OPTIMAL ROBOT POSITIONING FOR SUSTAINABLE PROCESS EXECUTION

Venkatachalam Srinivasan*, Yassine Ghanjaoui*, Philip Satwan*, Jörn Biedermann*, Björn Nagel*
* Institute for System Architectures in Aeronautics, Hein-Saß-Weg 22, 21129 Hamburg

Abstract

In this paper, the methodology to implement an inference algorithm based on the robot parameters for process execution on different Key Performance Indicators (KPI's) is presented. This work is part of holistic objective to implement a system to perform virtual validation of automated process with robots, where the impact of relative position of robot to required task and vice-versa could be inferred virtually. For the above implementation, it is proposed to develop the digital model of the necessary infrastructure in the simulation environment. To start with, computational data from simulation is stored to a database, which is then analysed and appropriate regression based inference model is formulated. To ensure that the digital model imitates the real-time system, the feedback data from hardware execution is used to improve the parameters of regression model. With this implementation, the digital model would represent the digital twin (DT) of the hardware under consideration. The whole execution is performed on the pre-assembly cell at the Institute for System Architectures in Aeronautics, Hamburg. Use case for the digital twin implementation is the pre-assembly of *overhead structural truss* that assists in realizing the modularized cabin assembly process. Implementation of simulation model for cabin assembly is a two-fold approach, where the robot localization for reachability is computed followed by computation of joint trajectories. From the obtained trajectories, the energy and time consumed by the robot for a given task is calculated. The computed information is then stored in a database, which is then fed to an inference algorithm. Implementation of the algorithm is desired based on the time taken for path computation, and using this algorithm the optimal *robot location* and *joint angles* for any new unknown task could be computed.

Keywords

virtual validation; digital twin; robotics; cabin; efficient process; simulation

1. INTRODUCTION

Automation in Aerospace has been less emphasized due to the reason that the product volumes are insignificant compared to the product life-cycle. Due to the fact that products are highly customizable depending on customer's requirements, deploying full-fledged automation in such environment is uneconomical. A relevant constraint to realize an automation system is the implementation deadline [1]. To improve automation's footprint in an economical manner, FMS[1] is deployed [2]. For a successful FMS[1] implementation, virtual commissioning(VC) plays greater role to test the necessary systems in advance for successful deployment [3] [4] [5].

Validating the shop-floor integration beforehand in order to speed up the real-time installation process, *VC*[2] is gaining importance [6]. It is estimated that VC reduces the number of on-site man hours for installation to 50%. This is achieved by building the production systems virtually and to simulate the commissioning process. This allows the organization to validate the operation of a new systems before implementing in the physical environment. Understanding the behaviour of the different systems plays important role in successful validation for VC[2] [7]. Usually in VC, it is essential to perform "*Hardware-in-loop (HIL)*" with PLC controlling the hardware or at best "*Software-in-loop*" with an emulated PLC [6]. The presented work tries to enhance VC, in an attempt to make it more informative and more virtual.

There are different ways to test the control systems [3] in an industrial shop-floor:

1. Both shop-floor and the control system are real (*Traditional way*).
2. Simulated shop-floor with real control system (*Simulated shop-floor*].
3. Simulated shop-floor control system in real shop-floor (*Reality in-loop*).
4. Both shop floor and it's control are simulated (*Offline simulation*).

Of all the above ways, offline simulation is cost effective, but the feasibility isn't proven since everything is virtual. All the other methods need either shop-floor or the control system to be physical. This work enhances the validation through offline simulation, such that the virtual analysis of the hardware would not only be cost effective but also paves the way for better shop-floor planning. Since it differs from traditional VC[2], the term *"virtual validation"* is used to indicate the work's contribution. Also it is important to understand why the term "digital twin" used is this work, although there is only virtual validation being made. In the holistic

[1] Flexible Manufacturing Systems
[2] Virtual Commissioning

implementation, the digital model would interact with the real world objects, and this model exhibits the real time hardware behaviour. Hence the term "digital twin" is used to represent the virtual model.

For successful understanding of behaviour virtually, it is important to generate the virtual model that exactly imitates the real hardware. With the generated virtual model, it would be possible to simulate and perform validation based on different KPIs[3] (such as cycle time, energy etc). Usually faster cycle time is preferred, but Industry 5.0 emphasizes the energy efficient automation for a sustainable production. Energy conservation is becoming highly important with increasing automation as energy becomes more expensive and scarce with increasing population and demand [8]. Importance of energy conservation could be justified by the number of researches performed on this topic [9].

Contributing to optimal process execution, DLR-SL[4] is working on implementing the digital thread for the aircraft cabin assembly process. In the perspective of assembly process, digital thread refers to link between modules starting from design until assembly. This work is part of digital thread, where implementation of *digital twin* model is envisioned for automated cabin assembly process. There are two stages in the digital twin implementation, first the computations are performed based on robot kinematics and dynamics model for different robot tasks and the results are stored in database. The results of the robot parameters estimation are compared with parameters from real time execution, and the error is computed. This error is used for tuning the learning parameter in the regression model. This implementation ensures that the digital twin of the hardware under consideration is complete and correct. With the accurately described digital twin, it would be then possible to perform virtual validation that is accurate and reliable. This paper illustrates the part of whole approach, in positioning the robot for different tasks and computing their associated time and energy.

The paper is structured in following manner: Section 2 describes the literature overview of existing work in robot energy optimization. Section 3 summarizes technical background on hardware infrastructure and process work-flow with regards to this work. Methodology of the whole implementation ideas is presented in section 4 followed by actual implementation to the overhead-structural truss use-case in section 5. The results of the implementation is discussed in section 6, where the effectiveness of regression model is analysed. Next steps is presented in section 7 that describes the further work of the holistic idea.

[3]Key Performance Indicators

[4]Institute for System Architectures - Deutsches Zentrum für Luft- und Raumfahrt e.V

2. LITERATURE REVIEW

Virtual Commissioning

As already mentioned existing approaches on VC[2] simulates the hardware with real control system. Tobias Lechler et. al [6] proposed digital twin based on virtual factory in virtual reality (VR), where real time bidirectional data integration between physical systems and VR is established. M. Dahl et.al [4] developed a formal model that is used for visualizing, analysing and simulating robot based production system. A framework *"Integrated Virtual Preparation and Commissioning"*(IVPC) is introduced that supports HIL and it performs continuous iterations on the "Virtual Manufacturing Model (VMM)". Process sequencing is performed by the control logic that is applied on VMM, and any changes to VMM structure (such as adding a new robot) would automatically update the control logic. An enhancement is developed in [10], where optimizing the process sequence based on dynamic VMM behaviour is shown. Multi-robot interactive control using mixed reality is proposed by [11] et. al., where a system architecture for multi-robot control in a common coordinate system is implemented. In their previous work [12], a mixed-reality(MR) interface to program and visualizing robot path is developed. Based on the goal defined by the user (via MR), the shortest path is computed. Moreover it is possible for the user to customize the computed path, by changing/introducing intermediate points and appropriate time optimal trajectory is computed. Schamp [5] et. al. used Automation ML(AML) for communicating information between different hierarchical levels in the Computer Integration Manufacturing(CIM) pyramid.

Shahria et. al. [13] introduced a mixed reality framework for collaborative robot in the context of patient's rehabilitation in field of medicine. A framework is established where the interaction between MR control to the digital twin (DT) of the robot is established, with DT mirroring the co-bot under consideration. DT data is then collected to a Azure platform, machine learning is proposed for understanding the patients' improvement based on correlating sensor data (like torque, temperature and force) to the ailment key metrics such as range of motion, resistance etc.

Time and energy conservation in industrial environment

Immense work has been performed in trajectory planning of robots with different objectives. Time optimization and smooth trajectory planning is proposed by Zhang,Tie [14], where the optimization is performed in parameter space of the TCP (Tool Center Point). The time optimal trajectory is obtained based on constraining acceleration and velocity. Trajectory optimization is then followed by vibration control using feed-forward method. The proposed method is tested with a kuka robot, where the TOSTP[5] yielded better results for residual vibration compared to TOTP[6]. Zribi,Sameh [15] proposed seven-degree polynomial function to suppress snap (derivative of jerk) in order to minimize residual vibrations. Continuous jerk is achieved for minimizing the vibrations and the smoothness of trajectory is verified with the acceleration profile.

There has been quite some work carried on the research in the field of energy conservation for robotic systems. A list of existing research work are provided by *Giovanni et. al* [9], which explains the different existing strategies for energy conservation from both hardware and software perspectives. Researches based on different hardware capabilities such as *robot type*, *energy storing devices* and *lighter robot arms* are presented. From Software perspective, the approaches on *trajectory optimization* and *operation scheduling* are presented. Software approaches are more relevant to our work, since *trajectory optimization* is envisioned for energy optimal process execution. Not all of the existing approaches consider payload characteristics while designing energy efficient trajectories. The other approach on *operation scheduling* is out-of scope for this work.

Scalera, Lorenzo et al. [16] proposed a methodology to optimally place the robot task for a 4-DOF[7] manipulator for minimal energy consumption. Using *Lagrangian approach*, the required torque is computed for a task with *trapezoidal* speed profile. Energy consumption is calculated and influence of start and target points of robots within robot workspace is presented. The placement of task in the robot workspace could be determined based on the analysis for minimal energy consumption. Spensieri, Domenico, et al [17] presents the methodology in positioning the robot for specific task, where the robot location is ascertained for minimal cycle time. The evaluation of robot location is carried for different complex levels (such as *number of spot welding points*, *no collision scenario* and *brute force method for minimal cycle time*) for welding application.

Meenakshi Sundaram, Ashok [18] developed robot capability maps (CMAPs) for robot assisted surgeries. The idea is to find the optimal location for robot, so to avoid any complications on robot reachability during the time of surgery. Task-based preoperative setup is proposed based on genetic algorithms (GA), with highest possible dexterity and good overall reachability. Computing the inverse kinematics for reachability followed by inverse dynamics for different configurations is presented by Mohammed Abdullah et. al. [19], where the first three robot joints are used for computing inverse kinematics on robot's reachability. The joint configurations from inverse kinematics are then used for inverse dynamics computation, wherein

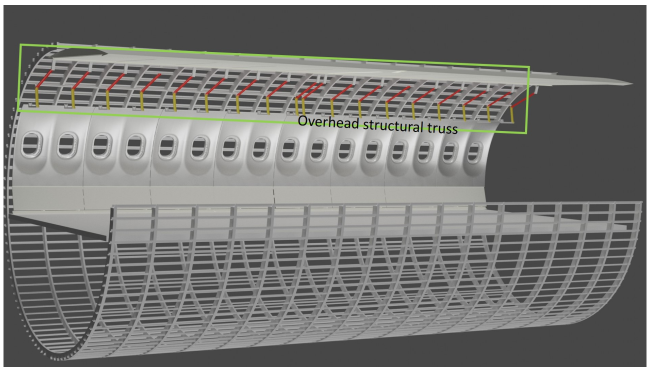

FIG 1. *Overhead structural truss* in a cabin environment

Newton-Euler algorithm is used for computing necessary parameters for energy consumption. Based on the computed energy consumption, joint configurations (from inverse kinematics) corresponding to lower energy usage is chosen.

From the existing researches it is seen that the robot performances is based of different factors such as *robot position w.r.to task and vice versa*, *hardware devices*, *type of trajectories* etc. The work presented in this paper tries to evaluate the effect of above mentioned factors and to present the result of analysis in MR. The idea of utilizing regression model is to reduce the interaction time in MR for any new target object in the environment.

3. TECHNICAL BACKGROUND

The use-case for digital twin implementation is the pre-assembly of *overhead structural truss*. This structural element supports the physical assembly segregation of the overhead cabin components from frames. It helps in modularizing the assembly process, where integration and disintegration of cabin overhead components from the frames are easier. Taking modularized cabin assembly into account, this element allows the possibility to perform some assembly operations outside cabin environment. The element is shown in FIG 1, where it can be seen that cabin components such as *ceiling panels*, *sidewall panels* are connected to the *Overhead structural truss*. This structural element is the main support to attach the overhead bins (also known as hat-racks) to the frames as shown in FIG 2. In the pre-assembly process the necessary components of the *overhead structural truss* are picked, placed and riveted (where-ever applicable) by the robots.

The whole execution is taking place in pre-assembly station as shown in FIG 3. There are two robots (*UR10e* and *Bosch-APAS*) placed on the linear axis on the pre-assembly cell. There are sensors installed in the pre-assembly cell, so that collaborative work between human and robots is possible. The virtual model of the pre-assembly cell is shown in FIG 4. This work on digital twin retrieves input from process planning, that delivers information on *component lo-*

[5]Time Optimal Smooth Trajectory Planning
[6]Time Optimal Trajectory Planning
[7]Degrees of Freedom

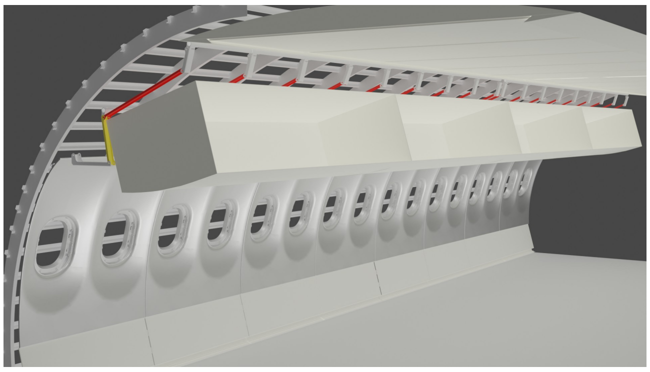
FIG 2. Overhead bins on *Overhead structural truss*

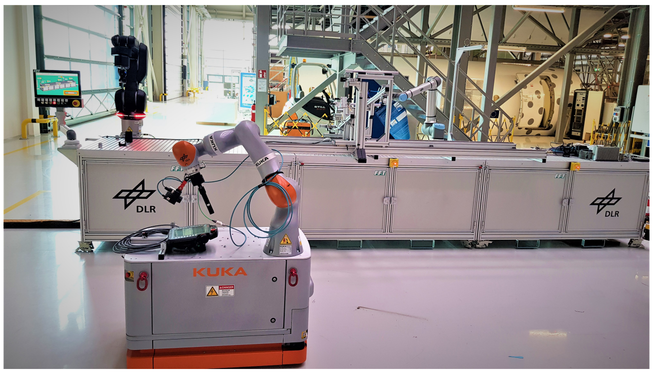
FIG 3. Pre assembly cell at DLR-SL Hamburg

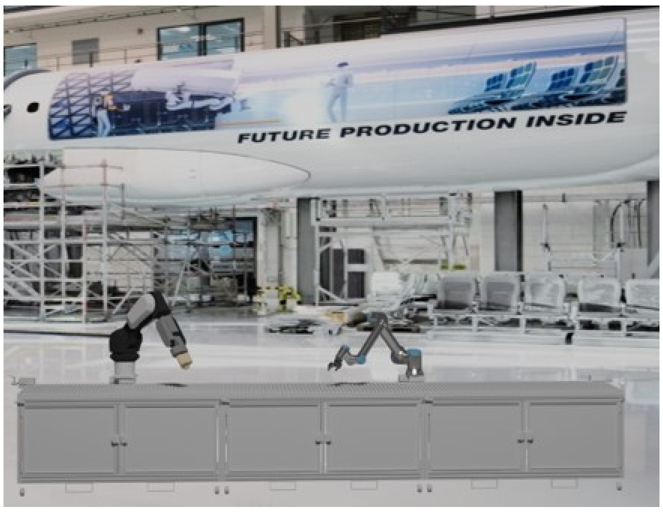

FIG 4. Virtual representation of the pre assembly cell

gistics and *components availability* along with target location (or the task location) on the pre-assembly cell. This information is provided by FUGA, (Fuselage Assembly Generator : a knowledge based fuselage design tool developed by DLR), which describes the target location in target frame. The information on sequence and target location is given as a .json file, and this looks as in Listing 1.
Listing 1 describes the different locations on the linear axis on the pre-assembly cell, where the robot perform the assembly operations. The field *ranking*, describes the order of sequence and *id* describes the corresponding process to be executed in the simulation. The fields *tcp_y*, *tcp_x*, *tcp_z* describes the location of robot on the linear axis. Since, the linear axis moves in **Y**-direction, only the values of *tcp_y*, changes in the provided json data. The location data is used as initial information to analyse the robot's reachability.

```
{"process1": {"tcp_y": 0.300, "tcp_x": 0.595,
    "tcp_z": 1.103, "robot_id": "APAS", "
    ranking": 1, "id": "subset_1"},
  "process2": {"tcp_y": 1.040, "tcp_x": 0.595,
     "tcp_z": 1.103, "robot_id": "APAS", "
    ranking": 2, "id": "subset_2"},
  "process3": {"tcp_y": 1.800, "tcp_x": 0.595,
     "tcp_z": 1.103, "robot_id": "APAS", "
    ranking": 3, "id": "subset_3"},
  "process4": {"tcp_y": 2.500, "tcp_x": 0.595,
     "tcp_z": 1.103, "robot_id": "APAS", "
    ranking": 4, "id": "subset_4"},
  "process5": {"tcp_y": 3.100, "tcp_x": 0.595,
     "tcp_z": 1.103, "robot_id": "UR10e", "
    ranking": 1, "id": "subset_5"},
  "process6": {"tcp_y": 3.100, "tcp_x": 0.595,
     "tcp_z": 1.103, "robot_id": "UR10e", "
    ranking": 2, "id": "subset_6"},
  "process7": {"tcp_y": 3.800, "tcp_x": 0.595,
     "tcp_z": 1.103, "robot_id": "UR10e", "
    ranking": 3, "id": "subset_7"},
  "process8": {"tcp_y": 4.700, "tcp_x": 0.595,
     "tcp_z": 1.103, "robot_id": "UR10e", "
    ranking": 4, "id": "subset_8"}
}
```

Listing 1. Sample JSON data for sequential execution

The information on *target location* is provided by FUGA, which generates models for different fidelity levels based on knowledge based approach [20]. From the assembly perspective, relevant information such as assembly locations of the robots are provided. This can be seen in Listing 2, where the attachment information is provided between the hat-racks and the frame. With this attachment information, it would be therefore possible to compute the robot orientation for optimal energy in assembly operations and to simulate the robot for assembly operations inside the cabin environment. The information on assembly locations on the pre-assembly table is also modeled similar to the Listing 2.

```
{"LuggageCompartmentsAttachment0000": {
    "LuggageCompartments0001": {"
    transformation": {"translation": { "x":
    -0.0, "y": 1.246, "z": 1.787}}},
    "C01": {"transformation": {"translation":
    { "x": -0.0, "y": 1.361, "z": 1.787}}}},
  "LuggageCompartmentsAttachment0001": {
    "LuggageCompartments0001": {"
    transformation": {"translation": { "x":
    0.533, "y": 1.246, "z": 1.787}}},
    "C02": {
      "transformation": {"translation": { "x":
     0.533, "y": 1.361, "z": 1.787}}}}}
```

Listing 2. Sample JSON data from FUGA regarding assembly information

Using the sequence information and the assembly location from process planning as inputs, this paper elucidates the methodology to determine the robot parameters for optimal time and energy task execution.

The presented work begins with, analysing the robot position on the linear axis followed by the robot task analysis. Total energy consumed is the sum of two components as shown:

$$\begin{aligned} Energy_{\text{Total}} &= Energy_{\text{LinearAxis}} + Energy_{\text{Robots}} \\ Time_{\text{Total}} &= Time_{\text{LinearAxis}} + Time_{\text{Robots}} \end{aligned} \tag{1}$$

4. METHODOLOGY

The proposed holistic approach involves describing the system architecture for virtual validation of cabin assembly process. In the vision of implementing FMS[8] for a cabin shop-floor, this work supports the implementation by developing a framework that is useful to analyse and understand robot behaviour for shop-floor tasks. The framework contains accurate digital twin (DT) model of the robot under consideration. The purpose of DT implementation is to understand the robot behaviour for real-time robot tasks in mixed reality(MR) environment.

4.1. Offline computation

Describing the digital twin is a two-step process, in which theoretical computations based on mathematical description is performed first, followed by generating a linear regression model. In the second stage, the computed data is validated against the real-time data and regression model improvisation is proposed. Initially, the kinematic analysis of the robot for different robot tasks are performed. The kinematic analysis is performed for different trajectories such as *splines*, *polynomial functions* etc. Based on the outcome of kinematic analysis, dynamic analysis follows to compute the time and energy consumed by the robot. This is performed by analysing the robot dynamic model using *Lagrangian* approach with inclusion on effects of external payload (crown module components). The outcome of the whole simulation is stored in MongoDB. Using the stored information, a linear regression model is trained. To ensure that the DT describes respective hardware accurately, the accuracy of linear regression model prediction is improved based on the real-time execution of robot tasks. The above elucidated methodology of DT is shown in FIG 5.
The detailed information on different computational modules are presented here:

Initial analysis on robot reachability

This work extensively uses klampt python library for robot analysis. As earlier mentioned, this paper describes the two-fold approach. At first, the inverse kinematics is performed at different locations on the linear axis. Based on the reachability analysis using inverse kinematics, the probable locations of the robot on the pre-assembly cell are determined. Out of the obtained probable locations, the path planning for the task is carried out. The path planning indicates the possibility of task execution based on the environmental constraints. This is shown in FIG 6.

[8]Flexible Manufacturing Systems

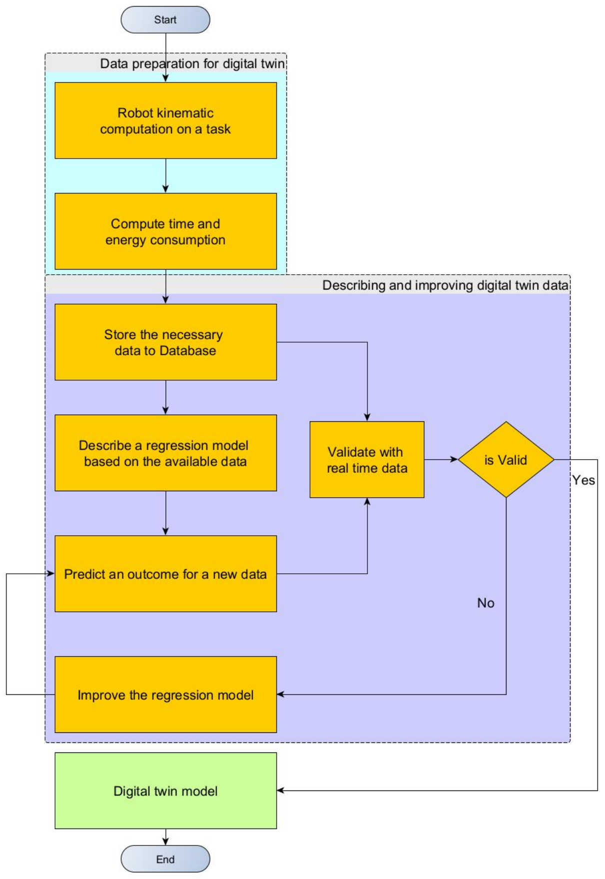
Start
Data preparation for digital twin
Robot kinematic computation on a task
Compute time and energy consumption
Describing and improving digital twin data
Store the necessary data to Database
Describe a regression model based on the available data
Validate with real time data
is Valid
Yes
No
Predict an outcome for a new data
Improve the regression model
Digital twin model
End

FIG 5. Digital twin model flow

Trajectory analysis

For each of the obtained locations on the pre-assembly cell from 4.1, the initial path planning is made for the robot task. The robot task in this work corresponds to place of crown module jigs components in the respective holding structure. The trajectory planning in this work performs the planning

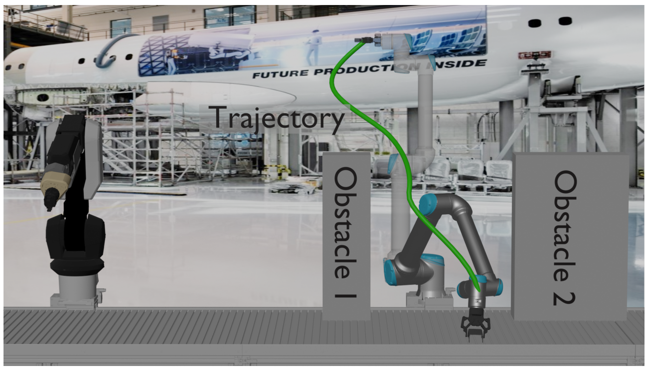

FIG 6. Initial linear trajectory based on reachability analysis

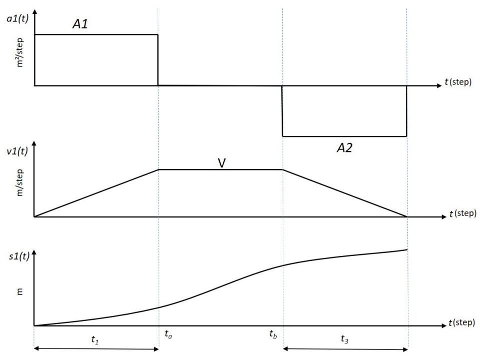

FIG 7. Trapezoidal trajectory of the robot

based on robot kinematics and not the dynamics. Due to this reason, the influence of external load isn't taken into consideration for trajectory planning. From the obtained path from motion planning algorithm, the trajectory planning is carried for different trajectory types (such as cosine, trapezoidal etc.). The time and energy required for each trajectory is then computed. Different trajectories that are analysed in this work is presented hereafter.

4.1.1. Trapezoidal trajectory

This trajectory profile is widely used in the industrial motor drives due to it's simplicity and the capability to achieve low task cycle time [21]. It comprises three motion steps: 1. *constant acceleration* 2.*constant velocity* and 3.*constant deceleration*. This profile isn't suited for high-speed operations, due to the nature of induced jerk [22] and jerk isn't well suited for minimal energy operations [23]. This work assumes that high speed operations are avoided due to the collaborative nature of the environment, so jerk is not going to be infinite. The trapezoidal trajectory is shown in FIG 7. By default *Klampt* library ramps up the velocity for 1/4th of the total execution time, stays constant for the 1/2th of the time and ramps down for the 1/4th of the time.

4.1.2. Triangular trajectory

Another form of trajectory that velocity ramped up for first 1/2 of the time and ramped down for another 1/2 of the time. This is not very much suitable for energy optimization, since the ramping up and ramping down may introduce jerk at certain velocities and the limiting velocity is identified based on loading conditions, such that jerk is minimum.

4.1.3. Cosine trajectory

In this work two forms of cosine trajectories for energy optimization is verified. By definition, cosine trajectory in *klampt* plans cosine trajectory based on the following equation:

(2) $$f(x) = \frac{1 - cos(x)}{2}$$

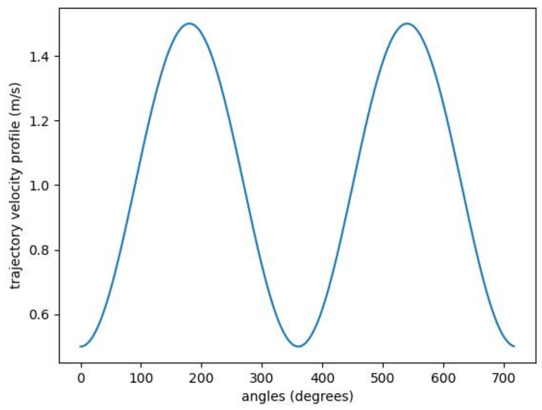

FIG 8. Theoritical cosine plot of robot joint velocity

The velocity profile of such trajectory is shown in FIG 8, where the velocity vector experiences sharp directional changes. With an attempt to minimize *jerk* this work proposes the other formulation as prescribed in [24], where jerk is minimized which is desired for optimal energy consumption. Due to this the following second order cosine trajectory is adopted [24]:

(3) $$x = acos((2\pi t)/T) + bt^2 + ct + d$$

(4) $$v = -a(2\pi/T)sin(2\pi t/T) + 2bt + c$$

(5) $$a = -a(2\pi/T)^2 cos(2\pi t/T) + 2b$$

(6) $$j^* = a(2\pi/T)^3 sin(2\pi t/T)$$

The different terms *a, b, c, d* are solved based on the boundary conditions for different parameters (displacement, velocity, acceleration and jerk) and they are described as follows:

(7) $$t = 0; \theta = 0 \qquad t = t_f; \theta = \theta_f$$

(8) $$t = 0; \dot{\theta} = 0 \qquad t = t_f; \theta = 0$$

(9) $$t = 0; \ddot{\theta} = 0 \qquad t = t_f; \ddot{\theta}_f = 0$$

(10) $$t = 0; \dddot{\theta} = 0 \qquad t = t_f; \dddot{\theta}_f = 0$$

These boundary conditions only describe the limits on *first* and *final* points of the trajectory. Intermediate boundary conditions are described accordingly (like continuous velocity, jerk etc.)

4.1.4. Parabolic trajectory

Parabolic trajectories are better in terms of smoothing trajectories. With a blend profile near to the junction of trajectories, it is possible to steer the robot at constant acceleration [25]. This is shown in FIG 9, where the trajectory waypoints in robot C-space is shown. Based on the blend duration, the blend profile is generated.

4.1.5. Minimum jerk trajectory

Smooth trajectories are normally modelled with cubic polynomials, but they provide constant jerk pro-

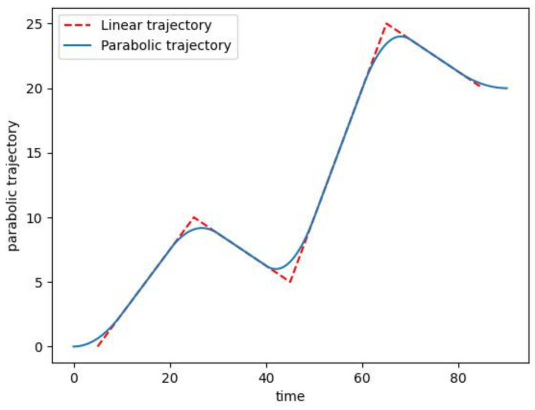

FIG 9. Trajectory with parabolic blend vs linear trajectory

file [26]. *klampt* uses quintic polynomial function for achieving minimum jerk trajectory.

Energy consumption analysis of linear axis

Based on the location of the linear axis from the previous position, the energy consumed by the linear motor can be computed. Out of the probable locations from initial inverse kinematics computation, the energy required by the linear axis motors is computed. The formulation is given here:

$$force = mass * acceleration \tag{11}$$

By computing the force by the known mass and acceleration due to gravity, torque is calculated as below:

$$torque = length\_of\_radial\_arm * force * sin(\theta) \tag{12}$$

From the known motor characteristics, required power is computed

$$power\_linear\_axis = torque * motor\_speed/9.5488 \tag{13}$$

Energy consumption analysis of robot

For each of the computed trajectory profile in section 4.1, the energy consumed by the robot is calculated based on robot dynamics. Dynamic model of the robot is described using the *Lagrangian* formulation. Based on the different influential factors such as *joint orientation*, *load*, *Coriolis and centrifugal forces*, the generated torque is computed based on the inverse-dynamics. From the computed torque, energy consumed by the robot is calculated.

$$B(q)\ddot{q} + C(q,\dot{q}) + G(q) = \tau + \sum_i J_i(q)^T f_i \tag{14}$$

q is the joint configuration
$\dot{q}$ is the velocity
$\ddot{q}$ is the acceleration
B(q) is the positive semidefinite mass matrix
C(q,$\dot{q}$) is the Coriolis force matrix
G(q) is the generalized gravity
τ is the link torque
f_i is the external forces

Based on the acceleration for any trajectory profile under consideration, the *inverse dynamics* for the robot joints are solved. The output is the *torque* required by the robot for a given configuration. The above equation holds good for robot with no load, and the digital twin implementation includes the external load characteristics. As proposed in [14] [27], external load is added to external forces term in equation(14). The force due to external load can be computed using Newton's law of motion: $F = m * a$. The mass of the different components of sub-element of *overhead structural truss* is shown in table 1.

Component	Mass
HE1	0.420kg
VIE1	0.290kg
HCE 1	0.37kg
HIE1	0.65kg
VIE2	0.170kg
HCE2	0.390kg

TAB 1. Mass of components belonging to sub-element

From the computed torque, the power consumed by the robot at any instant *t* can be calculated using [28]:

$$power\_robot(t) = \sum_{i=1}^{6} \tau_{est}(i,t)\dot{Q}(i,t) \tag{15}$$

$\dot{Q}$ is the angular velocity

4.2. Inferencing the parameters for robot task

Inferencing algorithm

Inferencing an unknown task is envisioned with the idea to reduce the computational time during the real time interaction. This can be inferred from the FIG 10. Time for computing path with obstacles takes almost 7minutes which may be too long for an user with

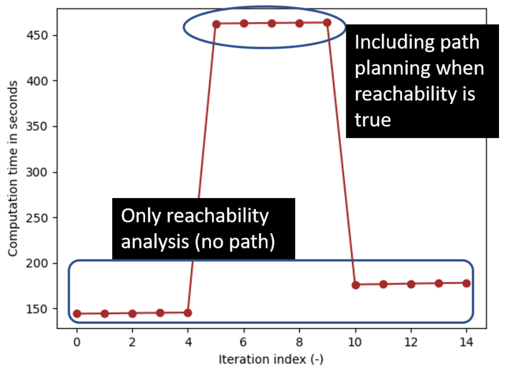

FIG 10. Time required for path and trajectory planning

MR interaction. For the real time interaction using mixed-reality, it would be more valuable if the results are delivered in shorter time, thereby accelerating the decision time. With a motivation to reduce the computation time, approach of regression analysis is tried out. Results of the regression model implementation is discussed in section6.
Correlating the data for meaningful inferencing is important to have a successful prediction on unknown information. *Feature selection* plays an important role in enhancing the performance of regression model. In this work, Pearson correlation is used for understanding the impact of *input* features to the dependent *output* variable. Based on the identified features, regression model is evaluated further.

4.3. Validating the regression model

There are standard measures to analyse the logistic regression model. They are:

Confusion matrix

It denotes the performance of a classification model. It comprises of 4 quadrants specifying 4 different combinations of predicted and actual values. The further validation specified in this section is based on the numbers that are obtained from confusion matrix and is shown in matrix-table 2.

		Actual values		
		Positive	Negative	Total
Predicted values	Positive	a	b	$a+b$
	Negative	c	d	$c+d$
	Total	$a+c$	$b+d$	N

TAB 2. Confusion matrix

Sensitivity (Se)

Sensitivity is the measure to ascertain the percentage of positive values that are classified as positive [29].

(16) $$Sensitivity = TP/(TP+FN)$$

Specificity (Sp)

Specificity complements sensitivity as it measures the percentage of negative values that are classified as negative. This is given by:

(17) $$Specificity = TN/(TN+FP)$$

Accuracy (A)

Accuracy is computed based on the area under the ROC (Receiver operating characteristic) curve. This curve shows the probability of correctly classified positive values against the probabilities of incorrectly classified negative values. Area under the ROC (*a.k.a* AUC) describes the capability of model to discriminate. Also area under ROC is used to trade-off between sensitivity and specificity. Mathematically, accuracy is given by:

(18) $$Accuracy = (TP+TN)/(TP+TN+FP+FN)$$

Precision (P)

Precision describes the measure that how good the model is predicting positive values that are actually positive. It is given by:

(19) $$Precision = TP/(TP+FP)$$

f1_score

f1_score is an alternative measure to describe models' accuracy. It is computed using the precision and sensitivity values. It is given by:

(20) $$f1\_score = (2 \times P \times Se)/(P+Se)$$

For the linear regression model, the following measures are used for understanding the performance of model:

Residual plots

Residual plots describe the difference between observed values and the learned response values.

[8]TP : True Positive, FN : False Negative
[8]2, TN : True Negative, FP : False Positive

Residuals are the difference between data values to the regression line. An regression (a.k.a identity) line is formed based on the predicted values from regression data and the scattering pattern around the identity line, describes if the linear regression model is valid.

Mean squared error (MSE)

It describes the closeness of regression line to the data points. Larger MSE means that the data points are widely dispersed and the smaller MSE describes the closer dispersion around the regression line. Smaller MSE values are preferred due to the fact that the errors are less and so better is the regression model.

R^2

Measures how well the data fits the regression model. Higher the R^2 value is desired but it doesn't mean that the regression model is good, and it is based on the regression data. Higher the R^2 values may also indicate over-fitting. In case of over-fitting the regression model performs well on test data and not on any unknown new data.

4.4. Real time interaction

With the implementation of DT based on analysis from 4.2, it would be possible to infer the robot behaviour for any new robot tasks in the MR environment. The following strategy is proposed for understanding robot behaviour interactively:

Identifying the objects with point cloud data

The objects in real-world is identified by performing point cloud analysis from 3D sensor data. The robot tasks are described based on combining the identified object and the user input from the MR. For examsple, point cloud data would identify a "side-wall panel" and if the user input would be "pick", the robot behaviour is analysed for "pick sidewall panel".

Perform robot task analysis

Regression analysis is performed on the new ("pick sidewall panel") robot task. This is a two-step procedure, where the reachability is first computed and based on the outcome, kinematic and dynamic data are computed. Time taken for the task along with energy required is computed.

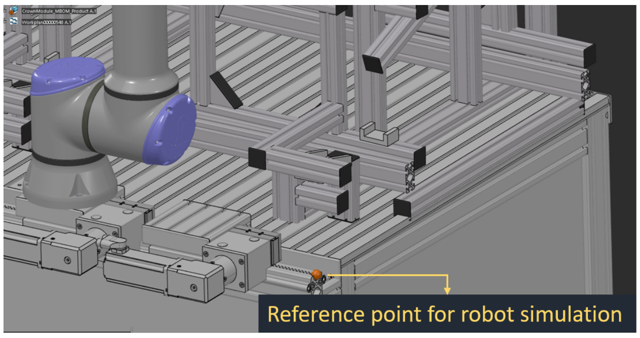

FIG 11. Reference point for computation and transformation

Virtual representation in MR

The computed parameters are displayed for the user in MR. The continuous interaction by the virtual system to the real world objects enables the possibility to correlate the effects of relative position and orientation of the object w.r.to the robot.
The whole implementation is planned to be deployed in an human-robot collaborative environment. The understanding of impact on relative position and orientation of object to robot could be useful to perform virtual commissioning, based on the multi-faceted analysis. An example of such analysis could be comparing the impact of robot execution with human execution on a specific task based on different parameters such as execution time, load etc. This understanding is then extended to understand the capabilities of PLC with other hardware on different communication parameters for an enhanced *virtual commissioning*.
Since real-time implementation is not part of this paper, any further information on real time interaction is not provided.

5. IMPLEMENTATION

The pre-assembly cell is modelled on 1:1 scale to the real time hardware. The placement of the linear axis on the pre-assembly table is taken into consideration for analysing the robot reachability for any required task. This is shown in FIG 11 and due to the reason that linear axis motion in real-time hardware is programmed with this point as reference, it is easier to perform transformation between simulation and real-world without any additional computation.
The *overhead structural truss* comprises of identical sub-elements that are spaced at regular intervals and are interconnected such that the load bearing capacity for holding the overhead bins is achieved. One such sub-element is shown in FIG 12. Following assumptions are applicable to this work:
1. Only the regular shapes are considered.
2. The robot handles component near to the it's centre of mass.
Each of the individual components of the sub-element are required to be placed on the jig for successful assembly process execution.

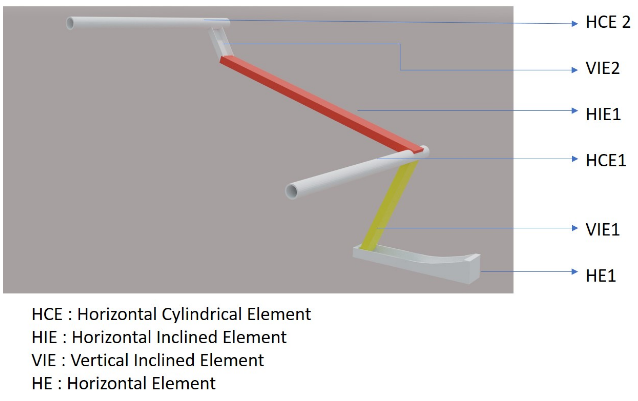
HCE 2
VIE2
HIE1
HCE1
VIE1
HE1
HCE : Horizontal Cylindrical Element
HIE : Horizontal Inclined Element
VIE : Vertical Inclined Element
HE : Horizontal Element

FIG 12. Representative diagram of a overhead structural truss element

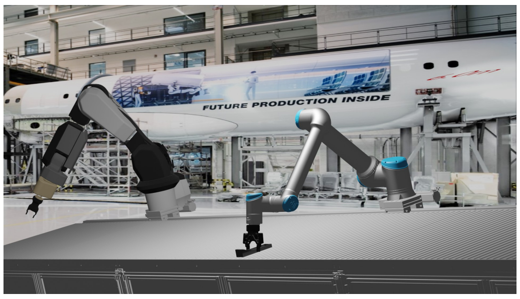

FIG 13. Initial reachability analysis

5.1. Initial analysis on robot reachability

Initial analysis on probable locations on the pre-assembly cell is computed for the different robot tasks that correspond to either pick or place on the pre-assembly cell. This is illustrated in FIG 13 [9], where the robot reachability for specific task is performed based on inverse kinematics analysis. There are more positions on the pre-assembly cell that are reachable for any robot task. This is shown in FIG 14, where the robot is placed at different locations on the pre-assembly cell and it can be seen that the robot task is successful.

The pre-assembly cell under consideration is 6m long. Performing the reachability analysis of the robot for whole length of the axis is not an appropriate execution, considering the "reach" of the robot arms. To ensure that the reachability analysis is carried appropriately, only the section of pre-assembly cell is considered relative to the target location in y-axis. This is shown in FIG 15, where the range of robot motion w.r.to target is defined. As shown in FIG 15, the range is described as ±1500mm (both directions on the pre-assembly cell) for UR10e robot, that has reachability of 1300mm.

[9] Image is shown without overhead structural truss assembly jig for easier understanding

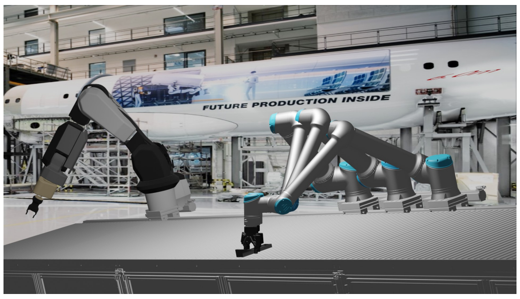

FIG 14. Multiple reachable positions on linear axis

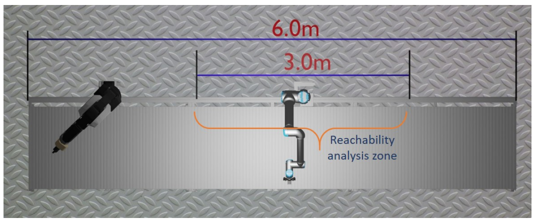

FIG 15. Range on linear axis for analysing reachability

Based on the reachability analysis, the locations on pre-assembly cell for a task is classified to be either *reachable* or *non-reachable*. This paves the further way to perform path planning which is described in the following sub-section.

5.2. Path planning and trajectory generation

For all the reachable locations, the path planning analysis is performed. The output from inverse kinematics is taken as input to the path planning module, and the path is planned for each joint of the robot. To maximize the benefit of motion planning in collision avoidance with objects around, robot configuration space is used.

In general path planning is described by following pipeline, [30]:

1. **Construction of the planning problem**: Configuration space (C-space) is described for better motion planning. In this work, C-space is described to avoid collisions with the pre-assembly cell and jigs for *overhead structural truss*.

2. **Instantiating the planning algorithm**: Using the constructed C-Space, it would be possible to plan the robot's motion. Different sampling based planning algorithms are used for planning the collision-free robot motion. In this work *Rapidly-exploring Random Trees (RRT)* algorithm is used to perform motion planning. The necessary values such as *planner type*, *perturbation radius* are set while instantiating the planner.

3. **Planner execution**: *RRT* algorithm works on random sampling in robot C-space [31]. The following steps are generalization of steps performed by *RRT* algorithm:

```
Until target_not_reached:
  1. set start_point, goal_point
  2. Generation of random point aka nodes.
  2. If the random point collides with
    obstacles, change the random point.
  3. If random point == goal_point:
      target_not_reached = False
```

Listing 3. Sequence of steps by RRT algorithm

The required start and goal positions are set followed by executing the planner in loop. The explored nodes are stored as *edges* by klampt library. One such explored edges are shown in FIG 16, where the list of joint configuration for first joint (a.k.a *"shoulder_pan_joint"*) is shown. The successful nodes that contribute to the final trajectory are stored as *milestones*. Since the algorithm works on random exploration in C-space, the loop execution is performed for certain number of iterations. More the number of iterations, more probability on successful path exploration, but also more computational time. In this work it is set to be 1000.

```
{
  "_id": {
    "$oid": "64bf7efd828ce5f0c0ef20e9"
  }
  "comp_name": "place_target_HFork_1",
  "target_position_x": 0.25,
  "target_position_y": 0.5,
  "target_position_z": 1.5,
  "traj_name": "trapezoidal",
  "isReachable": true,
  "robot_placement_location": 0.2,
  "req_angle_J1": 3.939444239980029,
  "req_angle_J2": 4.913539662982361 6,
  "req_angle_J3": -1.774851800771272 8,
  "req_angle_J4": -4.70941073198356 6,
  "req_angle_J5": -1.570852717506841 4,
  "req_angle_J6": -5.485367151729124 5,
  "power_consumed": 0.00007448406056987926,
  "time_consumed": 123.48270030090431,
  "date": {
    "$date": "2023-07-25T07:51:25.986Z"
  }
}
```

Listing 4. Data format stored in MongoDB

On the successful outcome of path planning, it is then possible to generate robot trajectories. The different velocity trajectories mentioned in 4.1 are generated, and the corresponding milestones and execution time-points are deduced for further analysis. A generated joint position and velocity trajectory profile for trapezoidal motion is shown in FIG 17 with time-points in x-axis and joint milestones in y-axis. More information about trajectory profile and their comparison on sustainable process execution is discussed in section6.
The whole flow from performing inverse kinematics calculations until energy and time computations are shown in FIG 18. The shown process flow depicts the implementation sequences for generating robot parameters concerning energy and time for different objects at different locations.

5.3. Regression model for computing required parameters

The procedure described in 5.1 and 5.2 are performed for different target locations, and the results are stored in MongoDB. The single block of stored information is shown in listing 4.
Based on the stored data, two different regression models are trained. First a logistic regression model is trained based on the *target positions* to the *reachability*. This is followed by training a linear regression model that operates on computing the required *joint angles*, *energy* and *time* consumed based on input *trajectory* (such as trapezoidal, triangular etc.) . Based on the regression model, it is then possible to infer the information on reachability and joint parameters for any new target.

5.3.1. Validating regression model

Features identification

Based on Pearson correlation, the features influences are studied. It is seen that the *Robot placement location* has larger influence on the reachability analysis, followed by *target y location*. This is shown in FIG 19. The similar analysis is made for linear regression model to compute joint angles for a robot task. The impact of *input features* on the joint 1 (*aka shoulder pan joint*) angle is shown in FIG 20. It can be seen that the target location in x-direction has greater influence on computing joint 1 angle and this analysis is made for all the other joints. It is seen that different features have different degree of influence on computing angles for different joints.
The regression model is built based on the identified features that have direct impact on the time and energy. A simple *multivariate multiple linear regression* model is described based on the identified features, that computes the time and energy consumption by the robot for any new task.

Validating logistic and linear regression

The different validating measures (like sensitivity, specificity) are computed based on the acquired data. More details on the data analysis for logistic regression validation is provided in next section 6.

6. RESULTS AND DISCUSSION

6.1. Analysing robot trajectories for sustainable execution

In the scope of optimal execution, it is necessary to compute the energy and time consumed by the whole system. As already established, both the linear axis and the robot contribute to the total energy and time. Time and energy consumed by the linear axis is directly computed from the formulas mentioned in 4.2 and it is influenced based on the target location. Energy consumed by the robot depends not only on the target location, but also on the chosen trajectory.
The influence of trajectories on time consumption can be interpreted in FIG 22, where the joint position and their velocities are plotted against time for different trajectory types. It can be seen that the *minimum-jerk* trajectory takes more time than the other trajectories while *trapezoidal* trajectory consume less time than others. The effect of time consumption vs en-

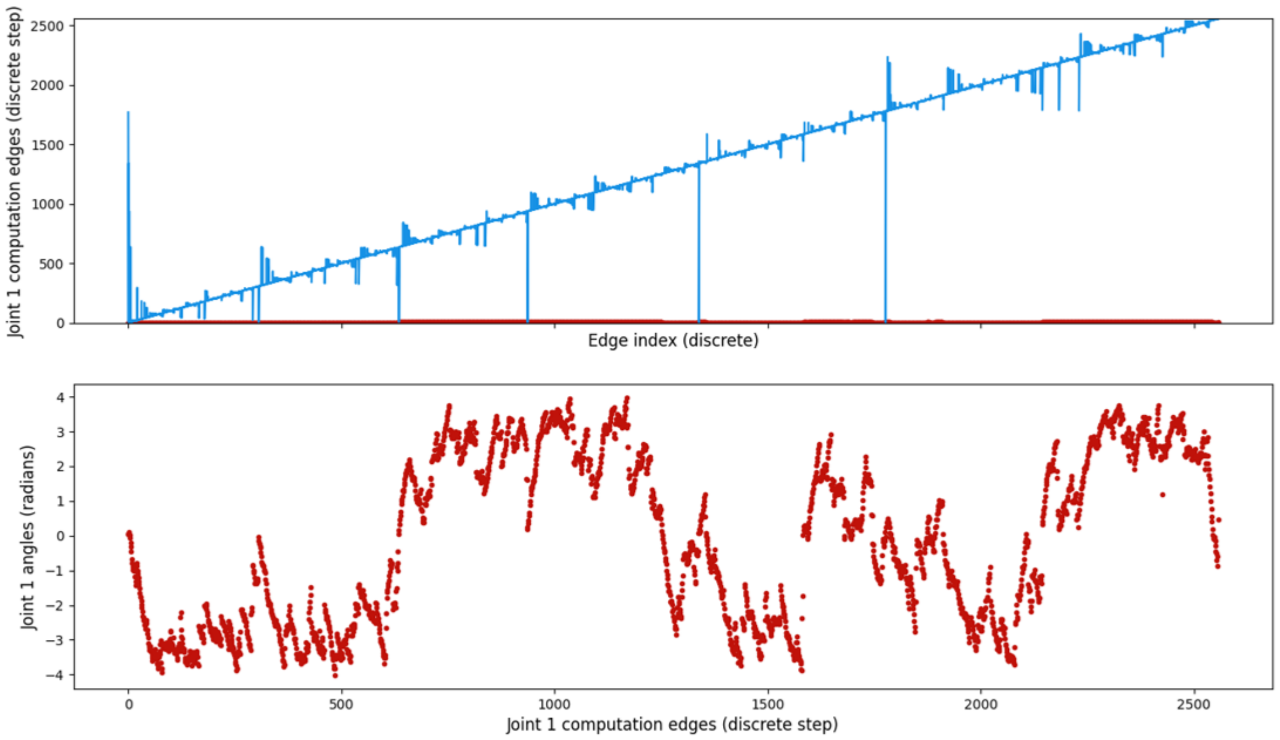

FIG 16. Computational edges and corresponding angles for first joint of UR10e

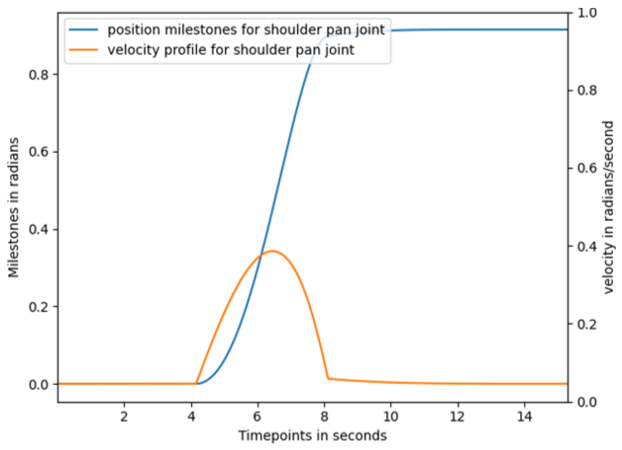

FIG 17. Joint position and velocity trajectories for trapezoidal profile

ergy could be clearly seen in FIG 21, where it can be understood that the *minimum-jerk* trajectory takes more time and consumes less energy and the *trapezoidal* trajectory is faster than other trajectories at different locations. The influence of robot's location on pre-assembly to time and energy consumption can be understood by interpreting the graph. Although *trapezoidal* trajectory is faster than other trajectories at robot base location 0.4m, it is not the same when the robot is placed at 0.3m and 0.5m. The consumed energy and required time for different trajectories is greatly influenced by base location of robot as it can be seen in FIG 21. It is evident from the values that was observed for different target at different locations. In general it is seen that Minimum jerk, triangular and trapezoidal trajectories consume less energy and take more time to perform. Since, FIG 21 describes the energy consumption only for the time period of trajectory, the observed values on energy consumption are too less. This information can be easily extrapolated depending on the hours of robot operations. For instance, if the robots are operated in shop-floor for *20hours/day*, it would be around 1000kWh. The cross-correlation of time and energy consumption by different trajectories to the base-location of robot lays the foundation for inferencing algorithm.

6.2. Correlating the regression model

The simulation was performed for different targets at different locations on the pre-assembly cell. A total of *243004* data were generated based on the structure described in listing4. Initially, logistic regression model is tested for reliable assertion. Part of the stored data are used for training and remaining for testing the model. *scikit* library is used for describing and testing the regression model. One way to understand the performance of logistic regression model is by plotting the confusion matrix as shown in FIG 23. Based on the correlation from different quadrants of confusion matrix, the effectiveness of regression model is identified. The simple logistic regression model didn't deliver good results for classifying reachability of robots on pre-assembly cell. Using the *train_test_split* in *scikit* library, 20% of the data were used for testing the model (48601 data out of 243004). The specificity of model with test dataset is really good at 0.9962 but the sensitivity of model isn't not good with a value of 0.1505. The different computational values are given here (based on FIG 23):

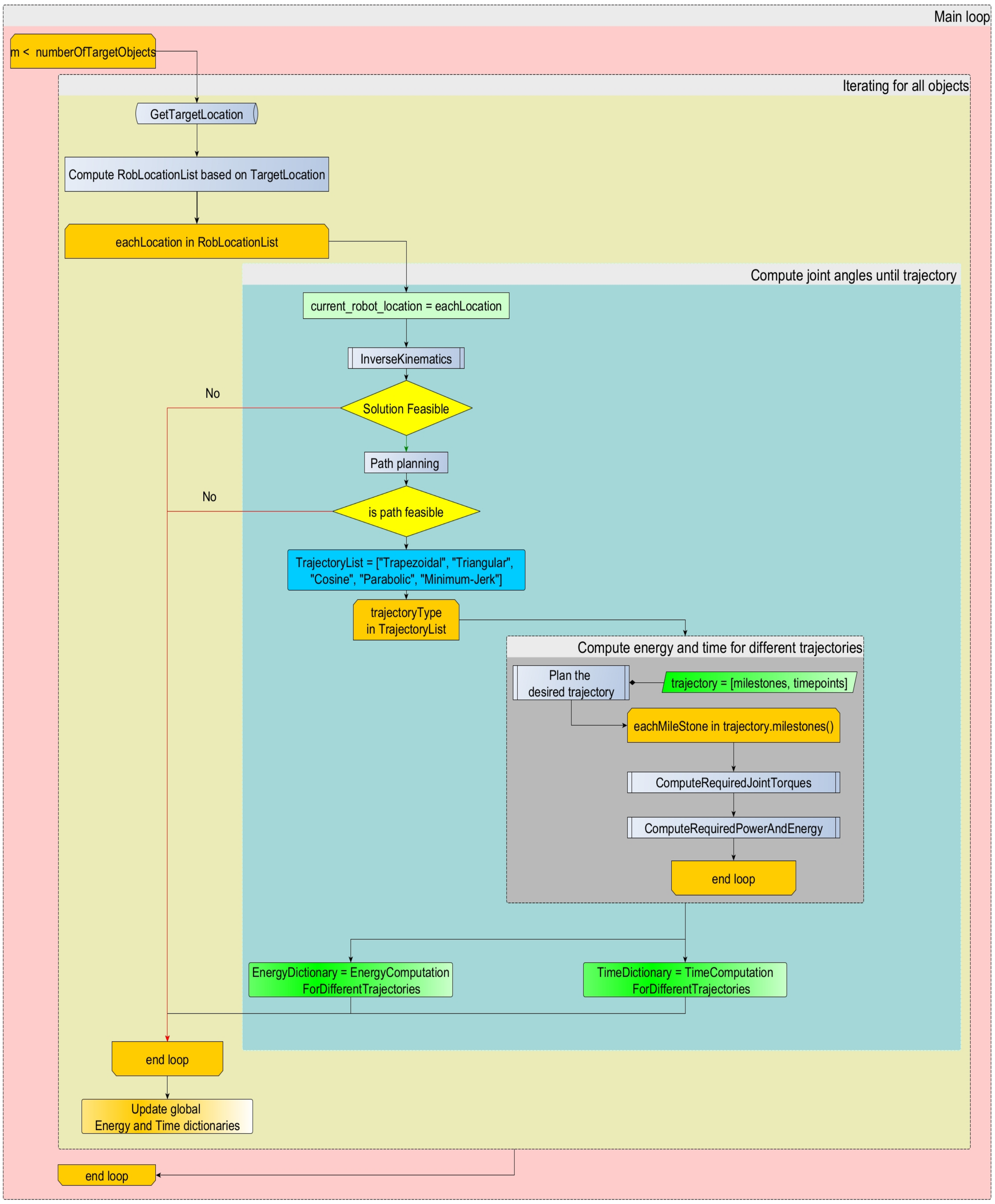
Main loop
m < numberOfTargetObjects
Iterating for all objects
GetTargetLocation
Compute RobLocationList based on TargetLocation
eachLocation in RobLocationList
Compute joint angles until trajectory
current_robot_location = eachLocation
InverseKinematics
No
Solution Feasible
Path planning
No
is path feasible
TrajectoryList = ["Trapezoidal", "Triangular", "Cosine", "Parabolic", "Minimum-Jerk"]
trajectoryType in TrajectoryList
Compute energy and time for different trajectories
Plan the desired trajectory
trajectory = [milestones, timepoints]
eachMileStone in trajectory.milestones()
ComputeRequiredJointTorques
ComputeRequiredPowerAndEnergy
end loop
EnergyDictionary = EnergyComputation ForDifferentTrajectories
TimeDictionary = TimeComputation ForDifferentTrajectories
end loop
Update global Energy and Time dictionaries
end loop

FIG 18. Process flow in reachability analysis and computation of KPI's

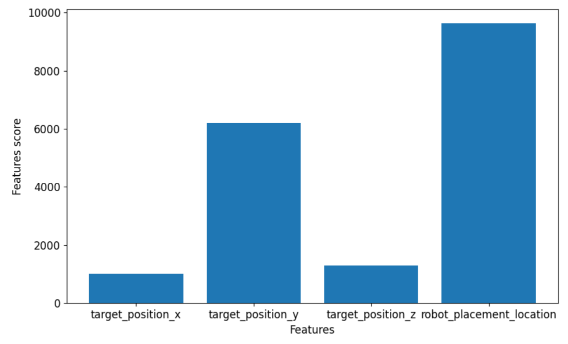

FIG 19. Feature selection analysis for predicting robot's reachability (Logistic Regression)

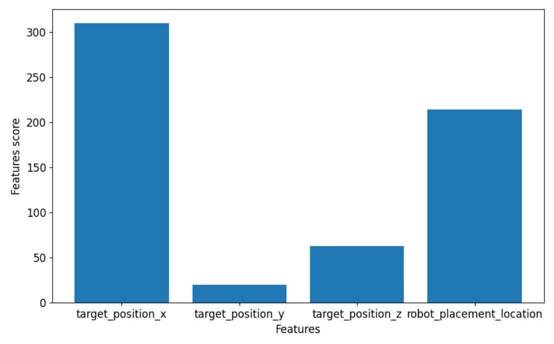

FIG 20. Feature selection analysis for predicting joint 1 angle (Linear Regression)

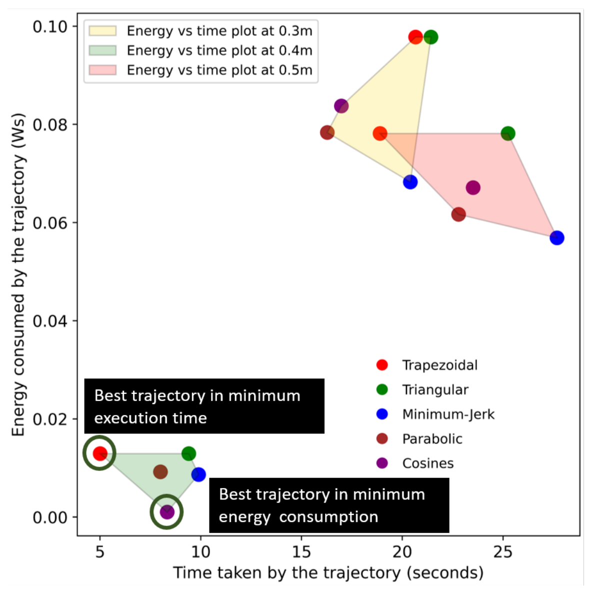

FIG 21. Time required and energy consumed by the different joint trajectories at different locations

Sensitivity : TP/(TP+FN)
151/(151 + 852) = 0.1505
Specificity : TN/(TN + FP)
47419/(47419 + 179) = 0.9962
Accuracy : (TP+TN)/(TP+TN+FP+FN)
(47419 + 151)/ (47419+151+179+852) = 0.9787
Precision : TP/(TP+FP)
151 / (151 + 179) = 0.4575
Due to underperforming simple logistic regression model, polynomial regression model is tried to improve the model classification behaviour. It improved to some extent as seen in FIG 24. Results are shown for third degree polynomial since it showed better classification behaviour compared to other degree polynomials. The different computational values are shown here (based on FIG 24):
Sensitivity : TP/(TP+FN)
463/(463 + 540) = 0.46
Specificity : TN/(TN + FP)
47339/(47339 + 254) = 0.9945
Accuracy : (TP+TN)/(TP+TN+FP+FN)
(47339 + 463)/ (47339 + 463 + 259 + 540) = 0.9835
Precision : TP/(TP+FP)
463 / (463 + 259) = 0.6412
Although the values are improved compared to simple logistic regression model, the model's sensitivity is still not sufficient for classification on unknown/new dataset.

7. CONCLUSION AND NEXT-STEPS

7.1. Conclusion

The presented work in this paper is part of the whole research on implementing the methodology to implement shop-floor digital twin model for virtual commissioning. The proposed implementation of digital twin begins with simulation model generation for robot shop-floor, and to optimize the simulation parameters based on the feedback information from hardware. With this accurately described simulation model, it would be then possible to perform virtual commissioning that would assist the users to understand the behaviour of robot performances for any task, which could be visualized through AR[10] model. With the assistance from AR, it would be possible to position the robot relatively to the task or vice-versa based on different KPIs[11] in real time environment.
This paper illustrates the methodology to model the robot behaviour for minimal time and energy consumption. The input for simulation is provided by process planning module, which delivers the target location for robot tasks. Using this input, a two-fold approach is proposed, where the probable locations of robot on pre-assembly cell are computed followed by trajectory computation. Performing dynamic analysis on the robot would provide the energy consumed by the robot for trajectory under consideration. Based

[10] Augmented Reality
[11] Key Performance Indicators

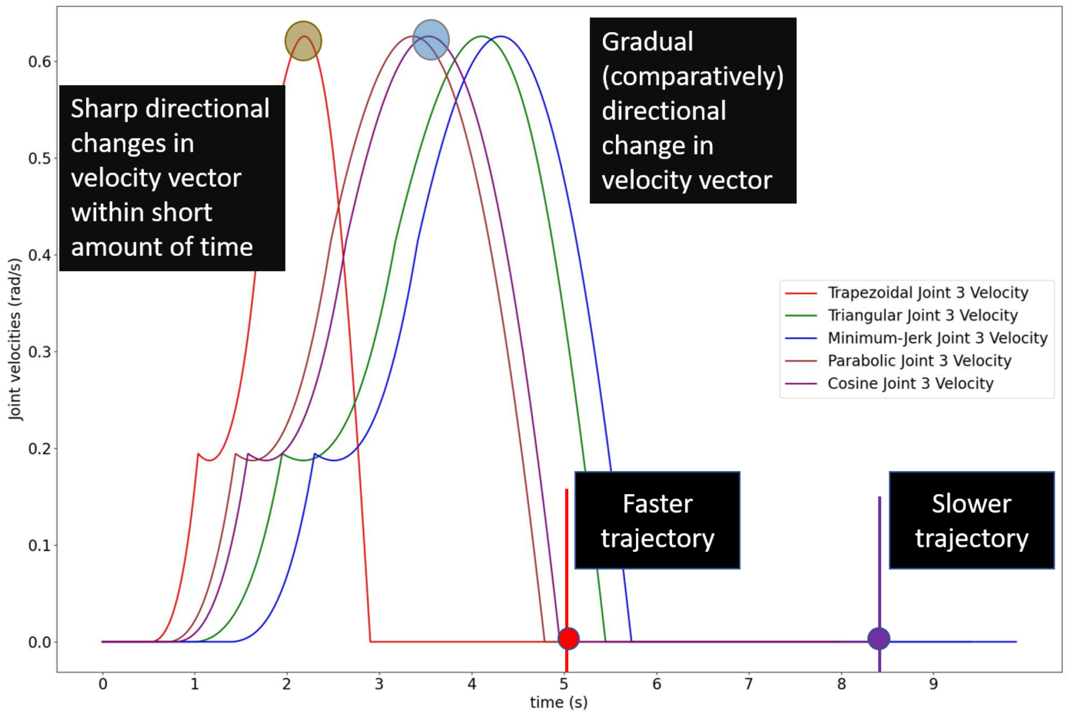

FIG 22. Joint position and velocity plot for different trajectories over time (robot at 0.1m on the pre-assembly cell)

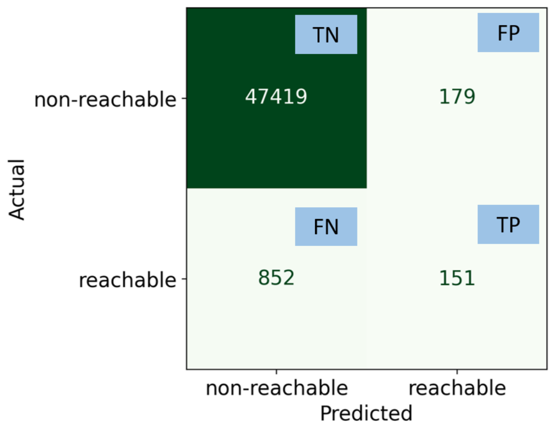

FIG 23. Confusion matrix for simple logistic regression

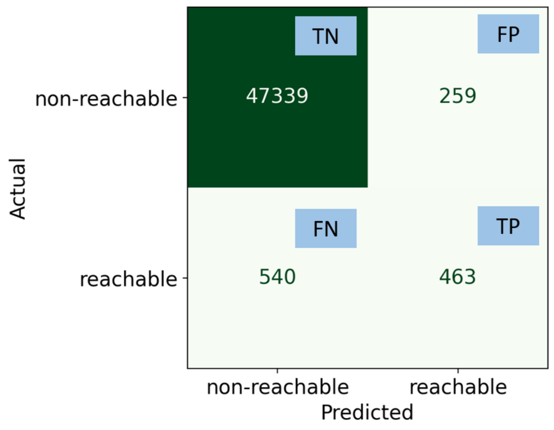

FIG 24. Confusion matrix for 3rd degree polynomial regression

on the total energy consumed by the robot for a given task, the robot location on pre-assembly cell corresponding to minimum energy is chosen. The robots are then simulated for visual inspection. Data is generated from simulation and is stored for further analysis. The idea to implement regression model arose from the time requirement for path planning algorithms, which may not be ideal for an mixed reality interaction. But the implementation of suitable regression algorithm has to be reinvestigated, since it didn't deliver expected results.On identifying the right regression model, the output of regression analysis could be integrated to process sequencing module or to layout planning, which could make informed decision based on robot capabilities.

7.2. Next steps

The implementation of inverse kinematics and path planning for different trajectories work fine. Analysis of different regression models would be carried in order to understand if it is an effective solution for the provided conditions. Different logistic regression models implementation (such as neural networks) will be investigated and the impact of such analysis will be performed before deployment. The trapezoidal profile has a ramp duration and the next step would be parametrizing the ramp duration to external load, such that any jerk due to the load in conjunction to motion is suppressed. For the parabolic trajectory it would be interesting to infer the relation between blend duration to the external load (which is in preliminary stage). Once the feedback data is available, the simulation has to be validated based on the real-time data, with which simulation parameters would be tuned. Usage of ROS[12] has been explored for digital shadow in cabin assembly process [32] and the idea on implementation has to be revisited, since the implementation time on acquiring ROS data would be considerably slower than using same interface as the feed-forward loop. Since data frequency is not playing central role in this work unlike remote monitoring, trade-off in interfaces for easier implementation would be adopted.

Contact:

venkatachalam.srinivasan@dlr.de

References

[1] Carlos Cesar Aparecido Eguti and Luís Gonzaga Trabasso. The virtual commissioning technology applied in the design process of a flexible automation system. *Journal of the Brazilian Society of Mechanical Sciences and Engineering*, 40(8), 2018. ISSN: 1678-5878.

[2] Philip Webb, Seemal Asif, Susanne Hogger, Thomas Kosche, and Paul Kiernan. Advanced flexible automation cell control for aerospace manufacturing. *Aircraft Engineering and Aerospace Technology*, 87(2):156–164, 2015. ISSN: 0002-2667.

[3] Thierry Berger, Dominique Deneux, Thérèse Bonte, Etienne Cocquebert, and Damien Trentesaux. Arezzo-flexible manufacturing system: A generic flexible manufacturing system shop floor emulator approach for high-level control virtual commissioning. *Concurrent Engineering*, 23(4):333–342, 2015. ISSN: 1063-293X.

[4] M. Dahl, K. Bengtsson, P. Bergagård, M. Fabian, and P. Falkman. Integrated virtual preparation and commissioning: supporting formal methods during automation systems development. *IFAC-PapersOnLine*, 49(12):1939–1944, 2016. ISSN: 24058963.

[5] Matthias Schamp, Thibaut Demasure, Stijn Huysentruyt, Jan Lamote, El-Houssaine Aghezzaf, and Johannes Cottyn. Multi-level approach to virtual commissioning: a reconfigurable assembly system case. *IFAC-PapersOnLine*, 55(10):3208–3213, 2022. ISSN: 24058963.

[6] Tobias Lechler, Eva Fischer, Maximilian Metzner, Andreas Mayr, and Jörg Franke. Virtual commissioning – scientific review and exploratory use cases in advanced production systems. *Procedia CIRP*, 81:1125–1130, 2019. ISSN: 2212-8271.

[7] Herman Vermaak and Johan Niemann. Virtual commissioning: A tool to ensure effective system integration. In *Proceedings of the 2017 IEEE International Workshop of Electronics, Control, Measurement, Signals and their Application to Mechatronics (ECMSM)*, pages 1–6, Piscataway, NJ, 2017. IEEE. ISBN: 978-1-5090-5582-1. DOI: 10.1109/ECMSM.2017.7945899.

[8] Mehmet Bayram Yildirim and Gilles Mouzon. Single-machine sustainable production planning to minimize total energy consumption and total completion time using a multiple objective genetic algorithm. *IEEE Transactions on Engineering Management*, 59(4):585–597, 2012. ISSN: 0018-9391.

[9] Giovanni Carabin, Erich Wehrle, and Renato Vidoni. A review on energy-saving optimization methods for robotic and automatic systems. *Robotics*, 6(4):39, 2017.

[10] M. Dahl, K. Bengtsson, M. Fabian, and P. Falkman. Automatic modeling and simulation of robot program behavior in integrated virtual preparation and commissioning. *Procedia Manufacturing*, 11:284–291, 2017. ISSN: 2351-9789.

[11] M. Ostanin, R. Yagfarov, D. Devitt, A. Akhmetzyanov, and A. Klimchik. Multi robots interactive control using mixed reality. *International Journal of Production Research*, 59(23):7126–7138, 2021. ISSN: 0020-7543.

[12] M. Ostanin and A. Klimchik. Interactive robot programing using mixed reality. *IFAC-PapersOnLine*, 51(22):50–55, 2018. ISSN: 24058963.

[13] Md Tanzil Shahria, Md Samiul Haque Sunny, Md Ishrak Islam Zarif, Md Mahafuzur Rahaman Khan, Preet Parag Modi, Sheikh Iqbal Ahamed, and Mohammad H. Rahman. A novel framework for mixed reality–based control of collaborative robot: Development study. *JMIR Biomedical Engineering*, 7(1):e36734, 2022.

[14] Tie Zhang, Meihui Zhang, and Yanbiao Zou. Time-optimal and smooth trajectory planning for robot manipulators. *International Journal of Control, Automation and Systems*, 19(1):521–531, 2021. ISSN: 1598-6446.

[15] Sameh Zribi, Marouene Mejerbi, Hatem Tlijani, Jilani Knani, and Vicenc Puig. Smooth motion profile for trajectory planning of a flexible manipulator. In Abdessattar Ben Amor, Naoufel Machta, and Salwa Elloumi, editors, *2018 International Conference on Advanced Systems and Electric Technologies (IC_ASET)*, pages 376–380, Piscataway, NJ, 2018. IEEE. ISBN: 978-1-5386-4449-2. DOI: 10.1109/ASET.2018.8379885.

[16] Lorenzo Scalera, Paolo Boscariol, Giovanni Carabin, Renato Vidoni, and Alessandro Gasparetto. Enhancing energy efficiency of a 4-dof

[12]Robot Operating System

parallel robot through task-related analysis. *Machines*, 8(1):10, 2020.

[17] Domenico Spensieri, Johan S. Carlson, Robert Bohlin, Jonas Kressin, and Jane Shi. Optimal robot placement for tasks execution. *Procedia CIRP*, 44:395–400, 2016. ISSN: 2212-8271.

[18] Ashok M. Sundaram, Nikola Budjakoski, Julian Klodmann, and Máximo A. Roa. Task-specific robot base pose optimization for robot-assisted surgeries. *Frontiers in robotics and AI*, 9:899646, 2022.

[19] Abdullah Mohammed, Bernard Schmidt, Lihui Wang, and Liang Gao. Minimizing energy consumption for robot arm movement. *Procedia CIRP*, 25:400–405, 2014. ISSN: 2212-8271.

[20] Jan-Niclas Walther, Christian Hesse, Jörn Biedermann, and Björn Nagel. Extensible aircraft fuselage model generation for a multidisciplinary, multi-fidelity context. 09 2022.

[21] Giovanni Carabin and Renato Vidoni. Energy-saving optimization method for point-to-point trajectories planned via standard primitives in 1-dof mechatronic systems. *The International Journal of Advanced Manufacturing Technology*, 116(1-2):331–344, 2021. ISSN: 0268-3768.

[22] H Z Li, Z M Gong, and W Lin. Motion profile planning for reduced jerk and vibration residuals.

[23] Yumeng Li, Zhengdao Wang, Hui Yang, Hua Zhang, and Yikun Wei. Energy-optimal planning of robot trajectory based on dynamics. *Arabian Journal for Science and Engineering*, 48(3):3523–3536, 2023. ISSN: 2193-567X.

[24] Zong Wu Xie, Cao Li, and Hong Liu. Cosine second order robot trajectory planning method. *Applied Mechanics and Materials*, 80-81:1075–1080, 2011.

[25] Tobias Kunz and Mike Stilman. Turning paths into trajectories using parabolic blends. 2011.

[26] Song Lu, Bingxiao Ding, and Yangmin Li. Minimum-jerk trajectory planning pertaining to a translational 3-degree-of-freedom parallel manipulator through piecewise quintic polynomials interpolation. *Advances in Mechanical Engineering*, 12(3):168781402091366, 2020. ISSN: 1687-8140.

[27] Wanjin Guo, Ruifeng Li, Chuqing Cao, Xunwei Tong, and Yunfeng Gao. A new methodology for solving trajectory planning and dynamic load-carrying capacity of a robot manipulator. *Mathematical Problems in Engineering*, 2016:1–28, 2016. ISSN: 1024-123X. DOI: 10.1155/2016/1302537.

[28] Koen Paes, Wim Dewulf, Karel Vander Elst, Karel Kellens, and Peter Slaets. Energy efficient trajectories for an industrial abb robot. *Procedia CIRP*, 15:105–110, 2014. ISSN: 2212-8271.

[29] Rosa Arboretti Giancristofaro and Luigi Salmaso. Model performance analysis and model validation in logistic regression. DOI: 10.6092/issn.1973-2201/358.

[30] Prof.Kris Hauser. Klampt python api documentation. http://motion.cs.illinois.edu/software/klampt/latest/pyklampt_docs/, 2023.

[31] Abdelfetah Hentout, Abderraouf Maoudj, Djelloul Yahiaoui, and Mustapha Aouache. Rrt-a*-bt approach for optimal collision-free path planning for mobile robots. *Algerian Journal of Signals and Systems*, 4(2):39–50, 2019. ISSN: 2543-3792. DOI: 10.51485/ajss.v4i2.81.

[32] Venkatachalam Srinivasan. Ros framework to generate digital shadow for automated cabin assembly process.