM, and 64 GB of DDR4 memory modules. The following material describes an existing industrial six-axis robot arm and its implementation, which can be controlled and programmed while performing virtual measurements after integrating into a Cyber-Physical system and using deep learning techniques.

KEYWORDS

digital twin, virtual robot unit, Industry 4.0, 3D modelling

1. INTRODUCTION

Our university has considered innovative technologies in order to improve education and overcome obstacles. With the outbreak of COVID-19, routine tasks that were previously thought to be routine were evaluated at a number of points. After all, the strict rules of epidemiology made it impossible to program a lot of industrial machines and test new systems.

In recent years, the foundations for a modern laboratory in the spirit of Industry 4.0 have been laid. As an isolated environment, the Cyber-Physical System (CPS) and vehicle manufacturing laboratories enable complex material handling tasks as well as image analysis tasks. The lab can also be used for other vehicle-related tasks, as there are automotive robots in the lab [1]. Previously, measurements on electric vehicles have also been carried out using the laboratory's equipment [2]. These have been used as a basis for further simulations [3] and for the construction of vehicles [4]. The lab will also play an important role in the development of autonomous vehicles [5].

After 2015, the industry adopted the relatively new digital twin technique, which can be thought of as an industry 4.0 optimization [6]. A user can partially interact with and control virtual models [7]. It is important to note that virtual objects do not necessarily need to be able to communicate with real-patterned physical objects. Replicated copies of production lines contribute to developing manufacturing concepts that can efficiently use available resources [8]. But as numerous studies have shown, intelligent manufacturing requires significant real-time data processing and storage [9]. This is because digital twins, as copies of

*Corresponding author.
E-mail: timoteierdei@eng.unideb.com

AKJournals

existing systems, contain all elements of a given production line, including textures and other behavioral characteristics [10]. Technology can play a crucial role in the testing phase before a process is implemented physically [11]. Maintenance is another area where it can be useful, extending the lifespan of a production line or product [12]. And when specific data is used, applying deep learning-based techniques can facilitate design decisions [13].

In our case, our laboratory uses a number of custom, in-house-designed elements that are part of internal developments and not part of any other simulation environment. In many cases, we manufacture our own 3D-CAD (computer-aided design) parts in the lab using a 3D printer. The robotic cell that students use in class is also an in-house development [14]. However, in order for it to be used "offline" by the current users, it needs to be implemented in a virtual environment. The digital twin to be created is not just a layout of a system; it is possible to interact with the individual elements and control the virtual robot model. Furthermore, it has educational implications, as each user will be able to solve a unique task based on a 2D geometric trajectory map generated from our own rendered data.

The integration of the laboratory shown in Fig. 1 into a virtual form is a complex task. This is due to the fact that simulations are classified into different groups, each with its own set of characteristics.

One of the more general definitions of simulation states that it is suitable for realistic imitation of a given process, considering changes in state variables, and is suitable for evaluating the process system.

The uses just mentioned are also important in our case. Because of the lab's continuous development, a number of unique, self-designed elements were added, all of which are critical to the system as a whole. As a result, their incorporation into the environment to be designed is also critical (see Fig. 1).

2. SIX DOF INDUSTRIAL ARC WELDING ROBOT ARM

Most of the information here applies to the KUKA KR5, a six-degree-of-freedom robot arm commonly found in the automotive industry. The main application for robot design is arc welding operations.

The robot's workspace is 8.4 m^3, allowing it to perform complex tasks. It also has a repetition accuracy of 0.04 mm [15].

The KUKA KR5 robot arm has six axes and a KRC2 (Kuka Robot Controller-2) type controller. The maximum payload is only 5 kg [16].

Several customized parts have been produced in the lab, despite the parts' lengthy design and manufacturing processes, as previously stated. Additional parts were also developed for the KUKA KR5 robot, which is required for logistical procedures.

Because the robot unit lacked a gripper, an adaptor was designed and manufactured in-house using a 3D printer and FDM (Fused Deposition Modelling) technology. Because the adapter will be used in an industrial setting, ABS+ (acrylonitrile-butadiene-styrene) filament, which is also used in the automotive industry, was chosen.

The following were the print statistics when the adapter was printed:

- Estimated time: 02:01:08
- Number of layers: 127
- Number of lines: 64,119
- Required thread length: 4,283

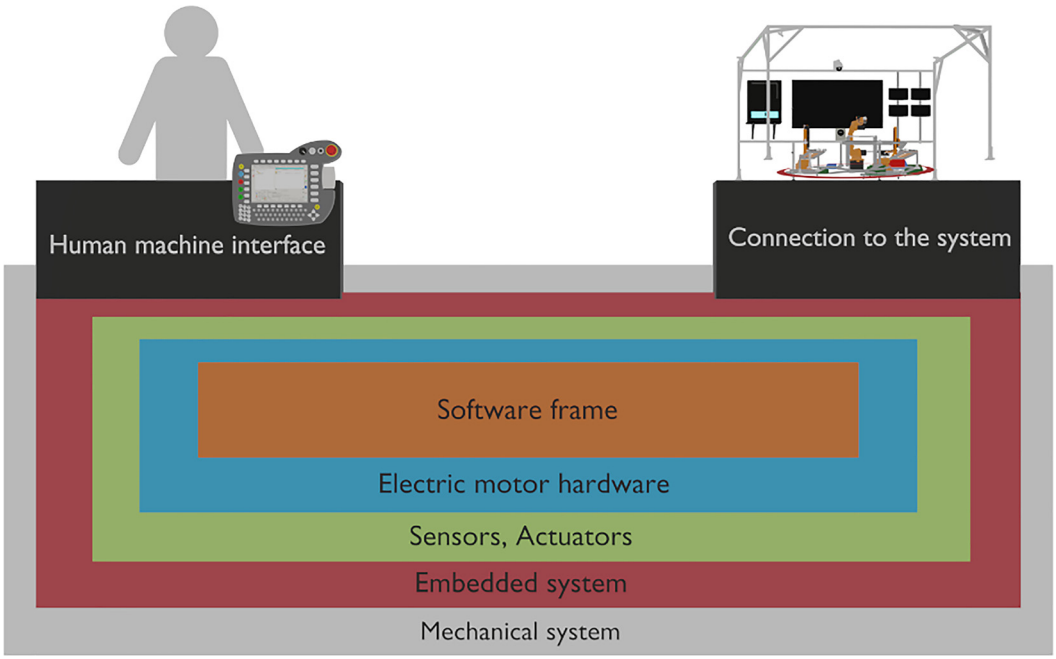


Fig. 1. CPS's architecture model 'Own source'

Using the manufactured adapter, the GmbH (Gesellschaft mit beschränkter Haftung - Limited liability company) electro-pneumatic gripper can be attached to the KUKA KR5 robot unit, expanding the range of operations that can be performed (shown in Fig. 2) [17].

One of the goals of the envisioned virtual robot cell is to allow for immediate pre-testing of the given element following the 3D CAD modelling process. However, the KUKA KR5 was not designed for the application, it could be tested in a virtual environment, and such virtual measurements could provide information on whether the component requires 3D printing.

In the future, testing new features or additional components in advance will suffice to determine whether or not there are any issues. This eliminates the need to implement new features or components physically.

3. 3D CAD MODELLING & IMPORTATION

It is essential to emphasize the fact that a specialized method has been designed to accommodate the file formats that will be utilized in the simulations. Each of the numerous 3D CAD modelling programs is equipped with its own distinct file format(s). Among the 3D CAD file formats used are .STL (Standard Triangle Language), .OBJ (Wavefront Object), .FBX (Filmbox), .ASM (Assembly Model List), .3DS (3D Studio Mesh Format), and .SLDPRT (SL Works Part).

Interoperability between 3D CAD design programs is difficult due to specific file formats. There are file extensions that are not suitable for 3D printing, but there are also formats that, once exported, can contain pre-saved animations or textures. In modelling, there are cases where only the 3D model creation itself is done in a particular 3D CAD software, and texturing or grouping of individual 3D elements is done in another program. This requires knowledge of file formats and which 3D CAD software can handle them after export (see Fig. 3). If the 3D model to be designed does not have a specific function (static model), the .STL (Standard Triangle Language) format is used. If you want to add texture to the 3D model, you can use the .OBJ (Wavefront Object) format.

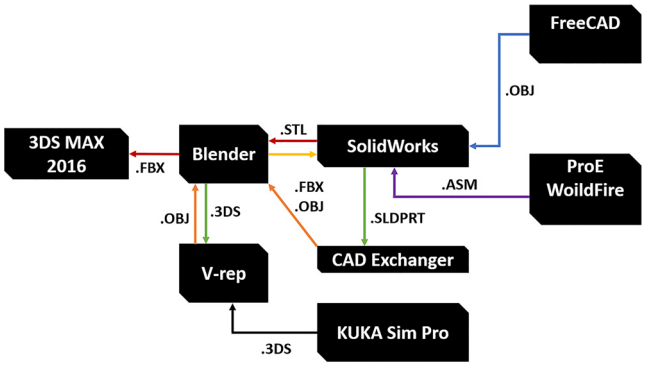


Fig. 3. CAD methodology for using 3D file extensions 'Own source'

In turn, these can only be imported into a simulation environment to a limited extent due to its specific requirements. Additionally, the models needed to be optimized, which meant that polygon reduction needed to be carried out.

The technique was used earlier at the University of Debrecen, Faculty of Engineering, Industry Days event. Within this framework, in 2021, a separate application was developed under the name "Cyber-Physical based Industry Days," where the entire building model was modelled in 3D and users could interact with the individual elements [18]. The application for Industry Days and the current project are very different in many ways. The former served as a tour guide where large spaces had to be created and imported. Also, video materials had to be assigned to different model units after conversion. Regarding interactions, camera movement and switching between Voice Over Internet Protocol (VoIP) links were the first order, and programming and control were not part of the interactions. Ultimately, Industry Days was a presentation event (see Fig. 4).

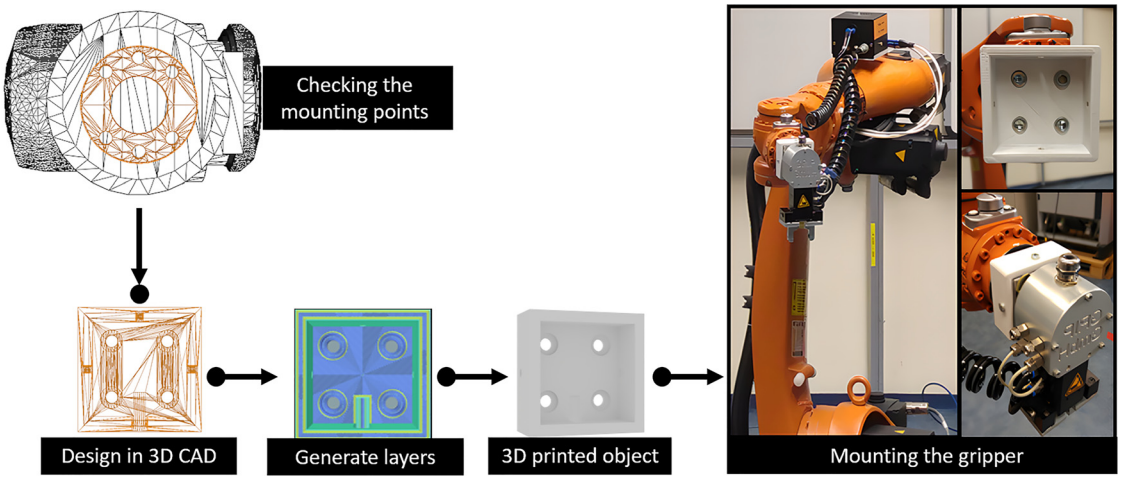


Fig. 2. The 3D-printed KUKA KR5 adapter 'Own source'

Fig. 4. Industry Days application – 2021 'Own source'

In our project now, the main 3D CAD modeler program is Blender, which is an open-source CAD modeler that can also be used to make mechanical simulations [19]. In our case, the KUKA KR5 model was created.

Following that, during the definition of the axes, the specified elements of the robot, from A1 to A6, were refined and named (see Fig. 5). Furthermore, the names of the Gmbh gripper and the "Tool" device used to measure the tool are different.

Blender requires grouping because the models will be able to move in a chain, but the axes will be free to move.

The robot cell system built around the robot unit was inspired by the KUKA KR5 robot arm.

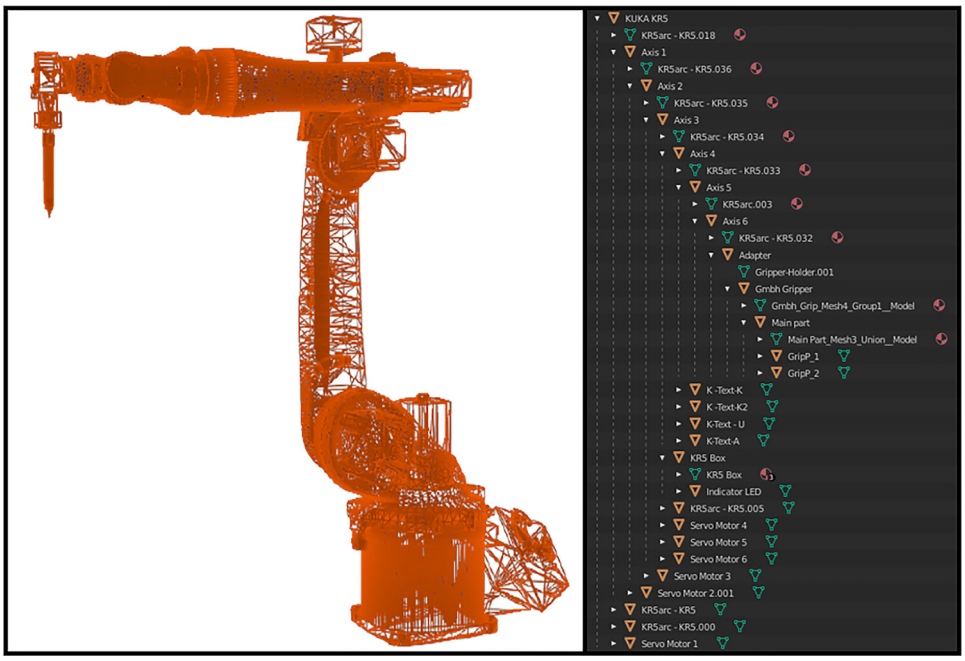


Fig. 5. Model of KR5 in grouped 3D CAD format 'Own source'

Given what has come before, a simulation framework was required to serve as the foundation for our proposed CPS.

The KUKA KR5 main robot unit and the robot cell have also been designed. However, all CAD models must be exported, which means the simulation environment must support .DAE, .OBJ, and .STL file extensions [20]. Furthermore, the defined requirements must be met, which include the ability to provide input and output data, create scripts for unique programs, and interact with the virtual model to perform various types of movements. Furthermore, because the CPS will be used as a pre-tester, virtual measurements must be taken in order to later compare them to real-world values.

The V-REP (Virtual Robot Experimentation Platform) satisfies the above-mentioned requirement for developing the CPS. Toshiba R&D initially developed the simulation framework for commercial and academic use. The platform is self-contained and certifiable. Its distributed control architecture allows it to write Lua, C/C++, and Python scripts. Physical simulations are also performed using the Bullet, ODE (Open Dynamics Engine), and Vortex engines [21]. The imported model is represented in Fig. 6.

4. DEEP LEARNING BASED STABLE DIFFUSION TEXT TO IMAGE GENERATOR

Stable diffusion is a new generative model-based image generator, the first version of which was published in 2022 by Google's research team [22]. Stable diffusion itself is open-source and designed to generate arbitrary images using so-called promt. In order for the image generation process to be successful, we had to train the checkpoint file (.ckpt) with a large number of images found on the Internet using Deep Neural Networks (DNNs) [23]. The training process is extremely resource-intensive, which is why the resolution of the generated images is only 512 × 512. Post-upscaling is, of course, possible, so high resolution is available. To use stable diffusion, we first download the v1.5.ckpt (checkpoint) model. Upload it to the Drive cloud storage account, which we then mount from there to the Google Colab cloud service. Then we install the necessary repositories, including AUTOMATIC1111 [24]. Then we start Webui via the public URL (Uniform Resource Locator), which gives access to additional configuration and promt management. Through the user interface, you can make a number of settings, such as the size of the image and how many images to generate. In addition, the Classifier-Free Guidance Scale (CFG) parameter can be set here. This value determines how much of the promt we specify should be considered during generations.

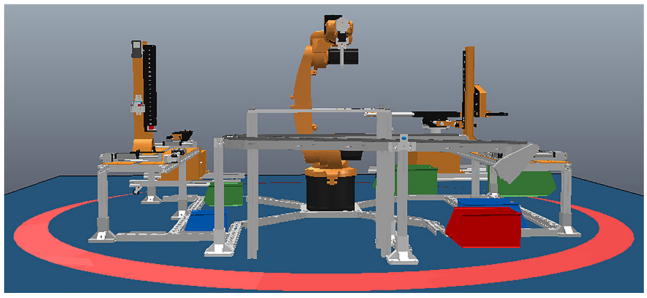


Fig. 6. Imported robot cell 'Own source'

The image generation process itself starts by first writing in promt what you want to see in the 512 × 512 resolution image. Then, as the model runs, it randomly generates noise, which it starts denoising in steps, and that's when the generated image we want starts to take shape. In our case, the prompt is "a green balloon is in front of the wall", which is also used to test that a model can be called and the Google Colab service is available (see Fig. 7).

5. GENERATE CUSTOM TRAJECTORY MAPS & TEXTURE WITH DREAMBOOTH

There has been a significant demand for innovative methodologies to generate novel assignments for our students. Even though point-to-point (PTP), linear (LIN), and circular (CIRC) movements are possible in the virtual environment, the use of a geometric shape map stuck to a metal sheet in the physical robot cell for foundational measurement purposes is somewhat limiting. This situation arose due to the immutability of the plate's placement and the fixed nature of its constituent elements. The geometric representation can be converted into a digital simulation as a texture; however, the texture's geometric configuration will remain unaltered.

The study utilized deep learning technology [25], an open-source framework enabling unrestricted training for any model. The dataset used by the generated model constituted the principal issue that required rectification. The utilization of Blender facilitated the creation of a customized geometric map, as the laboratory was limited to standard shapes that were commensurate with the plate utilized for the importation of the initial data.

The designed 2D geometric map comprises circles, squares, isosceles, triangles, and straight lines. The utilization of HSV (Hue, Saturation, Value) colors in Blender has been observed, specifically in utilizing black and white hues without any additional discernible color per shape being substantiated. The Cycle render engine [26] was utilized to derive the 2D object in the top view of the 3D graphics program. The camera positioned in the top view was programmed to rotate the image at 36° per frame on the Z axis. A total of 10 frames were rendered at a resolution of 512 × 512, as it is recommended to have a minimum of 10 frames for the models to be trained at the exact resolution, as depicted in Fig. 8.

Our reference model was trained using the generated datasets by implementing stable diffusion based on deep learning techniques [27]. Training is a fundamental requirement for text-to-image models, as without it, they cannot generate novel representations of items within a

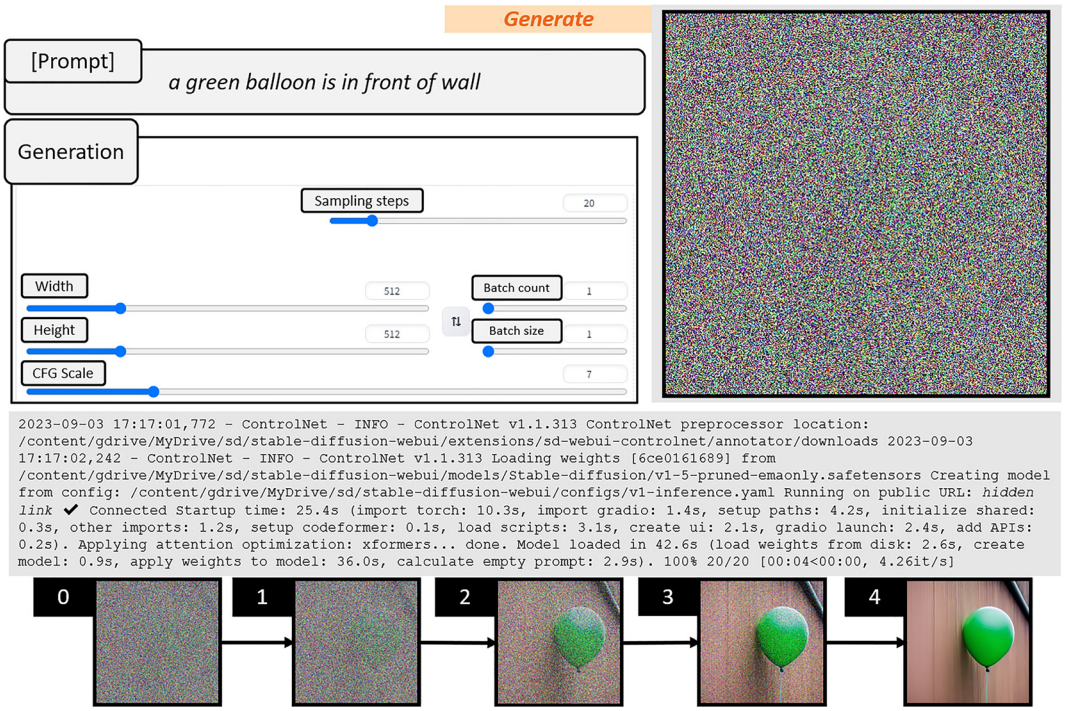


Fig. 7. Stabel diffusion image generation via promts 'Own source'

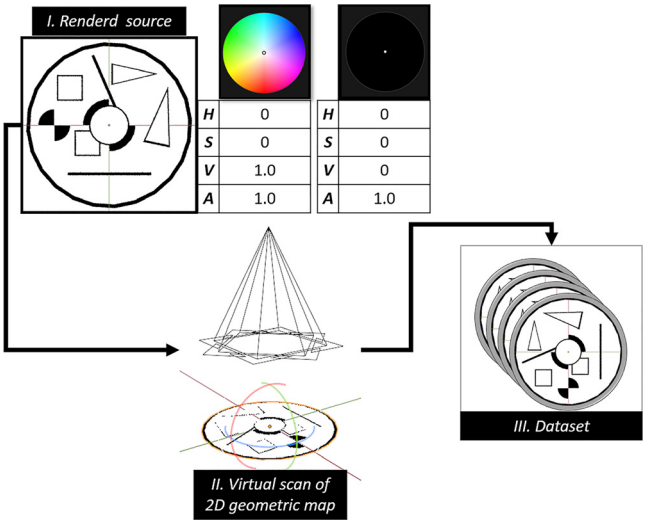


Fig. 8. The creation of dataset aimed at achieving stable diffusion 'Own source'

given reference set. In contrast, when a limited number of photographs are selected as training input and a specific trigger word is assigned, the model can effectively learn to generate outputs that exhibit a distinct style consistent with the input. As a result, the training data characteristics can be preserved in the resulting outputs.

The Dreambooth approach, as denoted by reference [28], exhibits certain limitations. One requirement is that the hardware must possess a high level of performance, with a minimum of 10 GB of VRAM, in addition to the prerequisite that the dataset (image) being trained must have dimensions of 512 × 512. Moreover, in the case of intricate simulations and the production of images, the computation predominantly relies on the graphics processing unit (GPU). Google Colab [29] provides users with complimentary access to a server provided by Google, enabling them to execute applications on said server. Utilizing a cloud server was employed in developing a pre-constructed text-image diffusion model to circumvent hardware limitations. The standard setup comprises an NVIDIA Tesla architecture T4 GPU with 16 GB of VRAM [30], an Intel Xeon CPU [31], and 12 GB of RAM. Upon successful login using a Gmail account, users can access the Google Colab top-up feature, which includes the Dreambooth function for reliable dissemination.

The DreamBooth Colab Notebook, which incorporates fast-stable diffusion, was utilized in our study [32]. During program execution, the files will be stored in a designated location, from which the image files to be utilized will be accessed. This location is established upon the initial authorization of our Drive account. Subsequently, Hugging Face [33] will generate a token essential for accessing the Colab cloud repository. It is imperative to note that the stable diffusion v1.5.ckpt (checkpoint) based [34] data model will be obtained from the source mentioned above. Subsequently, the DreamBooth methodology will be employed to instruct stable diffusion with our proprietary image data, culminating in the creation of a novel model [35]. Upon completing the download process, a session name (geometric map) will be assigned to the model intended for instruction.

Subsequently, the generated image files must be allocated a consecutive numerical identifier and uploaded to the disk

repository. In the present instance, the object under consideration is a geometric map with a unique identification number. Subsequently, to facilitate the training of the image files, it is imperative to establish two fundamental parameters. To effectively train the U-Net (Convolutional Neural Network for Image Segmentation) model [36], it is recommended to set the Steps value to 650 when working with a dataset of 10 image files. The second crucial factor pertains to the Text Encoder Training Steps [37], which are recommended to fall within the range of 250–450 for small-scale datasets. A value of 400 has been chosen for this particular scenario. Despite the abundance of hardware resources allocated for cloud-based Dreambooth training, the training process requires 1.2 h to reach completion from initiation. Subsequently, the file is expeditiously saved in the .ckpt file format into a Drive directory with a storage capacity of 1.98 gigabytes.

The logical flow of the image generation process consists of several steps (Fig. 9).

In general, the autoencoder, the U-Net, and the CLIP Text-Encoder are the three main parts. The two parts of an auto-encoder that work together to make the final image are the encoder and the decoder. That part converts the image to be generated for the U-Net, and the decoder part converts it back into an image. The role of the U-Net in the process is to convert the lower-resolution image representation into a higher-resolution image representation. The conversion process can generate noise, which is also corrected. The Text-Encoder interprets the input prompt, which serves as the content of the image to be generated [38].

In order to reduce the hardware resource requirements, low-resolution images are generated (64 × 64), which are later upscaled (512 × 512).

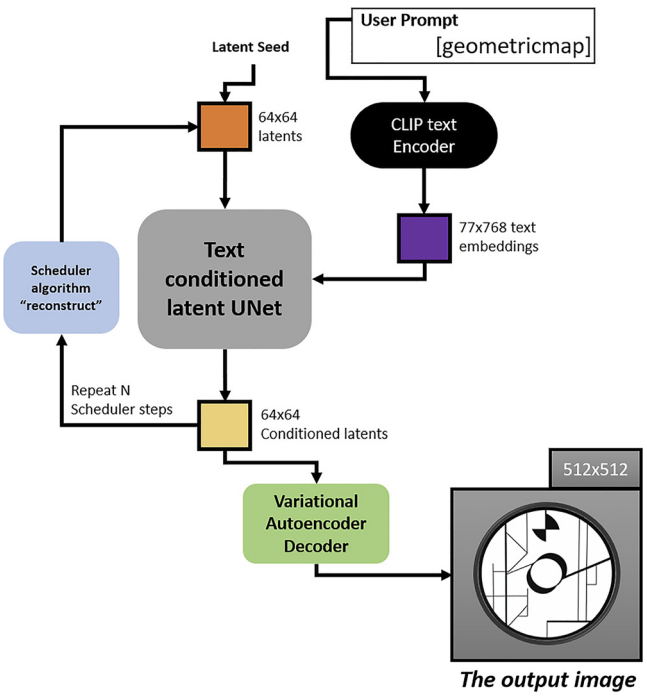


Fig. 9. The logical flow of image generating 'Own source'

In our case, more image files can be produced if we import the .ckpt file into the AUTOMATIC1111 web graphical user interface [39], as depicted in Fig. 10.

The .ckpt file was loaded into Automatic1111, a web GUI extension, to ensure stable dissemination. In the generation phase, the algorithm utilized was Euler, whereas for the sampling step, a value of 20 was assigned to it. They are increasing the value, which results in an enhancement of the visual fidelity of the image, albeit at the cost of a significant escalation in computational complexity. The significance of the CFG (Classifier-Free Guidance Scale) scale value lies in its impact on the extent to which the given prompts (geometricmap) are considered during the texture generation process. A smaller numerical value also signifies reduced adherence to the prompting frame, resulting in decreased coupling among the trained elements.

The images produced from the dataset were then compared with each other. This required the original image from which the dataset was generated and four randomly selected images from the generated images. Then, we uploaded the images to the free diff-Checker [40] online web interface, which matched the image files (see Fig. 11).

As can be seen in the figure, the difference between the images in % is also highlighted, and since the comparison is made at the pixel level, non-identical parts are highlighted in pink.

An identical technique was employed to enhance the authenticity of the models, whereby photographs of the authentic robot apparatus were utilized as an input source for textures. Subsequently, an additional model was trained with a specific emphasis on robot textures exclusively (see Fig. 12).

The texture we wanted to apply to the model was then selected. The contrast of textures in virtual environments can change depending on the lighting (they can be brighter or darker).

After completing the task mentioned above, we acquired the requisite textures for importing.

6. THE INVERSE & FORWARD KINEMATICS

Because of the importance of knowing the body's position at all times, forward and inverse kinematics are essential in this case. In addition, knowing the angular positions of the axes of the robot arm allows one to locate the TCP (Tool Centre Point) basic coordinate system.

Calculating forward kinematics (Direct Kinematics) for the given robot arm configuration yields a TCP situation. This must be determined by utilizing the dimensions and capabilities of the KUKA KR5.

As we know from kinematics synthesis studies in robotics, each link is connected to the subsequent link by some sort of joint. Every joint always connects two links, but the links are not necessarily connected by two joints. This is valid for the base and end effectors of the manipulator.

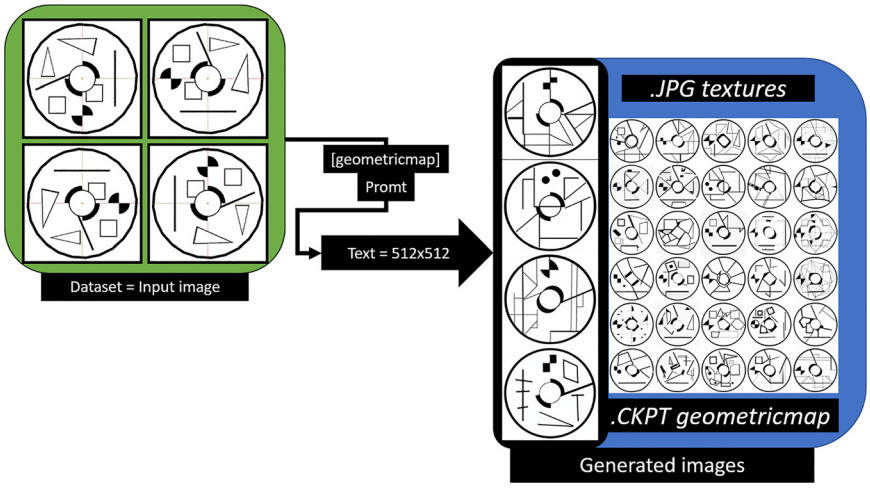


Fig. 10. The textures are produced using a geometric map 'Own source'

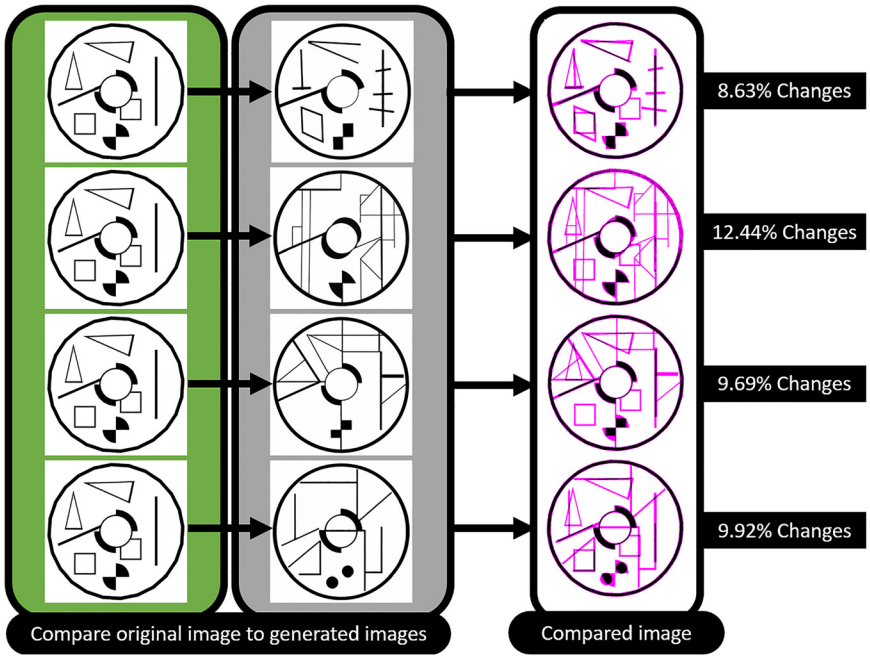


Fig. 11. Comparison of the original image with the generated one 'Own source'

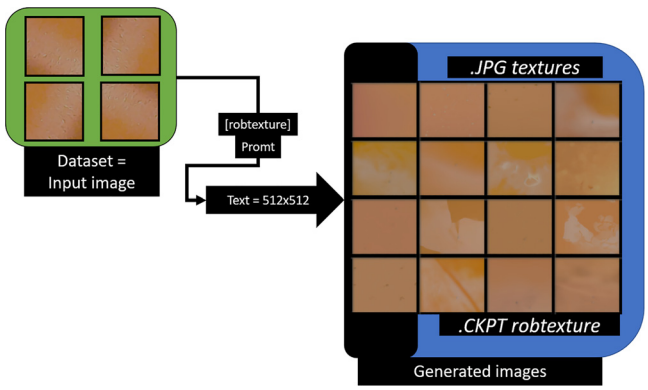


Fig. 12. Robot texture based on deep learning 'Own source'

The Denavit-Hartenberg matrix can be used to solve direct kinematic problems. The matrix is a straightforward method for describing members and wrists that can be applied to any robot configuration, regardless of its complexity [41]. Iteratively, each joint is determined by the required transformation from the previous joint. In other words, we designate a wrist to be n. Following that, n and the next joint are given a local reference frame [42].

The z-axis, which can be rotational or translational, defines each joint in local frame recording. Figure 13 and Table 1 are examples.

The example was created in the free MyApp simulation builder [43].

In our case, the parameters of the imported and positioned KUKA KR5 DH in V-REP are shown in Fig. 8 and the table below.

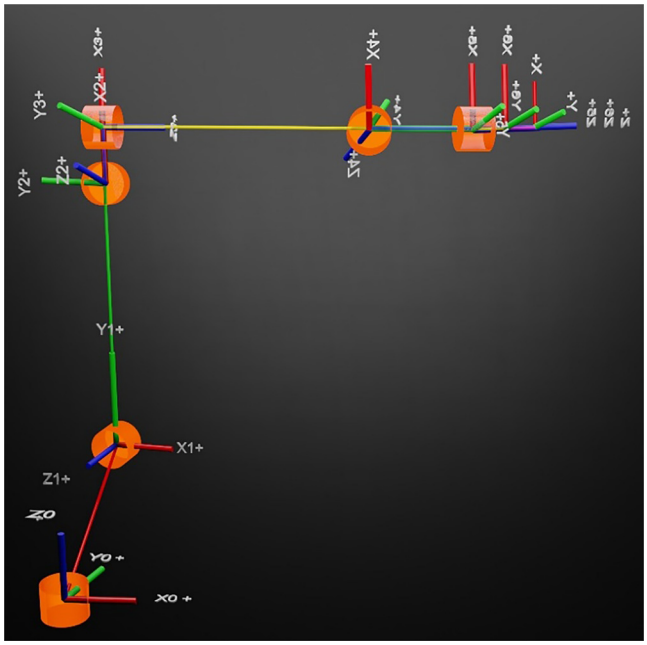


Fig. 13. Description of link and joints 'Own source'

Table 1. KUKA KR5 Arm Denavit-Hartanber parameter 'Own source'

Joints	Joint Offset *(d)* [mm]	Joint Angle *(θ)* [changeable]	Link Length *(a)* [mm]	Twist Angle *(α)* [degree]
1	400.014	θ_1	180.004	90
2	0	θ_2	600.005	0
3	0.003	θ_3	120.002	90
4	620.044	θ_4	0	90
5	0.002	θ_5	0	90
6	0	θ_6	0	0

The following are the meanings of the symbols used in the DH table:
θ: Rotation around a given Z-axis
d: Z-axis-aligned displacement (distance)
a: Offset joint (typical normal length)
α: Two Z-axes consecutively added together (in a joint twist) [44]

Inverse kinematics, as opposed to forward kinematics, is a much more complicated computational methodology. In the case of inverse kinematics, the position of the given robot unit is known, and the wrist variables are sought.

The robot's geometric model is the key to understanding its location in space. The kinematic model, in turn, explains problems with the velocity of the robot's individual parts. This necessitates the use of the Jacobi matrix, which differentiates between joint velocities and terminal speeds (End-Effector).

We need a Kinematic model to model the speeds of the robot and for that we need the following:

The joint speed:

- Rotational joint (angular velocity with respect to the axis of rotation).
- Joint translation (velocity linear along the axis of motion).

End-effector velocity:

- Angular velocity relative to x,y,z;
- Velocity along the x,y,z axes [35].

$$y = \begin{pmatrix} \nu \\ \omega \end{pmatrix}, where\ \nu = (\nu_x\ \nu_y\ \nu_z)\ \omega = (\omega_x\ \omega_y\ \omega_z) \quad (1)$$

It is necessary to start from a direct geometric problem when determining the relation.

The End-Effector's position and it's orientation.

$$x(\underline{q}) = \left[p_x\ (\underline{q})\ p_y\ (\underline{q})\ p_z\ (\underline{q})\ \alpha_x\ (\underline{q})\ \alpha_y\ (\underline{q})\ \alpha_z\ (\underline{q})\right] \quad (2)$$

A D-H ${}^{0}_{n}[T]$ last column 1–3 elements.

$$\dot{x} = \frac{\partial x}{\partial q} \cdot \dot{q}, \text{where } \frac{\partial x}{\partial q} = J = \begin{bmatrix} \frac{\partial x_1}{\partial q_1} & \frac{\partial x_1}{\partial q_2} & \cdots & \frac{\partial x_1}{\partial q_m} \\ \vdots & & \ddots & \vdots \\ \frac{\partial x_n}{\partial q_1} & \frac{\partial x_n}{\partial q_2} & \cdots & \frac{\partial x_n}{\partial q_m} \end{bmatrix} \quad (3)$$

Where:

m – Robotic degrees of freedom
n – Dimensions of the robot's working area
J – The robot's Jacobi matrix

The kinematics problem can be solved by finding the inverse Jacobi matrix:

$$\dot{q} = J^{-1}(q)\dot{x} \quad (4)$$

End-Effector and wrist acceleration relationship:

$$\ddot{x} = J(q)\ddot{q} + \dot{J}(q)\dot{q} \quad (5)$$

In light of the foregoing, the IK plugin environment of V-REP will be invoked and used. Because each robot is unique, "Base," "TCP," and "Target" must be specified as dummy units (shown in Fig. 14).

7. ROBOT CONTROL USING PTP, LIN, & CIRC MOTION

IK and FK were added to our current robot model so that we could later use the PTP (Point-To-Point), LIN (Linear), and CIRC (Circular) motion types.

However, because the goal is to precisely replicate the robot processes that we perform during physical education testing in the industrial robot lab, the base has also been imported as a texture and model [45]. The difficulty stems from the physical robot arm's inclusion of a control cabinet (KRC2) and a teaching pendant. They are not, however, Open-Source certified, so their replacement is essential if we want to use the movements typical of a known robot arm, which can be done using our own scripts.

PTP: Forward Kinematic was used during robot motion for PTP, or point-to-point motion. For this particular robot unit, the tool will select the quickest route to the desired location.

```
function sysCall_init()
    --way:
    local simBase=sim.getObjectHandle('Base')
    local simTip=sim.getObjectHandle('TCP')
    local simTarget=sim.getObjectHandle('Target')
    -- create an IK environment:
    ikEnv=simIK.createEnvironment()
    -- create an IK group:
    ikGroup_undamped=simIK.createIkGroup(ikEnv)
    -- set its resolution method to undamped:
    simIK.setIkGroupCalculation(ikEnv,ikGroup_undamped,simIK.method_pseudo_inverse,0,6)
    -- create an IK element based on the scene content:
    simIK.addIkElementFromScene(ikEnv,ikGroup_undamped,simBase,simTip,simTarget,simIK.constraint_pose)
    -- create another IK group:
    ikGroup_damped=simIK.createIkGroup(ikEnv)
    -- set its resolution method to damped:
    simIK.setIkGroupCalculation(ikEnv,ikGroup_damped,simIK.method_damped_least_squares,1,99)
    -- create an IK element based on the scene content:
    simIK.addIkElementFromScene(ikEnv,ikGroup_damped,simBase,simTip,simTarget,simIK.constraint_pose)
```

Fig. 14. Program for solving IK tasks 'Own source'

This type of motion is replicated in our case in such a way that when the virtual robot unit arrives at the point we specify in the workspace, the coordinates of the various joints are stored there.

We can store these positions from 1 to 6 in matrix form by calling Poscollect = function (quantity). To execute the motion the joints of the robot turn from the current to the required joint coordinate.

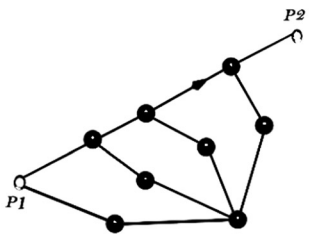
LIN: There is no LIN command, but joint movements can be created. The earlier mentioned Target point is located within the robot's working area, and linear progression is the desired outcome. As a result, we create a prismatic joint whose position, where the Target point is in 3D space, and orientation can be specified. The LIN movement becomes possible once this is assigned to the joint (shown in Fig. 15).

In this case, the rotation includes not only the rotation along the axes but also the Euler angles, which our program code calculates when the robot configuration is executed.

CIRC: Three points are typically assigned to the CIRC motion type. There are three starting points, one auxiliary point, and one end point (shown in Fig. 16).

Keeping this in mind, the code we wrote when we made the CIRC motion will also have three points. However, we won't specify the base position of the virtual KUKA KR5 (which will be the starting point), only "through" and "to."

```
LIN=function(TO)
 FROM=sim.createDummy(0)
    sim.setObjectPosition(FROM,Target,{0,0,0})
 Distance=sim.getObjectPosition(TO,FROM)
 Dis=math.sqrt((Distance[1])^2+(Distance[2])^2+(Distance[3])^2)
 Z1=(Distance[1])/(Dis)
 Z2=(Distance[2])/(Dis)
 Z3=(Distance[3])/(Dis)
if Z3 == 0 then
    sim.setObjectOrientation(FROM,world,{0.1,0.1,0.1})
 Distance=sim.getObjectPosition(TO,FROM)
 Dis=math.sqrt((Distance[1])^2+(Distance[2])^2+(Distance[3])^2)
 Z1=(Distance[1])/(Dis)
 Z2=(Distance[2])/(Dis)
 Z3=(Distance[3])/(Dis)
end
local J0=sim.createJoint(sim.joint_prismatic_subtype,sim.jointmode_passive,0,{0.00001,0,0001},nil)
    sim.setObjectPosition(J0,FROM,{0,0,0})
    sim.setJointInterval(J0,false,{-100,200})
    if Z3 < 0 then
    sim.setObjectOrientation(J0,FROM,{math.pi+math.atan(-(Z2)/(Z3)),(math.asin(Z1)),0})
    else
    sim.setObjectOrientation(J0,FROM,{math.atan(-(Z2)/(Z3)),(math.asin(Z1)),0})
    end
    sim.setObjectParent(Target, J0, true)
    sim.moveToConfig(-1,{0},nil,nil,{0.05},{1},{1},{Dis},nil,callback_ik,J0)
    sim.setObjectParent(Target, TO, true)
    sim.removeObject(J0)
    sim.removeObject(FROM)
end
```



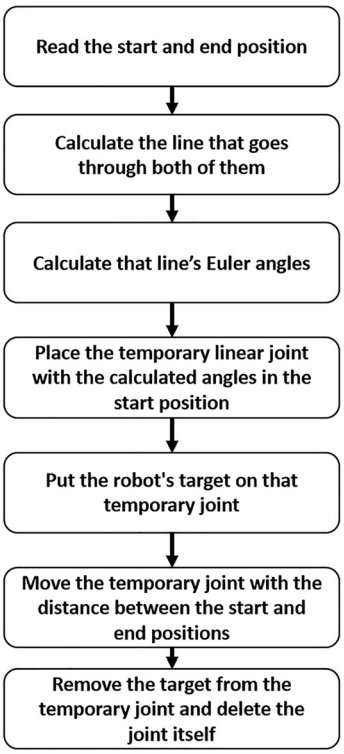


Fig. 15. LIN calculates motion program 'Own source'

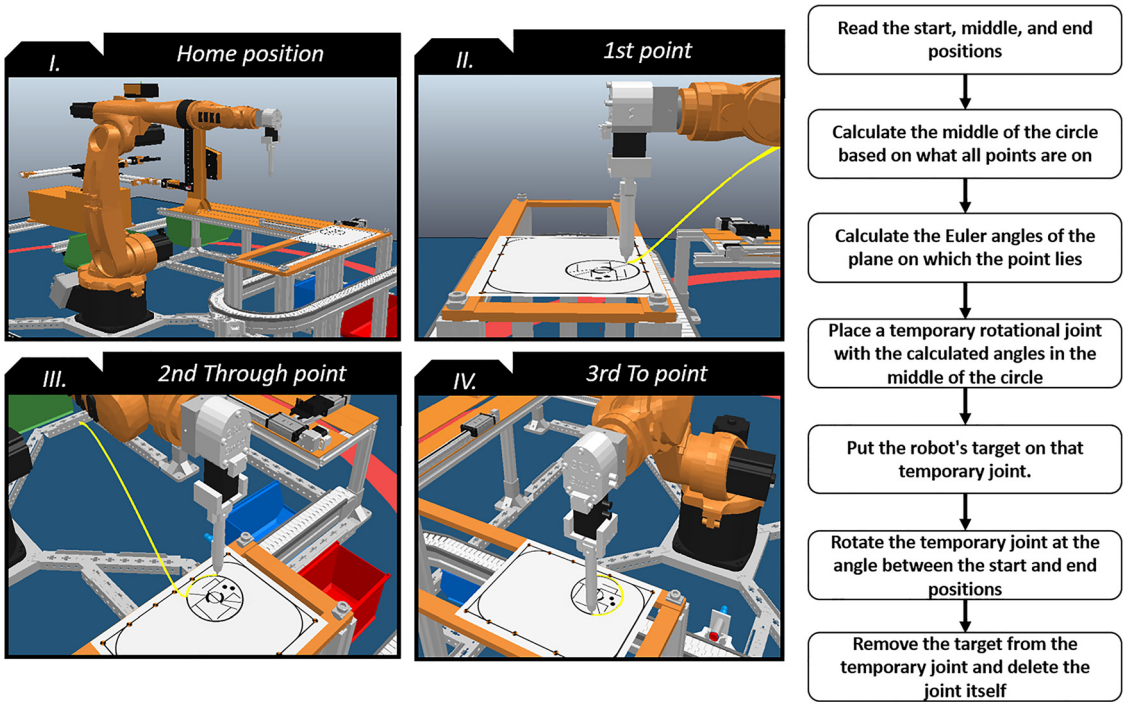


Fig. 16. Motion planning with CIRC 'Own source'

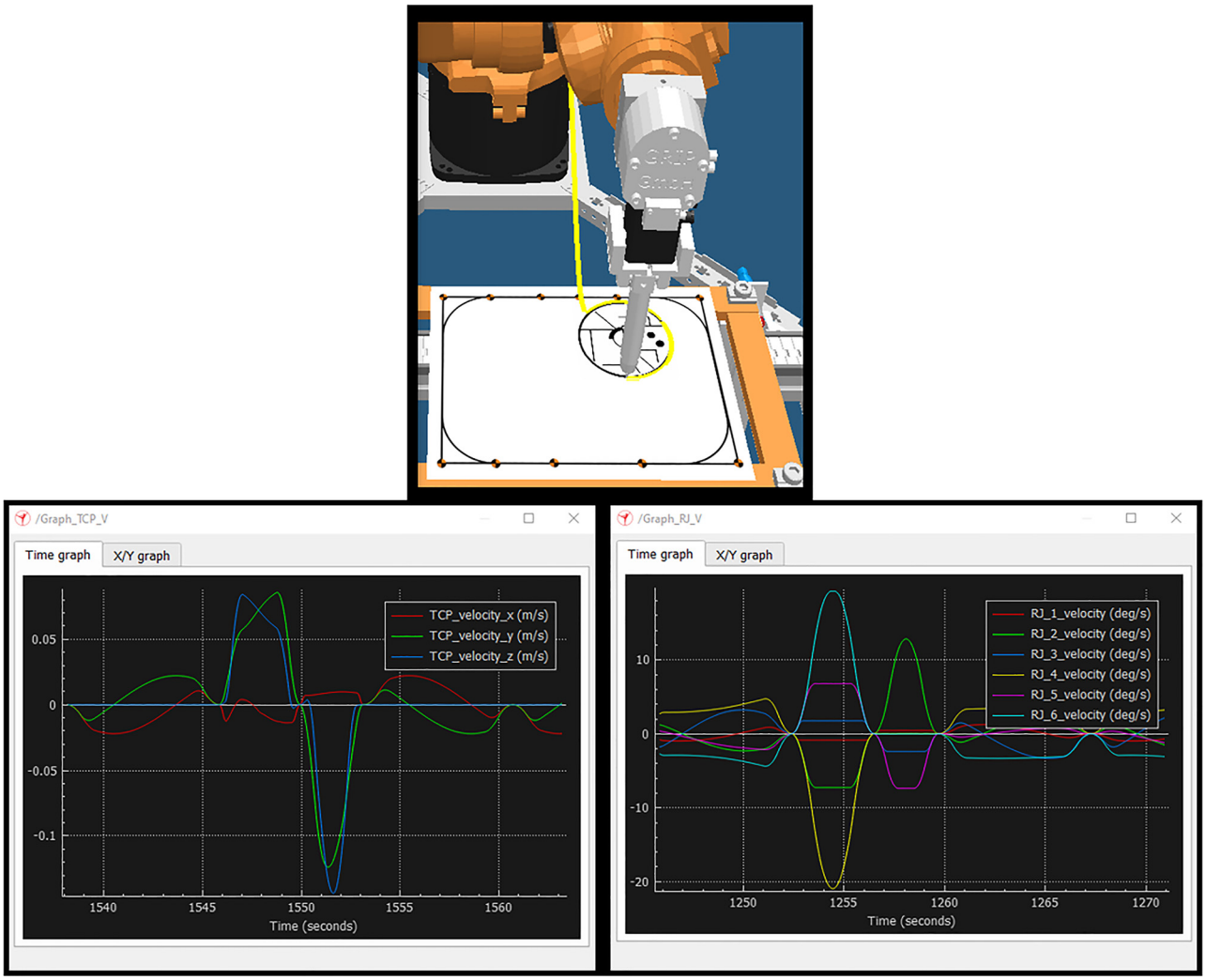


Fig. 17. Virtual measurement of TCP RJ velocity 'Own source'

In the script, the former is called the Auxiliary point, and the latter is called the End point.

From these three points, you can figure out where the centre of the circle is.

So that the path of the virtual robot unit is easier to see, it has been given a colour that is different from the background. This colour is yellow in our case.

8. VIRTUAL ROBOT MEASUREMENT & ANALYZATION

We can perform virtual measurements in the Cyber-Physical Space virtual environment that we have created. The measurements provide us with direction for future analyses. This is due to the possibility of differences between a real, physical KUKA KR5 and a KUKA KR5 in cyberspace. As a result, further adjustments to the measurements can be made in order to model the most accurate behaviour of the robot unit. TCP V and RJ V are virtual measurements. The former will calculate and graph the speed of the Tool Centre Point (TCP V). The latter will determine the rotational speed of the Rotation Joints (RJ V) along all six axes. Two graph elements were created during the implementation and are referenced in the function sysCall init (). Then we enter the values into the program code, such as the X, Y, and Z speeds, as well as the unit of measurement and the colours.

In the case of TCP, the values are displayed in millimeters per second (m s^{-1}) and RJ (degrees per second). The developed system can be used to control more than one robot arm. Any other individual point in space can be specified and measured graphically (shown in Fig. 17).

It is important to note that the tool measuring unit of the KUKA KR5 robot unit is in production, so the physical measurement and its comparison will be part of a later study. However, what is clearly different between the virtual and physical robot units is that for the virtual robot arm and robot cell, any additional 3D model can be added and removed. Also, compared to the physical robot unit, it is more cost-effective as it has no electrical consumption and no wear and tear during use. However, the control system of the robot itself is not open-source, meaning that there are differences in the execution of individual commands.

9. CONCLUSION

To improve the quality of education, innovative techniques are being used. A digital twin model of the industrial laboratory has been created, which includes the 3D models and their specific shape characteristics. In all cases, the development of the physics laboratory was preceded by 3D CAD modelling, so we had a complete asset set. However, the models were too detailed and had to be optimised to reduce the demand on hardware resources. The optimization was performed on each model. The created virtual system functions as a digital framework, and its flexibility allows for adding or removing additional 3D CAD elements. This allows particular problems to be studied under different environmental conditions, which can be measured virtually. In the virtual robot cell, we could apply a modified tool head assembly that was not produced by a 3D printer until later. During the test, the tool head unit was put on a copy of a multi-axis KUKA KR5, and PTP, LIN, and CIRC movements were done while the tool centre point and the speed of the joints were measured virtually. The performed virtual measurements can be compared with the real physical robot unit later on, and any discrepancies in the trajectory calculations can be investigated. A measuring tool for the physical robot unit is currently under development, so the comparison will be part of a future study.

On the other hand, the digital twin system has many uses, as it can be used both “online” and “offline” in education. Before the course, students can use this technique to acquire the skills needed for general robot handling. Although the virtual environment allows different types of robot movements, the sticker on the metal plate used initially for base measurement, which contained the basic 2D geometric shapes, was severely limited. To improve the educational experience, we used deep learning techniques to generate personalized trajectory navigation tasks for the users, which we taught using our rendered images. The images generated from our own dataset were compared and ranked as a percentage of each other to get a more accurate picture of whether there was a significant difference between them. This approach allows us to avoid the repetition of routine PTP, LNI, and CIRC movements while also allowing us to perform individualized tasks, thus preventing the possibility of creating a false sense of security.

Taking the above into account, the digital twin can be used. Considering the above, the digital twin can be used to test individual items, even robotic grippers, and to pre-train users so that the logistics of other developments in the physical lab can be better coordinated.

Conflict of interest: The third author, Géza Husi is a member of the Editorial Board of the journal. Therefore, the submission was handled by a different member of the editorial team and he did not take part in the review process in any capacity.

ACKNOWLEDGEMENTS

Project No. TKP2020-NKA-04 has been implemented with the support provided by the National Research, Development, and Innovation Fund of Hungary, financed under the 2020-4.1.1-TKP2020 funding scheme.

REFERENCES

[1] A. Szántó, É. Ádámkó, G. Juhász, and G. Á. Sziki, “Simultaneous measurement of the moment of inertia and braking torque of electric motors applying additional inertia,” *Measurement*, vol. 204, 2022, Art no. 112135.

[2] G. Á. Sziki, A. Szántó, J. Kiss, G. Juhász, and É. Ádámkó, "Measurement system for the experimental study and testing of electric motors at the faculty of engineering, University of Debrecen," *Appl. Sci.*, vol. 12, no. 19, 2022, Art no. 10095.

[3] A. Szántó, S. Hajdu, and G. Á. Sziki, "Optimizing parameters for an electrical car employing vehicle dynamics simulation program," *Appl. Sci.*, vol. 13, p. 8897, 2023. https://doi.org/10.3390/app13158897.

[4] A. Szántó, J. Kiss, T. Mankovits, and G. Á. Síki, "Dynamic test measurements and simulation on a series wound DC motor," *Appl. Sci.*, vol. 11, no. 10, p. 4542, 2021.

[5] S. Yildirim, "Design of neural network control system for controlling trajectory of autonomous underwater vehicles," *Int. J. Adv. Robotic Syst.*, vol. 11, pp. 1–17, 2014. https://doi.org/10.5772/56740.

[6] L. Rivera, H. Müller, N. Villegas, G. Tamura, and M. Jiménez, "On the engineering of IoT-intensive digital twin software systems," in *Proceedings of the IEEE/ACM 42nd International Conference on Software Engineering Workshops*, Seoul, Korea, 27 June–19 July 2020.

[7] S. Aheleroff, X. Xu, R. Y. Zhong, and Y. Lu, "Digital twin as a service (DTaaS) in industry 4.0: an architecture reference model," *Adv. Eng. Inform.*, vol. 47, 2021, Art no. 101225.

[8] E. Samadi and I. Kassou, "The relationship between IT and supply chain performance: a systematic review and future research," *Am. J. Ind. Bus. Manag.*, vol. 6, pp. 480–95, 2007.

[9] A. Gereco, M. Caterino, M. Fera, and S. Gerbino, "Digital twin for monitoring ergonomics during manufacturing production," *Appl. Sci.*, vol. 10, p. 7758, 2020.

[10] R. Carvalho and A. R. da Silva, "Sustainability requirements of digital twin-based systems: a meta systematic literature review," *Appl. Sci.*, vol. 11, p. 5519, 2021.

[11] C. Verdouw, B. Tekinerdogan, A. Beulens, and S. Wolfert, "Digital twins in farming systems," *Agric. Syst.*, vol. 189, 2021, Art no. 103046.

[12] I. Rojek, D. Mikołajewski, and E. Dostatni, "Digital twins in product lifecycle for sustainability in manufacturing and maintenance," *Appl. Sci.*, vol. 11, p. 31, 2021.

[13] E. Dostatni, I. Rojek, and A. Hamrol, "The use of machine learning method in concurrent ecodesign of products and technological processes," in *Advances in Manufacturing. Lecture Notes in Mechanical Engineering*, A. Hamrol, O. Ciszak, S. Legutko, and M. Jurczyk, Eds., Cham: Springer, 2018. https://doi.org/10.1007/978-3-319-68619-6_31.

[14] T. I. Erdei, Zs. Molnár, N. C. Obinna, and G. Husi, "A novel design of an augmented reality based navigation system & its industrial applications," in *15th IMEKO TC10 – Technical Diagnostics in Cyber-Physical Era – Organised by: MTA SZTAKI – Hungarian Academy of Sciences – Institute for Computer Science and Control.* Budapest, Hungary, 6 – 7 June, 2017.

[15] KUKA Robotics, "Official documentation of Industrial ARC Welder robot arm," https://www.eurobots.net/robot_kuka_kr5_arc-en.html. Accessed: Nov. 01, 2022.

[16] S. Wu, Q. Qiang Lin, and P. Wang, "Kinematics analysis and simulation of KUKA KR5-arc welding robot based on MATLAB," in *6th International Conference on Information Engineering for Mechanics and Materials*, ICIMM, 2016.

[17] T. I. Erdei and G. Husi, "Hibajavítás és megelőzés 3D printer FDM gyártási folyamat alkalmazása során," University of Debrecen – Thesis material. 20.12.2017.

[18] IndustryDays-2021, University of Debrecen "Online conference with virtual exhibition", https://hirek.unideb.hu/index.php/online-konferencia-virtualis-kiallitassal. Accessed: Nov. 01, 2023.

[19] D. Horváthová, P. Voštinár, and M. Mitter, "Using blender 3D for learning virtual and mixed reality," in *Conference: Proceedings of ICERI2019 Conference*, November 2019. https://doi.org/10.48550/arXiv.2211.01777.

[20] K. Mchenry and P. Bajcsy, "An overview of 3D data content, file formats and viewers," December 2008.

[21] L. Pitonakova, M. Giuliani, A. Pipe, and A. Winfield, "Feature and performance comparison of the V-REP, Gazebo and ARGoS robot simulators," in *Conference: Proceedings of the 19th Towards Autonomous Robotic Systems Conference (TAROS 2018)* At: Bristol, United Kingdom.

[22] R. Rombach, A. Blattmann, D. Lorenz, P. Esser, and T. Shimizu, "High-resolution image synthesis with latent diffusion models," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, June, 2022. https://doi.org/10.48550/arXiv.2112.10752.

[23] N. Kumari, B. Zhang, R. Zhang, E. Shechtman, and J-Y. Zhu, "Multi-concept customization of text-to-image diffusion," *10.48550/arXiv.2212.04488*, 2022.

[24] R. Hidalgo, N. Salah, R.C. Jetty, A. Jetty, and A. Varde, "Personalizing text-to-image diffusion models by fine-tuning classification for AI applications," in *Conference: Intelligent Systems Conference (IntelliSys) 2023*, Springer Publishers At: Amsterdam, The Netherlands.

[25] H. Munir, B. Vogel, and A. Jacobsson, "Artificial intelligence and machine learning approaches in digital education: a sys-tematic revision," *Information*, vol. 13, p. 203, 2022. https://doi.org/10.3390/info13040203.

[26] I. A. Astuti, I. H. Purwanto, T. Hidayat, D. A. Satria, Haryoko, and R. Purnama, "Comparison of time, size and quality of 3D object rendering using render engine eevee and cycles in blender," in *2022 5th International Conference of Computer and Informatics Engineering (IC2IE)*, Jakarta, Indonesia, 2022, pp. 54–9. https://doi.org/10.1109/IC2IE56416.2022.9970186.

[27] N. Dehouche and K. Dehouche, "What's in a text-to-image prompt? The potential of stable diffusion in visual arts education," *Heliyon*, vol. 9, 2023, Art no. e16757. https://doi.org/10.1016/j.heliyon.2023.e16757.

[28] N. Ruiz, Y. Li, V. Jampani, Y. Pritch, M. Rubinstein, and K. Aberman, "DreamBooth: fine tuning text-to-image diffusion models for subject-driven generation," arXiv preprint arxiv:2208.12242, 2022.

[29] M. Canesche, L. Bragança, O.P. V. Neto, J. A. Nacif, and R. Ferreira, "Google colab CAD4U: hands-on cloud laboratories for digital design," in *2021 IEEE International Symposium on Circuits and Systems (ISCAS)*, Daegu, Korea, 2021, pp. 1–5. https://doi.org/10.1109/ISCAS51556.2021.9401151.

[30] H. Choi and J. Lee, "Efficient use of GPU memory for large-scale deep learning model training," *Appl. Sci.*, vol. 11, 2021, Art no. 10377. https://doi.org/10.3390/app112110377.

[31] S. Akter, R. A. Prodhan, T. S. Pias, D. Eisenberg, and J. Fresneda Fernandez, "M1M2: deep-learning-based real-time emotion recognition from neural activity," *Sensors*, vol. 22, p. 8467, 2022. https://doi.org/10.3390/s22218467.

[32] N. Kumari, B. Zhang, R. Zhang, E. Shechtman, and J-Y. Zhu, "Multi-concept customization of text-to-image diffusion," 2022. https://doi.org/10.48550/arXiv.2212.04488.

[33] HuggingFace, "Open-source code and technologies,". https://huggingface.co/docs/transformers/index. Accessed: Jan. 01, 2023.

[34] R. Rombach, A. Blattmann, D. Lorenz, T. Esser, and Shimizu, "High-resolution image synthesis with latent diffusion models," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, June, 2022. https://doi.org/10.48550/arXiv.2112.10752.

[35] Dreambooth implanted in – Google Colab notebook cloud server. Available online: https://colab.research.google.com/github/TheLastBen/fast-stable-diffusion/blob/main/fast-DreamBooth.ipynb. Accessed: Dec. 18, 2022.

[36] L. Jiao, L. Huo, C. Hu, and P. Tang, "Refined UNet: UNet-based refinement Network for cloud and shadow precise segmentation," *Remote Sens*, vol. 12, 2020, 2001. https://doi.org/10.3390/rs12122001.

[37] S. Andreas, "Evaluating a synthetic image dataset generated with stable diffusion," 2022. https://doi.org/10.48550/arXiv.2211.01777.

[38] S. Patil, P. Cuenca, N. Lambert, and P. von Platen, "Stable diffusion,". https://huggingface.co/blog/rlhf. Accessed: May 05, 2023.

[39] AUTOMATIC1111/stable-diffusion-webui. https://github.com/AUTOMATIC1111/stable-diffusion-webui. Accessed: Jan. 05, 2023.

[40] Image Diff-Checker. https://www.textcompare.org/. Accessed: Sep. 10, 2023.

[41] S. B. Niku, *Introduction to Robotics Analysis Systems, Applications*. Upper Saddle River, NJ: Prentice Hall, 2001, pp. 67–71, 92–94.

[42] V. Abhishek, A. A. Hayat, A. D. Udai, and S. Saha, "Identification of dynamic parameters of an industrial manipulator," in *(IMSD 2014) and the 7th Asian Conference on Multibody Dynamics (ACMD 2014)*, At: Busan, Korea.

[43] RobotJS, "DH parameter calculator,". https://robot-viewer-qfmqx.ondigitalocean.app/cookbook. Accessed: May 05, 2023.

[44] M. Lőrinc and F. Áron, "Inverse geometry of robots," *Erdélyi Magyar Tudományegyetem, Scientia Kiadó*, 2019.

[45] KUKA Roboter, "KUKA system software V5.x," *KUKA Roboter GmbH Zugspitzstraße 140 D Augsburg Deutschland*.