Lashchev, Andrei; Veselov, Gennady; Vyatkin, Valeriy

Towards Distributed Trajectory Interpolation and Motion Control

Published in:
Proceedings - 2021 26th IEEE International Conference on Emerging Technologies and Factory Automation, ETFA 2021

DOI:
10.1109/ETFA45728.2021.9613176

Published: 01/01/2021

Document Version
Peer-reviewed accepted author manuscript, also known as Final accepted manuscript or Post-print

Please cite the original version:
Lashchev, A., Veselov, G., & Vyatkin, V. (2021). Towards Distributed Trajectory Interpolation and Motion Control: Prototyping with ROS. In *Proceedings - 2021 26th IEEE International Conference on Emerging Technologies and Factory Automation, ETFA 2021* IEEE. https://doi.org/10.1109/ETFA45728.2021.9613176

Towards Distributed Trajectory Interpolation and Motion Control: Prototyping with ROS

Andrei Lashchev[1], Gennady Veselov[1], Valeriy Vyatkin[2,3]

[1]Institute of Computer Technology and Information Security, Southern Federal University, Taganrog, Russia
[2]Dept. of Computer Science, Electrical and Space Engineering, Luleå University of Technology, Luleå, Sweden
[3]Dept. of Electrical Engineering and Automation, Aalto University, Espoo, Finland
e-mails: lashhev-andrej@yandex.ru, gev@sfedu.ru, vyatkin@ieee.org

Abstract—**This article presents an approach for implementing distributed multi-agent control of a multi-axis manipulator using decentralised trajectory interpolation. The approach is prototyped using communication architecture of the Robot Operating System (ROS) framework. Also, using the proposed approach, the simulation was performed in the Copella Simulator. The solution aims at enabling multi-axis machines with intelligent axes having embedded controllers, capable of following trajectories without central control system.**

Keywords— distributed multi-agent motion control, decentralised trajectory interpolation, ROS.

I. Introduction

The plug-and-produce concept [1] has been inspiring the automation world as enabler of manufacturing flexibility. When applied to manufacturing machines, it means the ability to quickly modify the machine by substituting certain mechatronic components with functionally equivalent ones, or even build a machine from a variety of available components and then make the machine running with minimum effort spent on configuration and programming. This implies the mechatronic components to be intelligent enough to allow for such a seamless and effortless integration.

A typical example of a manufacturing machine, composed of intelligent mechatronic components, a multi-axis handling system of AFAG AG, is shown in Figure 1. Each of the axes is equipped with an embedded motion control system based on a microcontroller. However, it still requires a central control system to translate the trajectory of the required motion to the control tasks of each axis controller.

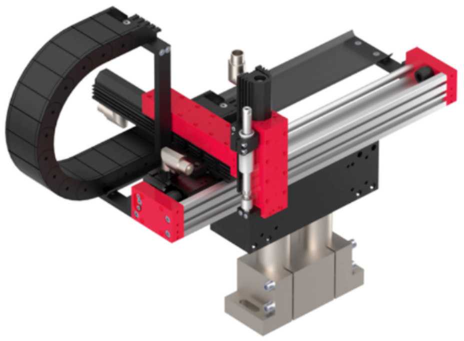

Figure 1. Multi-axis handling system of AFAG [3].

In the ongoing H2020 project 1-SWARM, the ambition is to achieve the equivalent behaviour of such machines without the central controller, applying the multi-agent control architecture. This would drastically increase the plug and produce capabilities of machines. Moreover, the solution is sought in terms of the distributed automation architecture of the IEC 61499 standard to enable portability and interoperability. This paper reports on the initial steps and results achieved. The paper is structured as follows. In Section II the problem is formulated. In Section III the related works are briefly observed. Section IV presents briefly the mathematical foundations of the forward and reverse kinematics problem. Section V reviews relevant services of the Robotic Operating System (ROS), which was used as a convenient prototyping framework. Section VI discusses the developed multi-agent distributed control method and Section VII presents simulation results. The paper is concluded with discussion of the results.

II. Problem Statement

The task of distributed multi-agent manipulator control consists in synthesizing motion control for each axis along the specified trajectory, that is defined by a set of points in spatial coordinates. The problem is divided to the following tasks:

1. Distribute control between the distributed control nodes to perform the trajectory task.
2. Develop a communication scheme between the node controllers.
3. Develop an algorithm for coordinated online adjustment of the local control actions.

III. Related Works

Multi-agent approach to the plug and produce implementation is proposed in [4].

A step towards decentralised motion control using the skills OPC-UA profile was investigated in [6], providing a standardised communication interoperability layer for plug and produce.

The work [7] proposes a kinematic model for a swarm of agents able to exhibit the formation of vortices around a given reference trajectory and to deal with uncertainty in the reference information.

The paper [8] presents a modular and distributable approach for kinematic and dynamic control of serial handling systems using IEC 61499 and exemplified with a Scara robot.

In [5] synchronization of distributed interpolation was proposed via cross-coupled control.

IV. Mathematical Foundations

A. Forward kinematics problem

The forward kinematics problem is formulated as a method, which uses the kinematic equations of a robot to compute the position of the end-effector from specified values for the joint parameters.[10]

Let's calculate transformation between a base coordinate system of a manipulator and coordinate system of the last link of a manipulator:

$$M_{n-1}^{n} = \begin{pmatrix} \cos\theta_n & -\sin\theta_n\cos\alpha_n & \sin\theta_n\sin\alpha_n & a_n\cos\theta_n \\ \sin\theta_n & \cos\theta_n\cos\alpha_n & -\cos\theta_n\sin\alpha_n & a_n\sin\theta_n \\ 0 & \sin\alpha_n & \cos\alpha_n & d_n \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$$r_0 = M_0^n r_g = M_0^1 M_0^1 \ldots M_{n-1}^n r_g = \begin{pmatrix} R_0^n & T_0^n \\ 0\quad 0\quad 0 & 1 \end{pmatrix} r_g \qquad (1)$$

where, R is 3×3 submatrix, which describes rotation movement, T is 3×1, submatrix, which describes translation movement, and N is number of joints [9][9]. M_{n-1}^n is the resulting transformation matrix from the n-link coordinate system to the manipulator base coordinate system. r_g is 4×1 vector, which represents the coordinates of the manipulator grip in the coordinate system of the *n*-th link.

$$p = \begin{pmatrix} T_0^g(1) & T_0^g(2) & T_0^g(3) & \varphi & \theta & \psi \end{pmatrix} \qquad (2)$$

where,

$$\varphi = \arctan\left(\frac{R_{2,3}}{R_{1,3}}\right), \ \theta = \arctan\left(\frac{R_{1,3}\cos\varphi + R_{2,3}\sin\varphi}{R_{3,3}}\right),$$

$$\psi = \arctan\left(\frac{R_{3,2}}{-R_{3,1}}\right),$$

and p is grip coordinates of the manipulator in the base coordinate system.

B. Inverse kinematics problem

The inverse kinematics problem (hereinafter referred to as IK) is the calculation of generalized coordinates for given linear and angular coordinates of the end effector of a manipulator. This problem is more complex than the direct kinematics problem, since it can lead to uncertainty of a solution (i.e., different configurations of a robot can correspond to the same position of end effector in space).

The inverse kinematics problem can be approximated using heuristic methods. These methods perform simple, iterative operations to gradually lead to an approximation of the solution. The heuristic algorithms have low computational cost (return the final pose very quickly), and usually support joint constraints. The most popular heuristic algorithms are: Cyclic Coordinate Descent (CCD) and Forward And Backward Reaching Inverse Kinematics (FABRIK) and Levenberg–Marquardt algorithm. [2][12]

The main task of this methods to solve the next equation:

$$\hat{\beta} \in \arg\min_{\beta} S(\beta) \equiv \arg\min_{\beta} \sum_{i=1}^{N} \left[y_i - f(x_i, \beta) \right]$$

As applied to the IK of a manipulator, instead of y_i, we substitute the coordinates of the target position of a gripper of the manipulator (1) into a basic coordinates system(BCS), and instead of $f(x_i, \beta)$ we can use expression (2).

V. ROS as a Framework for Prototyping

ROS provides standard operating system services, such as: hardware abstraction, low-level device control, implementation of frequently used functions, message passing between processes, and package management. ROS is based on a graph architecture, where data processing takes place in nodes that can exchange messages among themselves.

ROS has many user-supported packages (organized into sets called stacks) that implement various robotics functions: SLAM, scheduling, getting data from sensors, modelling, etc.

The most relevant ROS capabilities are:

1) It has a set of tools for dividing the code into different modules and managing their builds and dependencies for individual development.

2) This framework has a central repository of common parameters and files for all system components, available for all components (there is no need to implement a custom configuration file for parameters).

3) Communication between system agents (nodes) is carried out using a simple abstract solution. ROS has a built-in system of so-called topics that help to exchange data between nodes.

4) It provides a seamless communication between different machines connected to a common Ethernet network.[11]

All these characteristics of the ROS framework make it suitable for use as a communication framework in a distributed control architecture.

VI. Proposed Multi-Agent Control

Figure 2 shows architecture diagram of the proposed implementation of a distributed multi-agent control system.

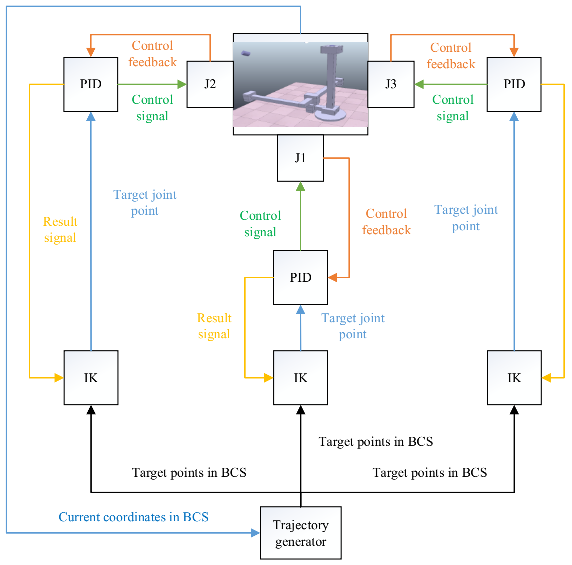

Figure 2. Functional diagram of the architecture of a distributed multi-agent control system.

All individual components of the system (nodes) exchange data using the ROS communication protocols. The manipulator has three links (J1-J3), each of which has an independent controller to control the drive.

Based on the current position of the manipulator grip, the trajectory generating unit sets a set of points in the base coordinate system and time stamps of arrival at each point of the trajectory, which forms the target trajectory of movement for the manipulator gripping. After receiving the "Ready" signal from the controllers, the trajectory generation unit sends this data to the inverse kinematics (IK) nodes of each link to recalculate the coordinates of the trajectory points from the base coordinate system to the generalized coordinate systems of each link (formula (4)). The algorithm is presented in flow-chart in Figure 3.

Then, after recalculating the coordinates, each controller polls the states of other controllers, and if all controllers send a "Ready signal" i.e. they are ready to send a command to their PID controllers. (PID)

This command is a ROS-action “FollowJointTrajectory“ [13] that contains target point, speed and acceleration data. This way the controllers are synchronized from one point to another.

Using functions from the API of the Coppella Simulator [14] simGetJointPosition(), simGetObjectFloatParameter (m_vrepJointsHandle, 2012, & vel), simGetJointForce(), PID regulators receive data on the current position, speed, moment of inertia of their link, respectively. And the drives are controlled are using the simSetJointPosition(), simSetJointTargetVelocity() and simSetJointForce() functions.

After the PID controllers bring their link to the target coordinate, a signal is sent back to the inverse kinematics node. Then the nodes of inverse kinematics are synchronized again and send the parameters of the next point of the trajectory to the regulators and the regulators again bring their link to the desired coordinate.

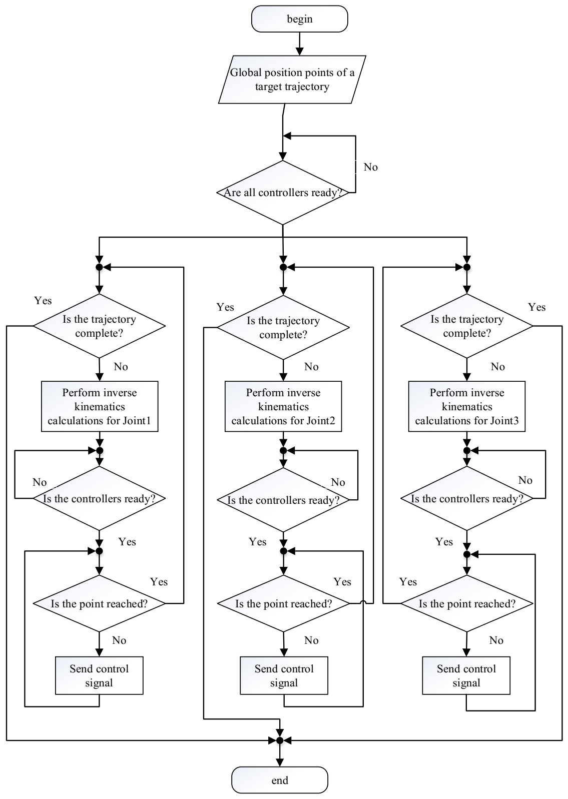
begin
Global position points of a target trajectory
Are all controllers ready?
No
Yes
Is the trajectory complete?
No
Perform inverse kinematics calculations for Joint1
Perform inverse kinematics calculations for Joint2
Perform inverse kinematics calculations for Joint3
Is the controllers ready?
Is the point reached?
Send control signal
end

Figure 3. Block diagram of the proposed algorithm for a distributed multi-agent control system.

VII. Simulation Results

According to distributed multi-agent manipulator control approach described in the section IV testing set was developed. All testing set components were run on the host computer as individual programs, using the communication architecture of a distributed multi-agent control system shown in Figure 2.

To test the proposed approach, the model of the three-link manipulator shown in Figure 4 was used.

To solve the direct and inverse problems of kinematics using the Denavit-Hartenberg method, a kinematic chain was created, as shown in Figure 5.

For each link of the manipulator model shown in Figure 4 the Denavit-Hartenberg parameters were calculated and presented in Table 1.

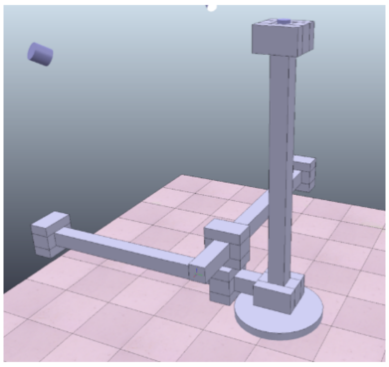

Figure 4. The model of a three-link manipulator in Coppella Sim.

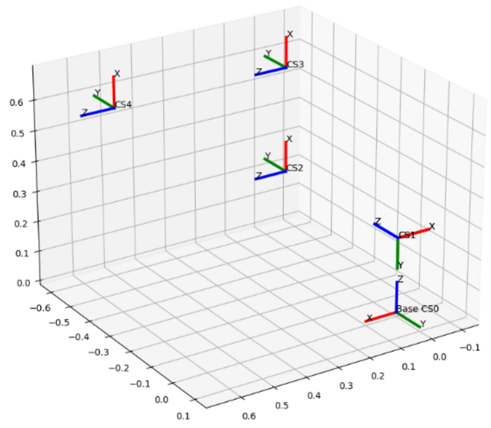

Figure 5. A kinematic chain for the manipulator model shown in Figure .

TABLE I. Denavit–Hartenberg parameters

Link number	Denavit–Hartenberg parameters			
	***d*, m**	***a*, m**	***θ*, rad**	***α*, rad**
1	$0.2469 + q_1$	0.0	π	$-\pi/2$
2	$0.5052 + q_2$	0.0	$-\pi/2$	$\pi/2$
3	q_3	0.35	0.0	0.0

Target and actual joint positions and the error between target and actual joint positions of the first link are presented in Figures 6 and 7 respectively. According to the results the error between target and actual joint positions less than 8 millimeters.

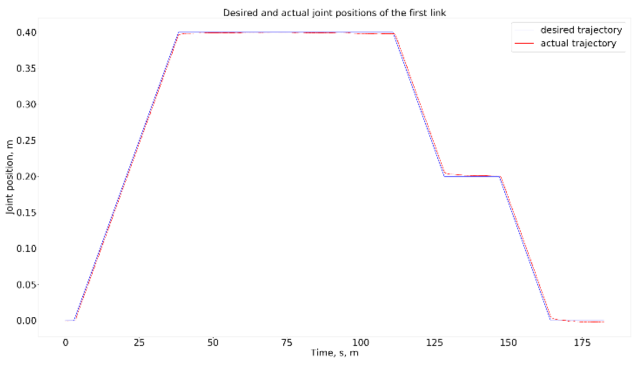

Figure 6. Target and actual joint positions of the first link.

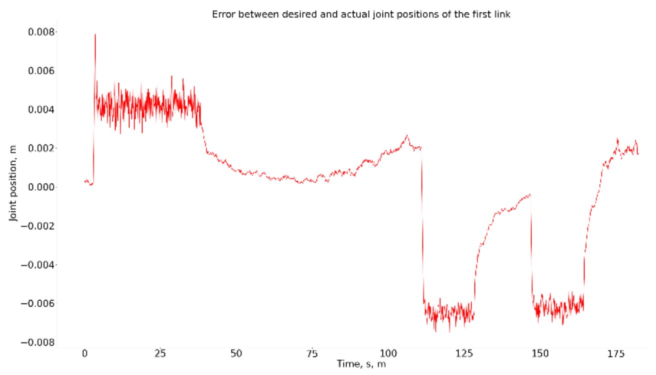

Figure 7. The error between target and actual joint positions of the first link.

For the second link, the error between target and actual joint positions less than 12 millimeters.

The error between target and actual joint positions of the third link are presented on Figure 8. According to the results the error between target and actual joint positions about 6 millimeters.

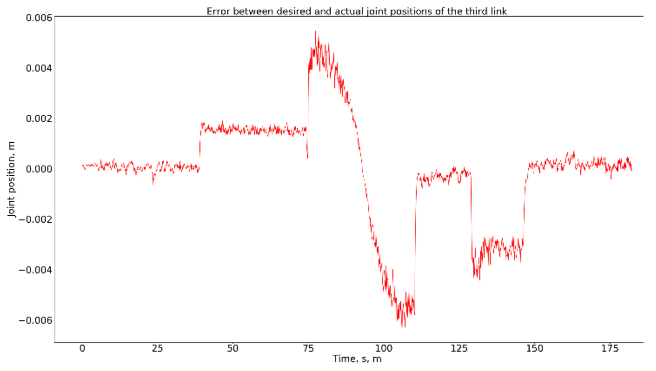

Figure 8. An error between target and actual joint positions of the third link.

The target trajectory of the manipulator's gripper and the trajectory obtained as the result of the simulation are shown in Figure 9.

According to the data obtained from the results of the simulation, the proposed approach to distributed multi-agent motion control of the manipulator provides high accuracy in performing the trajectory task.

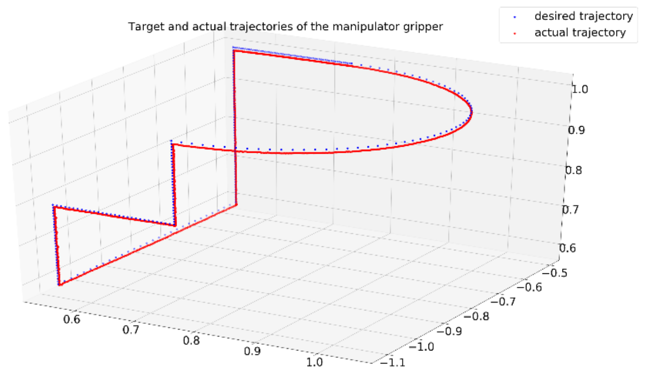

Figure 9.Target and actual trajectories of the manipulator gripper.

VIII. Conclusions

An approach to distributed multi-agent control with decentralised trajectory interpolation is presented and prototyped using ROS. The choice of the ROS framework was justified, an architecture diagram and an algorithm for implementing the problem of distributed multi-agent control were developed. Based on the proposed approach to the implementation of the distributed multi-agent control problem, simulation was carried out using the Coppella Simulator and the simulation results are presented.

Future work will include prototyping and testing of the solution using a distributed network of devices, migration to IEC 61499 architecture and optimization of performance.

IX. Acknowledgements

This work was supported, in part, by the H2020 project 1-SWARM co-funded by the European Commission (grant agreement: 871743).

References

[1] Arai, T., Aiyama, Y., Maeda, Y., Sugi, M. and Ota, J., 2000. Agile assembly system by “plug and produce”. *CIRP annals*, *49*(1), pp.1-4.

[2] Aristidou A., Lasenby J., Y. Chrysanthou, A. Shamir. Inverse Kinematics Techniques in Computer Graphics: A Survey. Computer Graphics Forum, 37(6): 35-58, 2018.

[3] Electric handling system EPS Mini XYZ, AFAG AG, Online: https://www.afag.com/en/products/detail/eps-mini-xyz-1.html

[4] Rocha, A.D., Peres, R.S., Flores, L. and Barata, J., 2015, December. A multiagent based knowledge extraction framework to support plug and produce capabilities in manufacturing monitoring systems. In *2015 10th International Symposium on Mechatronics and its Applications (ISMA)* (pp. 1-5). IEEE.

[5] Dripke, C., Ye, Z. and Verl, A., 2019, June. Synchronization of a distributed interpolation application via cross-coupled control. In *2019 IEEE International Conference on Engineering, Technology and Innovation (ICE/ITMC)* (pp. 1-8). IEEE.

[6] Beller, M., Schneider, B. and Zoitl, A., 2019, February. Skill-Based Motion Control with OPC UA and Deterministic Ethernet. In *International Conference on Computer Aided Systems Theory* (pp. 461-468). Springer, Cham.

[7] D’Alfonso, L., Bono, A. and Filice, A., 2020, September. A kinematic swarm model for vortex-like behavior around an uncertain target. In *2020 25th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)* (Vol. 1, pp. 435-440). IEEE.

[8] Steinegger, M., Plaschka, N., Melik-Merkumians, M. and Schitter, G., 2016, March. A framework for modular and distributable control of reconfigurable robotic systems. In 2016 IEEE International Conference on Industrial Technology (ICIT) (pp. 848-853). IEEE.

[9] Denavit J., Hartenberg R. S., 1955, A kinematic notation for lower-pair mechanisms based on matrices. Trans ASME J. Appl. Mech. **23**: pp. 215–221.

[10] Paul R., 1981. Robot manipulators: mathematics, programming, and control : the computer control of robot manipulators. MIT Press, Cambridge, Massachusetts, ISBN 978-0-262-16082-7.

[11] *Smart W. D., Gerkey B., Quigley M.*. Programming Robots with ROS. O'Reilly Media, Inc., 2015, pp. 3-5.

[12] McCarthy J. M., 1990, Introduction to Theoretical Kinematics, MIT Press, Cambridge, MA.

[13] ROS: control_msgs Msg/Srv Documentation. Available at: http://docs.ros.org/en/api/control_msgs/html/index-msg.html (accessed 6 June 2021).

[14] Coppella Sim User Manual . Available at: https://www.coppeliarobotics.com/helpFiles/index.html (accessed 6 June 2021).