## ICINCO 2020

17<sup>th</sup> International Conference on Informatics in Control, Automation and Robotics

## PROCEEDINGS

7 - 9 July, 2020

EDITORS

Oleg Gusikhin Kurosh Madani Janan Zaytoon

http://www.icinco.org

SPONSORED BY



SCITEPRESS

PAPERS AVAILABLE AT

# ICINCO 2020

Proceedings of the 17th International Conference on Informatics in Control, Automation and Robotics

### July 7 - 9, 2020

Sponsored by INSTICC - Institute for Systems and Technologies of Information, Control and Communication

> ACM In Cooperation ACM SIGAI - ACM Special Interest Group on Artificial Intelligence

In Cooperation with AAAI - Association for the Advancement of Artificial Intelligence euRobotics AISBL SPR - Portuguese Robotics Society INNS - International Neural Network Society BARA - British Automation and Robot Association

## Copyright © 2020 by SCITEPRESS – Science and Technology Publications, Lda. All rights reserved

Edited by Oleg Gusikhin, Kurosh Madani and Janan Zaytoon

Printed in Portugal ISSN: 2184-2809 ISBN: 978-989-758-442-8 Depósito Legal: 470573/20

http://www.icinco.org icinco.secretariat@insticc.org

## **CONTENTS**

#### **INVITED SPEAKERS**

Keynote Speakers			
From Robots Design to Navigation and Scene Understanding <i>Roland Siegwart</i>	5		
Cooperative MPC, Applications to Renewable Energy Systems Eduardo F. Camacho	7		
On the Regulation of Rivers Jean-Michel Coron	9		

#### INTELLIGENT CONTROL SYSTEMS AND OPTIMIZATION

#### FULL PAPERS

Convexification of Semi-activity Constraints Applied to Minimum-time Optimal Control for Vehicles with Semi-active Limited-slip Differential <i>Tadeas Sedlacek, Dirk Odenthal and Dirk Wollherr</i>	15
A Novel Model to Analyse the Effect of Deterioration on Machine Parts in the Line Throughput <i>E. Garcia, N. Montes, N. Rosillo, J. Llopis and A. Lacasa</i>	26
Using Semi-implicit Iterations in the Periodic QZ Algorithm Vasile Sima and Pascal Gahinet	35
Definition of a Walking with Starting and Stopping Motions for the Humanoid Romeo A. Kalouguine, V. de-León-Gómez, C. Chevallereau, S. Dalibard and Y. Aoustin	47
Implementation of Centralized MPC on the Quadruple-tank Process with Guaranteeing Stability       5         Roza Ranjbar, Lucien Etienne, Eric Duviella and José María Maestre       5	56
Control of Sewer Flow using a Buffer Tank K. M. Nielsen, T. S. Pedersen, C. Kallesøe, P. Andersen, L. S. Mestre and P. K. Murigesan	63
Towards Fully Automated Inspection of Large Components with UAVs: Offline Path Planning Constantin Wanninger, Raphael Katschinsky, Alwin Hoffmann, Martin Schörner and Wolfgang Reif	71
Adaptive Fault-Tolerant Control Allocation Schemes for Overactuated Systems with Actuator and Bias Faults Waseem Akram Francesco Tedesco and Alessandro Casavola	81
waseem Akram, Francesco reaesco ana Alessanaro Casavola	
CoachGAN: Fast Adversarial Transfer Learning between Differently Shaped Entities Mehdi Mounsif, Sébastien Lengagne, Benoit Thuilot and Lounis Adouane	89

#### SHORT PAPERS

Working Progress towards Lawn Mower Automation Alfredo Chávez Plascencia, Vítězslav Beran, Jaroslav Rozman and Aguilar Acosta Armando				
RBF Neural Network based Trajectory Control and Impedance Control of a Upper Limb Tele-rehabilitation Process <i>Ting Wang and Yanfeng Pu</i>	109			
Evaluation of Change Point Detection Algorithms for Application in Big Data Mini-term 4.0 <i>E. Garcia, N. Montes, J. Llopis and A. Lacasa</i>	117			
An Excited Binary Grey Wolf Optimizer for Feature Selection in Highly Dimensional Datasets Davies Segera, Mwangi Mbuthia and Abraham Nyete	125			
Optimal Reachability with Obstacle Avoidance for Hyper-redundant and Soft Manipulators Simone Cacace, Anna Chiara Lai and Paola Loreti	134			
Real-time Electrode Misalignment Detection Device for RSW Basing on Magnetic Fields D. Ibáñez, E. García, J. Martos and J. Soret	142			
Stability of Barrier Model Predictive Control Maxime Pouilly-Cathelain, Philippe Feyel, Gilles Duc and Guillaume Sandou	150			
Minimising the User's Effort during Wheelchair Propulsion using an Optimal Control Problem <i>Ouazna Oukacha, Chouki Sentouh and Philippe Pudlo</i>	159			
Ensuring Confidentiality of Information When Processing Operational Production Plans in Cloud Services Radda A. Iureva, Sergey V. Taranov, Alexander V. Penskoi and Artem S. Kremlev	167			
Efficient Construction of Neural Networks Lyapunov Functions with Domain of Attraction Maximization Benjamin Bocquillon, Philippe Feyel, Guillaume Sandou and Pedro Rodriguez-Ayerbe	174			
Damage Detection and Diagnosis for Offshore Wind Foundations Bryan Puruncajas, Yolanda Vidal and Christian Tutivén	181			
A New Neural Network Feature Importance Method: Application to Mobile Robots Controllers Gain Tuning Ashley Hill, Eric Lucet and Roland Lenain	188			
Real-time Prognosis of Failure of the IGBT in a Conversion Chain Kokou Langueh, Ghaleb Hoblos and Houcine Chafouk	195			
Detection and Estimation of Helicopters Vibrations by Adaptive Notch Filters Antoine Monneau, Nacer K. M'Sirdi, Sébastien Mavromatis, Guillaume Varra, Marc Salesse and Jean Sequeira	201			
Overview on Modeling for Control of Autonomous Road Vehicles Platoon Nacer K. M'Sirdi, Abdelhak Dahmani and Habib Nasser	208			

#### **ROBOTICS AND AUTOMATION**

#### FULL PAPERS

Learning to Close the Gap: Combining Task Frame Formalism and Reinforcement Learning for Compliant Vegetable Cutting Abhishek Padalkar, Matthias Nieuwenhuisen, Sven Schneider and Dirk Schulz	221
A Deep Learning Tool to Solve Localization in Mobile Autonomous Robotics Sergio Cebollada, Luis Payá, María Flores, Vicente Román, Adrián Peidró and Oscar Reinoso	232
A Bio-inspired Quasi-resonant Compliant Backbone for Low Power Consumption Quadrupedal Locomotion Edgar A. Parra Ricaurte, Julian D. Colorado, S. Dominguez and C. Rossi	242
Fuzzy Gradient Control of Electric Vehicles at Blended Braking with Volatile Driving Conditions Valery Vodovozov, Eduard Petlenkov, Andrei Aksjonov and Zoja Raud	250
PHRI Safety Control using a Virtual Flexible Joint Approach J. Diab, A. Fonte, G. Poisson and C. Novales	262
Vector based Control Routines for Swarms of Path Finding Robotic Devices Colin Chibaya	272
Global Estimation for the Convoy of Autonomous Vehicles using the Sliding-mode Approach <i>M-Mahmoud Mohamed-Ahmed, Aziz Naamane and Nacer K. M'sirdi</i>	284
Free-form Trap Design for Vibratory Feeders using a Genetic Algorithm and Dynamic Simulation Daniel Franzéen Haraldson, Lars Carøe Sørensen and Simon Mathiesen	294
Wilson Score Kernel Density Estimation for Bernoulli Trials Lars Carøe Sørensen, Simon Mathiesen, Dirk Kraft and Henrik Gordon Petersen	305
Sim-to-Real Transfer with Incremental Environment Complexity for Reinforcement Learning of Depth-based Robot Navigation <i>Thomas Chaffre, Julien Moras, Adrien Chan-Hon-Tong and Julien Marzat</i>	314
Solution of the Forward Kinematic Problem of 3UPS-PU Parallel Manipulators based on Constraint Curves Adrián Peidró, Luis Payá, Sergio Cebollada, Vicente Román and Óscar Reinoso	324
Interval-based Sound Source Mapping for Mobile Robots Axel Rauschenberger and Bernardo Wagner	335
Analyzing Decision Polygons of DNN-based Classification Methods Jongyoung Kim, Seongyoun Woo, Wonjun Lee, Donghwan Kim and Chulhee Lee	346
Model Predictive Control for Cooperative Insertion or Exit of a Vehicle in a Platoon Simone Graffione, Chiara Bersani, Roberto Sacile and Enrico Zero	352

#### SHORT PAPERS

Design and Experimental Study of a Pneumatic Bionic Stingray Undulatory Soft Robot Songzi Guo, Jinhua Zhang, Yuhan Yang, Haiyan Cheng and Jun Hong					
Analysis of Different Human Body Recognition Methods and Latency Determination for a Vision-based Human-robot Safety Framework According to ISO/TS 15066 <i>David Bricher and Andreas Müller</i>	369				
An Evaluation of New Global Appearance Descriptor Techniques for Visual Localization in Mobile Robots under Changing Lighting Conditions Vicente Román, Luis Payá, Sergio Cebollada, Adrián Peidró and Óscar Reinoso					
Mid-air Imaging for a Collaborative Spatial Augmented Reality System Dashlen Naidoo, Glen Bright and James Edward Thomas Collins	385				
Disturbance Compensator for a Very Flexible Parallel Lambda Robot in Trajectory Tracking Fatemeh Ansarieshlaghi and Peter Eberhard	394				
Stiffness Analysis of a New Tensegrity Mechanism based on Planar Dual-triangles Wanda Zhao, Anatol Pashkevich, Alexandr Klimchik and Damien Chablat	402				
Proactive-cooperative Navigation in Human-like Environment for Autonomous Robots Wanting Jin, Paolo Salaris and Philippe Martinet	412				
Design and Preliminary Evaluation of a Dextrous Encounter Type Force Feedback Interface Anthony Chabrier, Florian Gosselin and Wael Bachta	420				
RTFM: Towards Understanding Source Code using Natural Language Processing Maximilian Galanis, Vincent Dietrich, Bernd Kast and Michael Fiegert	430				
Centimeter-scaled Self-Assembly: A Preliminary Study Martin Jílek, Miroslav Kulich and Libor Přeučil	438				
Bridging the Reality Gap: Investigation of Deep Convolution Neural Networks Ability to Learn from a Combination of Real and Synthetic Data <i>Omar Gamal, Keshavraj Rameshbabu, Mohamed Imran and Hubert Roth</i>	446				
Design and Integration of a Dexterous Interface with Hybrid Haptic Feedback Florian Gosselin, Claude Andriot, François Keith, François Louveau, Guillaume Briantais and Pascal Chambaud	455				
Combination of Algorithms for Object Detection in Videos on Basis of Background Subtraction and Color Histograms: A Case Study <i>Theo Gabloffsky and Ralf Salomon</i>	464				
The Computation of the Inverse Kinematics of a 3 DOF Redundant Manipulator via an ANN Approach and a Virtual Function <i>Shahnaz Habibkhah and Rene V. Mayorga</i>	471				
Reduced Error Model for Learning-based Calibration of Serial Manipulators Nadia Schillreff and Frank Ortmeier	478				
PD Sliding Mode Controller based Decoupled Aerial Manipulation Kamel Bouzgou, Laredj Benchikh, Lydie Nouveliere, Yasmina Bestaoui and Zoubir Ahmed-Foitih	484				

Deep Learning Algorithm for Object Detection with Depth Measurement in Precision Agriculture Aguirre Santiago, Leonardo Solaque and Alexandra Velasco	490
Study of a Hybrid Actuated Exoskeleton for Upper Limb Rehabilitation Dimitar Chakarov, Ivanka Veneva, Mihail Tsveov and Pavel Venev	498
Functional Architecture using ROS for Autonomous UAVs Johvany Gustave, Jamy Chahal and Assia Belbachir	506
An Integrated Object Detection and Tracking Framework for Mobile Robots William Kristian Juel, Frederik Haarslev, Norbert Krüger and Leon Bodenhagen	513
On the Application of Safe-Interval Path Planning to a Variant of the Pickup and Delivery Problem Konstantin Yakovlev, Anton Andreychuk, Tomáš Rybecký and Miroslav Kulich	521
Survey and Preliminary Results on the Design of a Visual Light Communication System for Radioactive and Underwater Scenarios Giovanni Napoli, José Vicente Martí Avilés, Raúl Marín Prades and Pedro J. Sanz Valero	529
Backstepping Controller Applied to a Foldable Quadrotor for 3D Trajectory Tracking Saddam Hocine Derrouaoui, Yasser Bouzid, Mohamed Guiatni, Halfaoui Kada, Islam Dib and Noureddine Moudjari	537
Reliable Modeling for Safe Navigation of Intelligent Vehicles: Analysis of First and Second Order Set-membership TTC Nadhir Mansour Ben Lakhal, Othman Nasri, Lounis Adouane and Jaleleddine Ben Hadj Slama	545
Deep Learning with Transfer Learning Method for Error Compensation of Cable-driven Robot Aydar Akhmetzyanov, Maksim Rassabin, Alexander Maloletov, Mikhail Fadeev and Alexandr Klimchik	553
SIGNAL PROCESSING, SENSORS, SYSTEMS MODELLING AND CONTROL	
FULL PAPERS	
En dienst Medel haard Daarman Managements An Application to the P1 of MULTI THE L	

Functional Model-based Resource Management: An Application to the Electric Vehicle Thermal Control Baptiste Boyer, Philippe Fiani, Guillaume Sandou, Emmanuel Godoy and Cristina Vlad	565			
Backlash Identification in Industrial Positioning Systems Aided by a Mobile Accelerometer Board with Wi-Fi Mathias Tantau, Lars Perner, Mark Wielitzka and Tobias Ortmaier				
Current Loop Stability Analysis of a VIENNA-type Three-phase Rectifier for an Imaging System Power Supply Application	585			

Power Supply Application Matthieu Darnet, Emmanuel Godoy, Daniel Sadarnac and Stephane Gautrais

Scalable Electric-motor-in-the-Loop Testing for Vehicle Powertrains Thomas D'hondt, Yves Mollet, Arthur Jacques Joos, Leonardo Cecconi, Mathieu Sarrazin and 594 Johan Gyselinck

#### SHORT PAPERS

Deep Learning for Posture Control Nonlinear Model System and Noise Identification Vittorio Lippi, Thomas Mergner and Christoph Maurer				
Development of Portable Sound Source Direction Estimation Device Kenta Goto, Hiroki Kijima, Mizuki Usuda and Yoshihisa Uchida	615			
Gait-based Person Identification using Multiple Inertial Sensors Osama Adel, Yousef Nafea, Ahmed Hesham and Walid Gomaa	621			
Multi-sensor Gait Analysis for Gender Recognition Abeer Mostafa, Toka Ossama Barghash, Asmaa Al-Sayed Assaf and Walid Gomaa	629			
Improving Activity Mining in a Smart Home using Uncertain and Temporal Databases Josky Aïzan, Cina Motamed and Eugene C. Ezin	637			
Robustness Estimation of Large Deviations in Linear Discrete-time Systems with Control Signal Delay <i>Nina Vunder and Natalia Dudarenko</i>	643			
Simulative Investigation of Transfer Function-based Disturbance Observer for Disturbance Estimation on Electromechanical Axes <i>Chris Schöberlein, Armin Schleinitz, Holger Schlegel and Matthias Putz</i>	651			
Reduced-order Modeling of Parameter Variations for Parameter Identification in Rubber Curing Tobias Frank, Mark Wielitzka, Matthias Dagen and Tobias Ortmaier	659			
Performance Analysis of the Force Control for an Electromechanical Feed Axis with Industrial Motion Control Andre Sewohl, Manuel Norberger, Chris Schöberlein, Holger Schlegel and Matthias Putz	667			
Real-time Implementation and Evaluation of Magnetometerless Tracking System for Human and Humanoid Posture Control Benchmarking based on Inertial Sensors <i>Vittorio Lippi, Kai Günter Brands and Thomas Seel</i>	675			
Ammonium Sensor Fault Detection in Wastewater Treatment Plants David Tena, Ignacio Peñarrocha-Alós, Roberto Sanchis and Rubén Moliner-Heredia	681			
Parameter Estimator for Twin Rotor MIMO System based on DREM Procedure Nikita Shopa, Dmitry Bazylev, Sergey Vrazhevsky and Artem Kremlev	689			
Hand Detection Algorithm: Pre-processing Stage Raissa Likhonina	695			
Improved IMU-based Human Activity Recognition using Hierarchical HMM Dissimilarity Sara Ashry, Walid Gomaa, Mubarak G. Abdu-Aguye and Nahla El-borae	702			
Future Parking Applications: Wireless Sensor Network Positioning for Highly Automated in-House Parking Andrea Jung, Paul Schwarzbach and Oliver Michler	710			
Design and Application of a Reconfigurable Control to a Cyber-Physical System Imane Tahiri, Alexandre Parant, François Gellot, Alexandre Philippot and Véronique Carré-Ménétrier	718			
Lyapunov Stability of a Nonlinear Bio-inspired System for the Control of Humanoid Balance Vittorio Lippi and Fabio Molinari	726			

Evaluation of Lyapunov Function Candidates through Averaging Iterations Carlos Argáez, Peter Giesl and Sigurdur Hafstein	734
CPA Lyapunov Functions: Switched Systems vs. Differential Inclusions Sigurdur Hafstein	745
INDUSTRIAL INFORMATICS	
FULL PAPER	
Domain Optimization for Hierarchical Planning based on Set-Theory Bernd Kast, Vincent Dietrich, Sebastian Albrecht, Wendelin Feiten and Jianwei Zhang	759
SHORT PAPERS	
A Holistic Approach for the Development of a Digital Twin Focused on Commissioning and Control of Electromechanical Feed Axes Manuel Norberger, René Apitzsch, André Sewohl, Holger Schlegel and Matthias Putz	769
The Pipeline Concept as Key Ingredient for Modular, Adaptive Communication for Cyber-physical Systems Stefan Linecker, Jia Lei Du, Anderson Valões Silva and Reinhard Mayr	775
AUTHOR INDEX	781

#### Solution of the Forward Kinematic Problem of 3UPS-PU Parallel Manipulators based on Constraint Curves

Adrián Peidró<sup>®a</sup>, Luis Payá<sup>®b</sup>, Sergio Cebollada<sup>®c</sup>, Vicente Román<sup>®d</sup> and Óscar Reinoso<sup>®e</sup>

Automation, Robotics and Computer Vision Laboratory, Miguel Hernández University, Avda. de la Universidad s/n, 03202 Elche, Spain {apeidro, Ipaya, sergio.cebollada, v.roman, o.reinoso}@umh.es

Keywords: Forward Kinematics, k-d Trees, Parallel Robots, Algebraic Elimination, Curve Rendering, Angular Wrapping.

Abstract: Algebraic elimination methods for solving the forward kinematic problem of parallel manipulators are fast and obtain all solutions, but they require eliminating all unknowns except one, and solving a high-degree univariate polynomial whose coefficients often have expressions too complex to be obtained symbolically. This prevents parameterizing these coefficients in terms of all the kinematic parameters involved, which requires repeating the elimination process again whenever these kinematic parameters change. To avoid this, this paper presents an new method to solve the forward kinematics of 3UPS-PU parallel manipulators by eliminating only one unknown, reducing the system to an easily parameterizable set of planar constraint curves in the space of the remaining unknowns, which contain all real solutions of the forward kinematics. By sampling points from these curves densely, and sorting the sampled points using k-d trees, the proposed method manages to search all real solutions along these curves. The proposed method is compared to previous methods that obtain all solutions and is shown to perform about 100 times faster than these methods, but is less general than them.

#### **1 INTRODUCTION**

This paper presents a method for quickly obtaining all real solutions of the forward kinematic problem of 3UPS-PU parallel manipulators. This manipulator, which has been studied by several authors, is part of the well-known Tricept machining robot (Neumann, 1988) which has been widespread in industry.

Siciliano (1999) analyzed the inverse kinematics and manipulability of the Tricept robot. Fattah and Kasaei (2000) modelled the dynamics of 3UPS-PU parallel robots using Newton-Euler equations and the natural orthogonal complement method. Caccavale et al. (2003) also modeled the dynamics of the Tricept robot, and designed an impedance controller for interacting with the environment.

Kolláth et al. (2009) studied the workspace and positioning accuracy of this robot. Hosseini and Daniali (2011) studied the proximity to singularities of the Tricept manipulator, and optimized its design to maximize workspace volume. Hu (2014) analyzed

- <sup>a</sup> https://orcid.org/0000-0002-4565-496X
- <sup>b</sup> https://orcid.org/0000-0002-3045-4316
- <sup>c</sup> https://orcid.org/0000-0003-4047-3841
- d https://orcid.org/0000-0002-3706-8725
- e https://orcid.org/0000-0002-1065-8944

the workspace and forward/inverse kinematics at position, velocity, and acceleration levels, for a 6-DOF serial-parallel robot composed of two 3UPS-PU parallel robots connected in series.

Regarding singularities of this robot, Liu and Hsu (2007) and Peidró et al. (2019) analyzed its singularity locus in the output and input space, respectively, characterizing some important singularities. Lu et al. (2007) and Pendar et al. (2008) also analyzed some notable singularities of this robot.

Concerning the forward kinematic problem, Joshi and Tsai (2003) solved this problem for 3UPS-PU robots with flat platforms through algebraic elimination, obtaining the symbolic expression of a univariate polynomial whose roots are the solutions of the forward kinematics. Using also algebraic elimination, Innocenti and Wenger (2006) solved the forward kinematics of the general family of 3UPS-PRR manipulators (which includes 3UPS-PU as a special case), reducing the problem to a 28-th degree polynomial.

Algebraic elimination methods are widely used for solving the forward kinematics of parallel manipulators, due to their advantages: they boil down to finding the roots of a polynomial, which is a mature problem for which efficient root-finding techniques exist. They obtain all solutions, including nonreal

#### 324

Peidró, A., Payá, L., Cebollada, S., Román, V. and Reinoso, Ó.

In Proceedings of the 17th International Conference on Informatics in Control, Automation and Robotics (ICINCO 2020), pages 324-334 ISBN: 978-989-758-442-8

Solution of the Forward Kinematic Problem of 3UPS-PU Parallel Manipulators based on Constraint Curves.

Copyright © 2020 by SCITEPRESS – Science and Technology Publications, Lda. All rights reserved

ones. Lastly, they are very fast: once one obtains and codes the symbolic expression of the coefficients of the polynomial in terms of all parameters (joint displacements, links' lengths, etc.), one only needs to substitute the numerical values of these parameters into the expressions and get the roots of the polynomial (e.g., using command roots in Matlab).

However, elimination methods also have disadvantages: the process of eliminating unknowns until arriving at a univariate polynomial may introduce several spurious solutions, and the degree of such polynomial may increase quickly, especially if the elimination steps are not performed carefully. Furthermore, the symbolic expression of the coefficients of the final polynomial can be tremendous, with expressions so large and complex that span several pages (Joshi and Tsai, 2003) in the most optimistic cases. In other cases, they become too complicated to be obtained symbolically even with the aid of computer algebra, having to settle for numerical coefficients instead, obtained by giving concrete or random values to all parameters (Innocenti and Wenger, 2006).

When implementing programming code for the resolution of the kinematic problems of parallel manipulators, with the aim of simulating these problems for research or educational purposes (Peidró et al., 2016), it is desirable to implement resolution methods that provide all solutions (to provide information as complete as possible) and, at the same time, obtain these solutions as quick as possible, so that the user can change some kinematic parameter and instantaneously get the new updated solutions. These requirements are best satisfied by elimination methods, which require typing the symbolic expression of the coefficients of the polynomial in the programming code, parameterized in terms of all involved variables. However, for robots as complex as the general 3UPS-PU manipulator, this is not feasible, due to these coefficients having too large expressions to obtain them, even with the help of computer algebra software.

In order to solve this problem, this paper presents a new method for obtaining all real solutions of the forward kinematics of 3UPS-PU parallel manipulators, with the advantages of elimination methods (i.e., obtaining all solutions quickly) but avoiding the toocomplex-to-obtain symbolic expressions of the coefficients of the polynomial. The proposed method begins by eliminating some unknown of the system of equations (as ordinary elimination methods do), but instead of continue eliminating other unknowns until arriving at a final univariate polynomial of high degree (which has the undesired large coefficients), our method focuses on solving graphically a reduced and simpler system of equations obtained from elim-



Figure 1: 3UPS-PU parallel manipulator.

inating the first unknown. Such reduced system is interpreted geometrically as a constraint curve that contains all the sought solutions. Thus, the method proposed in this paper plots that curve and efficiently searches the desired solutions along it.

The remainder of this paper is organized as follows. Section 2 presents the kinematic model of 3UPS-PU manipulators. In Section 3, this kinematic model is transformed into a set of constraint curves that contain all real solutions of the forward kinematics of this manipulator. A method to search solutions along these curves is described in Section 4. The value of the proposed method is demonstrated through an example in Section 5. Finally, Section 6 summarizes conclusions and points out future work.

#### 2 THE 3UPS-PU MANIPULATOR

This section presents the kinematic model of the 3UPS-PU parallel manipulator, which is depicted in Figure 1. This manipulator is composed of a mobile platform connected to a fixed base through four serial kinematic chains connected in parallel. Three of such kinematic chains (denoted by  $A_i B_i$  in Figure 1) have architecture Universal-Prismatic-Spherical, with their prismatic joints being actuated, allowing one to control the length  $\rho_i$  of each leg (this chain is denoted by UPS, where underline means "actuated"). The remaining kinematic chain OC, with architecture PU, is passive, and grants three degrees of freedom to the relative motion of the mobile platform with respect to the fixed base: a translation z along a passive slider, and two rotations  $(\alpha, \beta)$  about the axes of a passive Universal joint (see Figure 1). Two frames OXYZ and CUVW are attached to the fixed base and mobile platform, respectively. By controlling the lengths  $(\rho_1, \rho_2, \rho_3)$  of the actuated chains, the position *z* and orientation  $(\alpha, \beta)$  of the platform can be controlled.

The kinematic and geometric parameters of the manipulator are defined as follows.  $\theta$  is the constant angle between OC and the Z axis, with OC contained in plane XZ. *z* is the signed position of C along the passive slider OC.  $\alpha$  and  $\beta$  are the angles by which the mobile platform is rotated about the axes of the universal joint centered at C.  $A_i$  is the universal joint of the *i*-th actuated kinematic chain, and  $\mathbf{a}_i = [a_{ix}, a_{iy}, 0]^T$  are its coordinates in frame OXYZ.  $B_i$  is the spherical joint of the *i*-th actuated chain, and  $\mathbf{b}_i = [b_{iu}, b_{iv}, b_{iw}]^T$  are its local coordinates in frame CUVW. The length of each actuated chain is  $\overline{A_i B_i} = \rho_i$ .

Note that we make no simplification for the geometry of the 3UPS-PU manipulator considered in Figure 1, which is fairly general, unlike other previous papers, which mainly focus on simplified designs for which both the mobile platform and fixed base are flat (Peidró et al., 2019). In case that some of the universal joints  $A_1A_2A_3$  were not contained in plane OXY as in Figure 1, one can always define plane XY passing through the centers of the universal joints  $A_1A_2A_3$ , then fix the origin of frame OXYZ in the point where line OC intersects this plane, and define the X and Z axes as the unit projections of line OC on plane XY and on its perpendicular, respectively.

In Figure 1, all variables except for  $(z, \alpha, \beta)$  and  $(\rho_1, \rho_2, \rho_3)$  are constant design parameters, i.e., they are fixed once a concrete design is chosen for this manipulator.  $(\rho_1, \rho_2, \rho_3)$  are the controlled inputs, whereas  $(z, \alpha, \beta)$  are the outputs, which can be controlled indirectly through the inputs. The relationship between inputs and outputs is derived by enforcing the condition that the distance between joints A<sub>i</sub> and B<sub>i</sub> must equal  $\rho_i$  (i = 1, 2, 3), which yields:

$$e_1 = \left\| z \cdot \mathbf{u}_{\theta} + \mathbf{R}_{\theta}^{y} \mathbf{R}_{\alpha}^{x} \mathbf{R}_{\beta}^{y} \mathbf{b}_1 - \mathbf{a}_1 \right\|^2 - \rho_1^2 = 0 \qquad (1)$$

$$e_2 = \left\| z \cdot \mathbf{u}_{\theta} + \mathbf{R}_{\theta}^{y} \mathbf{R}_{\alpha}^{x} \mathbf{R}_{\beta}^{y} \mathbf{b}_2 - \mathbf{a}_2 \right\|^2 - \rho_2^2 = 0 \qquad (2)$$

$$e_{3} = \left\| z \cdot \mathbf{u}_{\theta} + \mathbf{R}_{\theta}^{y} \mathbf{R}_{\alpha}^{x} \mathbf{R}_{\beta}^{y} \mathbf{b}_{3} - \mathbf{a}_{3} \right\|^{2} - \rho_{3}^{2} = 0 \quad (3)$$

where  $\mathbf{u}_{\theta} = [s_{\theta}, 0, c_{\theta}]^T$  ( $s_j$  and  $c_j$  denote the sine and cosine of subscript angle j, respectively), and  $\mathbf{R}_p^q$  denotes a rotation matrix of angle p around axis q.

The problem analyzed in this paper is the forward kinematics, which consists in solving  $(z, \alpha, \beta)$  from Eqs. (1)-(3) for given known values of  $(\rho_1, \rho_2, \rho_3)$ . A new solution to this problem will be presented in the next sections.

#### **3 CONSTRAINT CURVES**

For given  $\rho_i$ , Eqs. (1)-(3) constitute a nonlinear system of three equations in three unknowns  $(z, \alpha, \beta)$ . A typical method for solving these systems consists in performing algebraic manipulations in order to eliminate equations and variables, until reducing the original system to a univariate polynomial in one of the unknowns (e.g., z), which can be solved via well-known root-finding algorithms. However, as explained in the introduction, this method has some shortcomings: the final univariate polynomial obtained can have too high a degree if the elimination steps are not done carefully, spurious solutions may appear (i.e., solutions that do not satisfy the original system), and the general symbolic expression of the coefficients of this polynomial is usually too complex or impossible to obtain even with computer algebra software due to involving many terms, which prevents parameterizing the solution in terms of general parameters. The method proposed in this paper begins by eliminating one unknown from the system (z), reducing it to a set of curves that constrain the admissible combinations of the other two unknowns ( $\alpha$  and  $\beta$ ). Then, these curves can be searched for real solutions of the original system, without needing to further reduce the system to a univariate polynomial, which would lead to a high-degree polynomial with excessively complicated coefficients and likely spurious solutions.

First, note that the squared norm in Eqs. (1)-(3) can be expanded as follows:

$$e_i = z^2 + a_i^2 + b_i^2 + 2z \mathbf{u}_{\theta}^T (\mathbf{R} \mathbf{b}_i - \mathbf{a}_i) - 2\mathbf{a}_i^T \mathbf{R} \mathbf{b}_i - \rho_i^2$$
(4)  
where  $a_i^2 = \|\mathbf{a}_i\|^2$ ,  $b_i^2 = \|\mathbf{b}_i\|^2$ , and  $\mathbf{R} = \mathbf{R}_{\theta}^y \mathbf{R}_{\alpha}^x \mathbf{R}_{\beta}^y$ .  
Considering Eq. (4), it can be checked that subtracting  
Eq. (3) from Eq. (1) and dividing by 2 yields:

$$k_{13} + z \mathbf{u}_{\theta}^T (\mathbf{R} \mathbf{b}_{13} - \mathbf{a}_{13}) - \operatorname{tr}(\mathbf{c}_{13} \mathbf{R}) = 0 \qquad (5)$$

where the following parameters are constants:

$$k_{ij} = 0.5 \left[ (a_i^2 - a_j^2) + (b_i^2 - b_j^2) - (\rho_i^2 - \rho_j^2) \right]$$
(6)  
$$\mathbf{a}_{ij} = \mathbf{a}_{ij} - \mathbf{a}_{jj}$$
(7)

$$\mathbf{a}_{ij} = \mathbf{a}_i - \mathbf{a}_j \tag{7}$$

$$\mathbf{b}_{ij} = \mathbf{b}_i - \mathbf{b}_j \tag{8}$$

$$\mathbf{c}_{ij} = \mathbf{b}_i \mathbf{a}_i^T - \mathbf{b}_j \mathbf{a}_j^T \tag{9}$$

and where  $tr(\cdot)$  is the trace of a matrix. In order to obtain Eq. (5) from (4), it is necessary to make use of the linearity of the trace operator, and consider that  $\mathbf{a}^T \mathbf{R} \mathbf{b} = tr(\mathbf{b} \mathbf{a}^T \mathbf{R})$ . Similarly, subtracting Eq. (3) from Eq. (2) and dividing by 2 yields:

$$k_{23} + z \mathbf{u}_{\theta}^{T} (\mathbf{R} \mathbf{b}_{23} - \mathbf{a}_{23}) - \operatorname{tr}(\mathbf{c}_{23} \mathbf{R}) = 0 \qquad (10)$$

Note that subtracting  $e_3$  from  $e_1$  and  $e_2$  cancels the quadratic term  $z^2$  which was present in Eq. (4), and

the two new equations (5) and (10), which replace the original quadratic Eqs. (1)-(2), are linear in *z*. Accordingly, for a given pair ( $\alpha$ ,  $\beta$ ) to yield the same value of *z* from these two linear equations (i.e., to enforce these two equations to be consistent), the determinant of the following 2 × 2 matrix **M** must vanish:

$$\mathbf{M} = \begin{bmatrix} \mathbf{u}_{\theta}^{T}(\mathbf{R}\mathbf{b}_{13} - \mathbf{a}_{13}) & \operatorname{tr}(\mathbf{c}_{13}\mathbf{R}) - k_{13} \\ \mathbf{u}_{\theta}^{T}(\mathbf{R}\mathbf{b}_{23} - \mathbf{a}_{23}) & \operatorname{tr}(\mathbf{c}_{23}\mathbf{R}) - k_{23} \end{bmatrix}$$
(11)

$$\det(\mathbf{M}) = 0 \tag{12}$$

Recall that  $\mathbf{R} = \mathbf{R}_{\theta}^{y} \mathbf{R}_{\alpha}^{x} \mathbf{R}_{\beta}^{y}$  is a function of  $(\alpha, \beta)$ , so the only unknowns in Eq. (12) are  $(\alpha, \beta)$  (z has been eliminated from the system). Thus, Eq. (12) defines a set of constraint curves (possibly multiple disjoint curves) in plane  $(\alpha, \beta)$ , which contain the solutions of the forward kinematic problem. In particular, the points of these curves where  $e_3 = 0$  will be the sought solutions, since the other two equations of the system are automatically satisfied for some real z due to  $(\alpha,\beta)$  belonging to the curves defined by Eq. (12). The algorithm which will be described in the next Section 4 will perform an efficient search along these curves, looking for zero-crossings of  $e_3$ . However, before presenting the proposed algorithm, we will analyze in more detail the degree of such curves in plane  $(\alpha, \beta)$ . To that end, these curves will be intersected with straight lines for which  $\alpha$  or  $\beta$  remain constant.

#### 3.1 Intersection with Constant $\alpha$

The objective of this subsection is to intersect the curves defined by Eq. (12) in plane  $(\alpha, \beta)$  with a straight line defined by  $\alpha$  = constant. If  $\alpha$  is constant and known, then **M** is a function of the only unknown  $\beta$ , so **M** can be written as follows:

$$\mathbf{M} = \mathbf{C}_{\alpha} c_{\beta} + \mathbf{S}_{\alpha} s_{\beta} + \mathbf{Y}_{\alpha} \tag{13}$$

where the following newly defined matrices:

$$\mathbf{C}_{\alpha} = \begin{bmatrix} \mathbf{u}_{\theta}^{T} \mathbf{R}_{\theta}^{y} \mathbf{R}_{\alpha}^{x} \mathbf{B}_{c} \mathbf{b}_{13} & \text{tr}(\mathbf{c}_{13} \mathbf{R}_{\theta}^{y} \mathbf{R}_{\alpha}^{x} \mathbf{B}_{c}) \\ \mathbf{u}_{\theta}^{T} \mathbf{R}_{\theta}^{y} \mathbf{R}_{\alpha}^{x} \mathbf{B}_{c} \mathbf{b}_{23} & \text{tr}(\mathbf{c}_{23} \mathbf{R}_{\theta}^{y} \mathbf{R}_{\alpha}^{x} \mathbf{B}_{c}) \end{bmatrix}$$
(14)

$$\mathbf{S}_{\alpha} = \begin{bmatrix} \mathbf{u}_{\theta}^{T} \mathbf{R}_{\theta}^{Y} \mathbf{R}_{\alpha}^{X} \mathbf{B}_{s} \mathbf{b}_{13} & \operatorname{tr}(\mathbf{c}_{13} \mathbf{R}_{\theta}^{T} \mathbf{R}_{\alpha}^{X} \mathbf{B}_{s}) \\ \mathbf{u}_{\theta}^{T} \mathbf{R}_{\theta}^{Y} \mathbf{R}_{\alpha}^{X} \mathbf{B}_{s} \mathbf{b}_{23} & \operatorname{tr}(\mathbf{c}_{23} \mathbf{R}_{\theta}^{Y} \mathbf{R}_{\alpha}^{X} \mathbf{B}_{s}) \end{bmatrix}$$
(15)

$$\begin{bmatrix} \mathbf{u}_{\theta}^{T}(\mathbf{R}_{\theta}^{y}\mathbf{R}_{\alpha}^{x}\mathbf{B}_{Y}\mathbf{b}_{13}-\mathbf{a}_{13}) & \operatorname{tr}(\mathbf{c}_{13}\mathbf{R}_{\theta}^{y}\mathbf{R}_{\alpha}^{x}\mathbf{B}_{Y})-k_{13} \\ \mathbf{u}_{\theta}^{T}(\mathbf{R}_{\theta}^{y}\mathbf{R}_{\alpha}^{x}\mathbf{B}_{Y}\mathbf{b}_{23}-\mathbf{a}_{23}) & \operatorname{tr}(\mathbf{c}_{23}\mathbf{R}_{\theta}^{y}\mathbf{R}_{\alpha}^{x}\mathbf{B}_{Y})-k_{23} \end{bmatrix}$$
(16)

v

are constant for constant  $\alpha$ . In Eqs. (14)-(16), the following sparse matrices have been introduced:

$$\mathbf{B}_{c} = \operatorname{diag}(1,0,1), \qquad \mathbf{B}_{s} = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 0 & 0 \\ -1 & 0 & 0 \end{bmatrix}$$
(17)

where diag $(m_1, m_2, ..., m_n)$  denotes an  $n \times n$  diagonal matrix with numbers  $m_1, m_2, ..., m_n$  in its diagonal. The rewriting of Eq. (11) as Eqs. (13)-(16) has been achieved by decomposing the rotation  $\mathbf{R}^y_\beta$  as follows:

$$\mathbf{R}_{\beta}^{\gamma} = \begin{bmatrix} c_{\beta} & 0 & s_{\beta} \\ 0 & 1 & 0 \\ -s_{\beta} & 0 & c_{\beta} \end{bmatrix} = \mathbf{B}_{c}c_{\beta} + \mathbf{B}_{s}s_{\beta} + \mathbf{B}_{Y} \quad (18)$$

Now, performing the *tangent of half-angle* substitution in Eq. (13), i.e.:

$$c_{\beta} = \frac{1 - t_{\beta}^2}{1 + t_{\beta}^2}, \quad s_{\beta} = \frac{2t_{\beta}}{1 + t_{\beta}^2}, \quad \text{with } t_{\beta} = \tan\frac{\beta}{2} \quad (19)$$

Using (19), M in Eq. (13) will be written as follows:

$$\mathbf{M} = \frac{\mathbf{C}_{\alpha}(1 - t_{\beta}^2) + \mathbf{S}_{\alpha} 2t_{\beta} + \mathbf{Y}_{\alpha}(1 + t_{\beta}^2)}{1 + t_{\beta}^2}$$
(20)

Inserting (20) into (12) yields:

$$\frac{\det\left[\mathbf{C}_{\alpha}(1-t_{\beta}^{2})+\mathbf{S}_{\alpha}2t_{\beta}+\mathbf{Y}_{\alpha}(1+t_{\beta}^{2})\right]}{(1+t_{\beta}^{2})^{2}}=0 \quad (21)$$

Clearing the denominator in (21) and expanding the determinant in the numerator yields the following quartic equation in  $t_{\beta}$ :

$$\omega_4 t_\beta^4 + \omega_3 t_\beta^3 + \omega_2 t_\beta^2 + \omega_1 t_\beta + \omega_0 = 0, \qquad (22)$$

where the coefficients  $\omega_i$  are computed as follows:

$$\omega_{4} = C_{\alpha}^{11} \cdot (C_{\alpha}^{22} - Y_{\alpha}^{22}) + C_{\alpha}^{12} \cdot (Y_{\alpha}^{21} - C_{\alpha}^{21}) + C_{\alpha}^{21} \cdot Y_{\alpha}^{12} - C_{\alpha}^{22} \cdot Y_{\alpha}^{11} + Y_{\alpha}^{11} \cdot Y_{\alpha}^{22} - Y_{\alpha}^{12} \cdot Y_{\alpha}^{21}$$
(23)

$$\omega_{3} = -2 \cdot (C_{\alpha} \cdot S_{\alpha} - C_{\alpha} \cdot S_{\alpha} - C_{\alpha} \cdot S_{\alpha} + C_{\alpha}^{22} \cdot S_{\alpha}^{11} - S_{\alpha}^{11} \cdot Y_{\alpha}^{22} + S_{\alpha}^{12} \cdot Y_{\alpha}^{21} + S_{\alpha}^{21} \cdot Y_{\alpha}^{12} - S_{\alpha}^{22} \cdot Y_{\alpha}^{11})$$
(24)

$$\begin{split} \omega_{2} &= -2 \cdot \left( C_{\alpha}^{11} \cdot C_{\alpha}^{22} - C_{\alpha}^{12} \cdot C_{\alpha}^{21} - 2 \cdot S_{\alpha}^{11} \cdot S_{\alpha}^{22} \right. (25) \\ &+ 2 \cdot S_{\alpha}^{12} \cdot S_{\alpha}^{21} - Y_{\alpha}^{11} \cdot Y_{\alpha}^{22} + Y_{\alpha}^{12} \cdot Y_{\alpha}^{21} \right) \\ \omega_{1} &= 2 \cdot \left( C_{\alpha}^{11} \cdot S_{\alpha}^{22} - C_{\alpha}^{12} \cdot S_{\alpha}^{21} - C_{\alpha}^{21} \cdot S_{\alpha}^{12} \right) \end{split}$$

$$+C_{\alpha}^{22} \cdot S_{\alpha}^{11} + S_{\alpha}^{11} \cdot Y_{\alpha}^{22} - S_{\alpha}^{12} \cdot Y_{\alpha}^{21}$$

$$-S_{\alpha}^{21} \cdot Y_{\alpha}^{12} + S_{\alpha}^{22} \cdot Y_{\alpha}^{11})$$
(26)

$$\omega_{0} = C_{\alpha}^{11} \cdot (C_{\alpha}^{22} + Y_{\alpha}^{22}) - C_{\alpha}^{12} \cdot (C_{\alpha}^{21} + Y_{\alpha}^{21}) - C_{\alpha}^{21} \cdot Y_{\alpha}^{12} + C_{\alpha}^{22} \cdot Y_{\alpha}^{11} + Y_{\alpha}^{11} \cdot Y_{\alpha}^{22} - Y_{\alpha}^{12} \cdot Y_{\alpha}^{21}$$
(27)

The coefficients  $C_{\alpha}^{ij}$ ,  $S_{\alpha}^{ij}$ , and  $Y_{\alpha}^{ij}$  denote the entries in the *i*-th row and *j*-th column of matrices  $\mathbf{C}_{\alpha}$ ,  $\mathbf{S}_{\alpha}$ , and  $\mathbf{Y}_{\alpha}$ , respectively. All these coefficients are known constants for a given value of  $\alpha$ . The roots of (22) provide the intersection points between the constraint curves defined by Eq. (12) and a straight line defined by  $\alpha$  = constant. Since Eq. (22) is a quartic polynomial, it will have a maximum of four real roots, i.e., there will be up to four such intersection points. For each real root  $t_{\beta}^*$  of this polynomial, the corresponding value of  $\beta$  is obtained as  $\beta^* = 2 \arctan(t_{\beta}^*)$ . Finally, it is worth noting that Eq. (22) can be solved analytically, since its degree is lower than five (it admits closed-form solution).

#### **3.2** Intersection with Constant $\beta$

This case is analogous to the problem studied in §3.1. In this case, we wish to compute the intersections of the constraint curves defined by Eq. (12) in plane  $(\alpha, \beta)$  with a line defined by  $\beta$  = constant. Following similar steps as in §3.1, we begin by decomposing the rotation matrix  $\mathbf{R}^{x}_{\alpha}$  as follows:

$$\mathbf{R}_{\alpha}^{x} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & c_{\alpha} & -s_{\alpha} \\ 0 & s_{\alpha} & c_{\alpha} \end{bmatrix} = \mathbf{A}_{c}c_{\alpha} + \mathbf{A}_{s}s_{\alpha} + \mathbf{A}_{Y} \quad (28)$$

where  $A_c$ ,  $A_s$  and  $A_Y$  are the following matrices:

$$\mathbf{A}_{c} = \operatorname{diag}(0, 1, 1), \\ \mathbf{A}_{Y} = \operatorname{diag}(1, 0, 0),$$
 
$$\mathbf{A}_{s} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & -1 \\ 0 & 1 & 0 \end{bmatrix}$$
(29)

Inserting (28) into (11), then M is rewritten as:

$$\mathbf{M} = \mathbf{C}_{\beta} c_{\alpha} + \mathbf{S}_{\beta} s_{\alpha} + \mathbf{Y}_{\beta} \tag{30}$$

where the matrices  $C_{\beta}$ ,  $S_{\beta}$  and  $Y_{\beta}$  are constant for a given  $\beta$ , and they have the following expressions:

$$\mathbf{C}_{\beta} = \begin{bmatrix} \mathbf{u}_{\theta}^{T} \mathbf{R}_{\theta}^{y} \mathbf{A}_{c} \mathbf{R}_{\beta}^{y} \mathbf{b}_{13} & \text{tr}(\mathbf{c}_{13} \mathbf{R}_{\theta}^{y} \mathbf{A}_{c} \mathbf{R}_{\beta}^{y}) \\ \mathbf{u}_{\theta}^{T} \mathbf{R}_{\theta}^{y} \mathbf{A}_{c} \mathbf{R}_{\beta}^{y} \mathbf{b}_{23} & \text{tr}(\mathbf{c}_{23} \mathbf{R}_{\theta}^{y} \mathbf{A}_{c} \mathbf{R}_{\beta}^{y}) \end{bmatrix}$$
(31)

$$\mathbf{S}_{\beta} = \begin{bmatrix} \mathbf{u}_{\theta}^{T} \mathbf{R}_{\theta}^{y} \mathbf{A}_{s} \mathbf{R}_{\beta}^{y} \mathbf{b}_{13} & \operatorname{tr}(\mathbf{c}_{13} \mathbf{R}_{\theta}^{y} \mathbf{A}_{s} \mathbf{R}_{\beta}^{y}) \\ \mathbf{u}_{\theta}^{T} \mathbf{R}_{\theta}^{y} \mathbf{A}_{s} \mathbf{R}_{\beta}^{y} \mathbf{b}_{23} & \operatorname{tr}(\mathbf{c}_{23} \mathbf{R}_{\theta}^{y} \mathbf{A}_{s} \mathbf{R}_{\beta}^{y}) \end{bmatrix}$$
(32)

$$\mathbf{Y}_{\beta} = \begin{bmatrix} \mathbf{u}_{\theta}^{T} (\mathbf{R}_{\theta}^{y} \mathbf{A}_{Y} \mathbf{R}_{\beta}^{y} \mathbf{b}_{13} - \mathbf{a}_{13}) & \operatorname{tr} (\mathbf{c}_{13} \mathbf{R}_{\theta}^{y} \mathbf{A}_{Y} \mathbf{R}_{\beta}^{y}) - k_{13} \\ \mathbf{u}_{\theta}^{T} (\mathbf{R}_{\theta}^{y} \mathbf{A}_{Y} \mathbf{R}_{\beta}^{y} \mathbf{b}_{23} - \mathbf{a}_{23}) & \operatorname{tr} (\mathbf{c}_{23} \mathbf{R}_{\theta}^{y} \mathbf{A}_{Y} \mathbf{R}_{\beta}^{y}) - k_{23} \end{bmatrix}$$
(33)

Next, the same steps performed between Eqs. (19)-(22) should be repeated here: the tangent of halfangle substitution  $[\alpha = 2 \arctan(t_{\alpha})]$  is performed in Eq. (30) and the determinant of **M** is expanded, arriving at the following quartic equation:

$$\omega_4 t_\alpha^4 + \omega_3 t_\alpha^3 + \omega_2 t_\alpha^2 + \omega_1 t_\alpha + \omega_0 = 0, \qquad (34)$$

where the coefficients  $\omega_i$  depend on the (constant) entries  $C_{\beta}^{ij}$ ,  $S_{\beta}^{ij}$ , and  $Y_{\beta}^{ij}$  of matrices  $C_{\beta}$ ,  $S_{\beta}$ , and  $Y_{\beta}$ , and these coefficients can be computed using Eqs. (23)-(27), by substituting every  $\alpha$  for  $\beta$  in these equations.

We conclude from the analysis of Sections 3.1 and 3.2 that, when intersecting the constraint curves defined by Eq. (12) in plane  $(\alpha, \beta)$  with straight lines for which  $\alpha$  or  $\beta$  are constant, yields up to four intersection points, and these points can be obtained by solving analytically the quartic polynomials of Eqs. (22) or (34). This will be exploited in the method presented in the next section for finding all real solutions of the forward kinematics of 3UPS-PU manipulators.

#### 4 PROPOSED METHOD

The constraint curve defined by Eq. (12) in plane  $(\alpha, \beta)$  contains the real solutions of the forward kinematics of the 3UPS-PU manipulator. The method proposed in the present section consists in searching the desired solutions along these curves. This method consists of some steps, beginning by generating these curves, as explained in the following subsection.

#### 4.1 Step 1: Generating the Constraint Curves

In order to search for solutions along the constraint curves implicitly defined by Eq. (12), first it is necessary to generate, "render" or "plot" these curves in plane ( $\alpha$ ,  $\beta$ ), which will be accomplished by the sampling method described next, which was proposed in (Peidró et al., 2018). This method consists in generating a sequence by varying  $\alpha$  between  $-\pi$  rad and  $+\pi$  rad using some step  $\Delta\alpha$ , so that for each value of  $\alpha$  in this sequence  $\{-\pi, -\pi + \Delta\alpha, -\pi + 2\Delta\alpha, ..., \pi - 2\Delta\alpha, \pi - \Delta\alpha, \pi\}$ ,  $\beta$  is solved from Eq. (22), obtaining up to four real solutions. Geometrically, this process consists in intersecting the constraint curves with a bundle of parallel vertical lines uniformly separated a distance  $\Delta\alpha$  (see Figure 2a), storing all the obtained intersection points as a point cloud *Cloud*<sub> $\alpha$ </sub>.

This point cloud is a discrete approximation of the constraint curves, which may be used for searching the desired solutions of the forward kinematics, by checking if  $e_3$  changes sign between two neighboring intersection points (e.g., Q and R in Figure 2a). However, in order for this strategy to work satisfactorily, the density of intersection points along the curves should be approximately uniform, which is not accomplished yet. Indeed, note that, in Figure 2a, the regions where the constraint curves have vertical tan-



Figure 2: Discrete approximations of the constraint curves obtained by intersecting them with a bundle of vertical lines (a), horizontal lines (b), and a combination of both (c).

gent have lower density of intersection points (e.g., region S in Figure 2a). As a result of this lower density, neighboring points in these regions are too separated and the algorithm may consider that they are not truly neighbors, so it would not check if potential solutions exist between them, which may miss solutions.

In order to avoid this, another sequence is defined, this time by varying  $\beta$  between  $-\pi$  rad and  $\pi$  rad with some step  $\Delta\beta$ , and for each value of  $\beta$  in this sequence  $\{-\pi, -\pi + \Delta\beta, -\pi + 2\Delta\beta, ..., \pi - 2\Delta\beta, \pi - \Delta\beta, \pi\}$ ,  $\alpha$  is solved from Eq. (34), obtaining four real solutions at most. Geometrically, this process consists in intersecting the constraint curves with a bundle of horizontal parallel lines uniformly separated a distance of  $\Delta\beta$  (see Figure 2b), storing all the intersection points as another point cloud *Cloud*<sub> $\beta$ </sub>. Comparing Figures 2a-b, one sees that the low-density regions in Figure 2a (e.g., region S) are now better defined in Figure 2b, but at the expense of getting poorly sampled regions where the constraint curves have horizontal tangent (e.g., region T in Figure 2b).

As noted in (Simionescu, 2014; Peidró et al., 2018), the union of both point clouds yields a more accurate point cloud  $Cloud_{\gamma} = Cloud_{\alpha} \cup Cloud_{\beta}$  that presents an *approximately* uniform density of points along the curves, since the poorly defined regions of  $Cloud_{\alpha}$  are compensated by  $Cloud_{\beta}$  and vice versa (see Figure 2c). This is the solution that will be adopted here for obtaining a discrete approximation of the constraint curves, which can be searched for solutions of the forward kinematic problem.

After this, a set  $Cloud_{\gamma} = \{p_1, p_2, p_3, ..., p_N\}$  of points along the constraint curves is obtained. In general, these points do not follow any useful order in  $Cloud_{\gamma}$ , since they are points of plane  $(\alpha, \beta)$  that are stored in memory in the same order as they are obtained by intersecting the constraint curves with the aforementioned sequences of vertical and horizontal lines. Therefore, in order to search for solutions between these sampled points, it will be necessary to arrange these points in some data structure that allows for efficient searches of the neighbors of each point. This will be done in Step 2 of the proposed method.

Before proceeding to the next steps of the method, let us define a *unique* companion point  $\tilde{p}_i = (z_i, e_{3i})$ associated to each  $p_i = (\alpha_i, \beta_i)$ .  $\tilde{p}_i$  is calculated from  $p_i$  as follows: first,  $z_i$  is obtained by substituting  $(\alpha_i, \beta_i)$  into Eq. (5) or (10) and solving for z. Any of these linear equations is equally valid: since  $(\alpha_i, \beta_i)$ belongs to the constraint curve (they satisfy Eq. 12), it is guaranteed that these two equations provide the same solution for z. Then,  $\alpha_i, \beta_i$ , and  $z_i$  can be substituted into Eq. (3) to obtain a unique value of  $e_{3i}$ .

#### 4.2 Step 2: Sorting the Point Cloud

The second step of the proposed method consists in organizing the point cloud  $Cloud_{\gamma}$  using k-d (kdimensional) trees, which are data structures that sort points in a k-dimensional space (in this paper, k=2). A k-d tree  $\mathcal{T}$  is obtained from a point cloud by recursively dividing it at the point whose coordinate is the median (and the coordinate used for doing this division is switched cyclically after each division). For example: Figure 3a shows a set of seven points 1...7 in plane  $(\alpha, \beta)$ . Point 5 is the median along coordinate  $\alpha$ , so it divides the set into two branches  $br_1 = \{1, 4, 6\}$ and  $br_r = \{2, 3, 7\}$ . This division is performed again inside these two branches, but using now points 6 and 2, since these are the medians along coordinate  $\beta$  in each branch. This generates four new branches:  $br_{lu} = \{4\}, br_{ld} = \{1\}, br_{ru} = \{7\} \text{ and } br_{rd} = \{3\}.$  If we had more points, this process would continue recursively, alternating cyclically the coordinate along which the median divides each branch ( $\alpha$ ,  $\beta$ ,  $\alpha$ ,  $\beta$ ...). The resulting k-d tree is shown in Figure 3b.



Figure 3: A set of points (a), and their k-d tree (b).

Building a k-d tree is useful for efficiently retrieving data from a point cloud, e.g., obtaining all points whose coordinates lie between specific limits (this is what will be done in this paper). Since each layer divides the previous one into two sets depending if their coordinate is below or above the median, retrieving all points whose coordinates lie within desired limits is done faster than by checking *all* of them, since a k-d tree search discards half of the data at each layer. See (Press et al., 2007) for methods to build k-d trees.

#### 4.3 Step 3: Searching Solutions

After building a k-d tree data structure  $\mathcal{T}$  to sort the point cloud  $Cloud_{\gamma}$ , Algorithm 1 is executed to search solutions along the curves. First, a set S is defined as the empty set (line 1). After the algorithm finishes, this set S will contain the sought solutions. Then, for each point  $p_i = (\alpha_i, \beta_i) \in Cloud_{\gamma}$  (line 2), a set  $\mathcal{N}(p_i, r_{\alpha}, r_{\beta})$  of neighbors of  $p_i$  is retrieved from the k-d tree  $\mathcal{T}$  (line 3).  $\mathcal{N}(p_i, r_{\alpha}, r_{\beta})$  is defined as follows:

$$\mathcal{N}(p_i, r_{\alpha}, r_{\beta}) = Cloud_{\gamma} \cap \mathcal{B}(p_i, r_{\alpha}, r_{\beta}) \setminus \{p_i\} \quad (35)$$

where:

$$\mathcal{B}\left(p_{i} = (\alpha_{i}, \beta_{i}), r_{\alpha}, r_{\beta}\right) = \left\{\left(\alpha_{j}, \beta_{j}\right) : \max\left(\frac{|\alpha_{j} - \alpha_{i}|}{r_{\alpha}}, \frac{|\beta_{j} - \beta_{i}|}{r_{\beta}}\right) \le 1\right\} \quad (36)$$

Note that  $\mathcal{N}$  is composed of all points  $p_j$  of  $Cloud_\gamma$ (excluding  $p_i$ ) which fall inside a box  $\mathcal{B}$  centered at  $p_i$  and with width  $2r_\alpha$  and height  $2r_\beta$ , with  $r_\alpha$  and  $r_\beta$  being small radii (see Figure 4a). This retrieval operation is very fast because  $p_j$  have been arranged into a k-d tree  $\mathcal{T}$  previously (step 2), which allows one to search very efficiently points whose coordinates lie in specified intervals, like those imposed by box  $\mathcal{B}$ .

When retrieving the neighbors  $p_j$  of  $p_i$ , one should be careful with the fact that all these points



Figure 4: (a) Box  $\mathcal{B}$  containing the neighbors of  $p_i$ . (b) Constraint curves wrapped at  $\alpha, \beta = \pm \pi$ , with wrapped boxes  $\mathcal{B}_{uv}$ . Boxes are exaggerated here, they usually are smaller.

Algorithm 1: Searching solutions S along the constraint curves defined by Eq. (12).

_					
1:	$\mathcal{S} = \emptyset$				
2: for all $p_i = (\alpha_i, \beta_i) \in Cloud_{\gamma}$ do					
3:	Retrieve $\mathcal{N}(p_i, r_{\alpha}, r_{\beta})$ from $\mathcal{T}$				
4:	for all $p_j = (\alpha_j, \beta_j) \in \mathcal{N}(p_i, r_\alpha, r_\beta)$ do				
5:	if $e_{3i} \cdot e_{3j} \leq 0$ then				
6:	$\boldsymbol{\alpha}^* = (e_{3i}\boldsymbol{\alpha}_j - e_{3j}\boldsymbol{\alpha}_i)/(e_{3i} - e_{3j})$				
7:	$\beta^* = (e_{3i}\beta_j - e_{3j}\beta_i)/(e_{3i} - e_{3j})$				
8:	$z^* = (e_{3i}z_j - e_{3j}z_i)/(e_{3i} - e_{3j})$				
9:	$\mathbf{s}^* = [oldsymbollpha^*,oldsymboleta^*,z^*]^T$				
10:	if $\mathbf{s}^{*} \notin \mathcal{B}\left(\mathbf{s}, r_{oldsymbol{lpha}}, r_{oldsymbol{eta}} ight) orall \mathbf{s} \in \mathcal{S}$ then				
11:	do				
12:	$\Delta \mathbf{s}^* = -\mathbf{J}(\mathbf{s}^*)^{-1}\mathbf{F}(\mathbf{s}^*)$				
13:	$\mathbf{s}^* \leftarrow \mathbf{s}^* + \Delta \mathbf{s}^*$				
14:	while $\ \Delta \mathbf{s}^*\  > \epsilon$				
15:	$\mathcal{S} \leftarrow \mathcal{S} \cup \{\mathbf{s}^*\}$				

lie on the constraint curves in plane  $(\alpha, \beta)$ , which is a plane of angles. Since  $\alpha$  and  $\beta$  appear in Eq. (12) inside sines and cosines, the solution set of Eq. (12) in plane  $(\alpha, \beta)$  is periodic every  $2\pi$  rad along both axes. This is why  $\alpha$  and  $\beta$  were swept between  $-\pi$ and  $+\pi$  when generating the curves as explained in Section 4.1: all information is confined to this region  $\mathcal{R} = [-\pi, \pi] \times [-\pi, \pi]$  of the plane. As a result of this, the curves defined by Eq. (12) wrap at the lines  $\alpha = \pm \pi$  and  $\beta = \pm \pi$ , as illustrated in Figure 4b: e.g., a point  $W_1$  marching along these curves and crossing the vertical line  $\alpha = \pi$  in positive direction as indicated in Figure 4b would reappear as if coming from  $\alpha = -\pi$ , as indicated by point  $W_2$  in the same figure.

Due to this wrapping, when retrieving the neighbors of a point  $p_i$  that is close to the borders of region  $\mathcal{R}$ , one should not omit the *wrapped* neighbors of such  $p_i$ . For example: the neighbors to the right

of  $W_2$  in Figure 4b are also neighbors of  $W_1$  because, due to wrapping,  $W_1$  and  $W_2$  are identified as the same point. However, k-d tree methods for sorting and retrieving points do not work with wrapped angular coordinates, and would fail to identify the neighbors to the right of  $W_2$  as neighbors of  $W_1$  also.

To avoid this wrapping of angular coordinates, Peidró et al. (2018) proposed using the sines  $s_i$  and cosines  $c_i$  of angles as the variables of the problem, instead of working with the angles j directly, imposing the additional constraint:  $s_i^2 + c_j^2 = 1$ . This indeed avoids the problem of wrapping and permits using kd trees to sort and search points, since the sines and cosines do not wrap as their angles, but it also has the drawback of doubling the dimension of the problem, which increases computational time: each original angular variable *j* is substituted by two variables (the sine and cosine of j). In this paper, we propose an alternative method for searching points using k-d trees when these points have angular coordinates that wrap along the axes of the ambient space, without having to double the dimension of such a space by unfolding each angle into its sine and cosine.

The alternative method is as follows: in line 3 of Algorithm 1, when using k-d trees to retrieve the set  $\mathcal{N}$  of neighboring points inside a box  $\mathcal{B}$  centered at  $p_i$ , one must "replicate" this box if it intersects any of the wrapping lines  $\alpha = \pm \pi$  or  $\beta = \pm \pi$ , defining "replica boxes" whose centers are translated a distance of  $2\pi$ along the wrapped axes. More formally, let us define an auxiliary variable  $\sigma_{\alpha}$  as follows:

$$\sigma_{\alpha} = \begin{cases} +1 & \text{if } \mathcal{B} \text{ intersects the line } \alpha = +\pi \\ -1 & \text{if } \mathcal{B} \text{ intersects the line } \alpha = -\pi \\ 0 & \text{otherwise} \end{cases}$$
(37)

In (37), it is assumed that  $\mathcal{B}$  will not intersect simultaneously both lines  $\alpha = \pi$  and  $\alpha = -\pi$  since this box should be small, as in Figure 4b. Let us define an analogous variable  $\sigma_{\beta}$ , by substituting every  $\alpha$  in (37) for  $\beta$ . After defining  $\sigma_{\alpha}$  and  $\sigma_{\beta}$ , one only has to change  $\mathcal{B}$  in Eq. (35) for  $\overline{\mathcal{B}}$ , which is defined as follows:

$$\overline{\mathcal{B}}\left(p_{i}, r_{\alpha}, r_{\beta}\right) = \bigcup_{u=0}^{|\boldsymbol{\sigma}_{\alpha}|} \bigcup_{\nu=0}^{|\boldsymbol{\sigma}_{\beta}|} \mathcal{B}_{u\nu}(p_{i}, r_{\alpha}, r_{\beta}, \boldsymbol{\sigma}_{\alpha}, \boldsymbol{\sigma}_{\beta}) \quad (38)$$

where:

$$\mathcal{B}_{uv}(p_i, r_{\alpha}, r_{\beta}, \sigma_{\alpha}, \sigma_{\beta}) = \mathcal{B}\left(p_i - 2\pi(u\sigma_{\alpha}, v\sigma_{\beta}), r_{\alpha}, r_{\beta}\right)$$
(39)

and where  $\mathcal{B}$  is defined as in Eq. (36). Equation (38) effectively defines the union of  $2^{|\sigma_{\alpha}|+|\sigma_{\beta}|}$  wrapped copies or replicas of box  $\mathcal{B}$ , with each copy being translated a distance of  $2\pi$  along each axis in order to cover all the wrapped regions intersected with

the original box  $\mathcal{B}_{00}$ . For example, Figure 4b shows two boxes  $\mathcal{B}_{00}^1$  and  $\mathcal{B}_{00}^2$ , which intersect the wrapping lines  $\alpha = \pm \pi$  or  $\beta = \pm \pi$ , and their wrapped replicas  $\{\mathcal{B}_{00}^1, \mathcal{B}_{01}^1\}$  and  $\{\mathcal{B}_{00}^2, \mathcal{B}_{01}^2, \mathcal{B}_{10}^2, \mathcal{B}_{11}^2\}$ , whose centers are translated as indicated by Eq. (39).

Then, in order to retrieve the wrapped neighbors of  $p_i$  using k-d trees, one has to perform a search for all points in k-d tree  $\mathcal{T}$  whose coordinates fall in the intervals defined by each box  $\mathcal{B}_{uv}$  separately.

Continuing with Algorithm 1, after retrieving all neighbors  $p_i$  of current point  $p_i$  (including possible wrapped neighbors), then the sign of  $e_{3i}$  for the current point  $p_i$  is compared to the sign of  $e_{3i}$  for each neighbor. If  $e_3$  changes sign between  $p_i$  and any  $p_j$ of its neighbors, a solution exists between them, and this solution is approximated by a linear interpolation between  $p_i$  and  $p_j$ , by enforcing  $e_3 = 0$ . This linear interpolation is performed between lines 6-9 of Algorithm 1. Note that not only the coordinates  $(\alpha, \beta)$  are interpolated, but also z (line 8). Recall that the values of z and  $e_3$  are stored in each companion point  $\tilde{p}_i$  of  $p_i$  computed during the first step of the method. Note also that, in case of wrapping, when interpolating  $\alpha$ and  $\beta$  in lines 6-7, the values of  $(\alpha_i, \beta_i)$  must be those of point  $p_i$  translated by the displacement of Eq. (39).

After this linear interpolation, an approximate solution  $\mathbf{s}^*$  is obtained (line 9 of Algorithm 1). This solution is compared to all other solutions stored in Sso far (line 10), and if this solution is not similar to the previously stored solutions (i.e., if it does not belong to small boxes centered at other solutions, line 10), then it is regarded as a new solution and it is stored in S (line 15). Before storing a new solution, however, it is refined through Newton-Raphson method for nonlinear systems, which converges very quickly (in about 2-3 iterations) to a very accurate solution, since the starting approximate solution obtained by linear interpolation is already very close to the exact one. Newton-Raphson method is applied between lines 11-14, in which  $\Delta s^*$  is the improvement of the solution in each iteration,  $\varepsilon$  is a small threshold below which further improvements are considered negligible (stop criterion), and **F** and **J** are the constraint vector function and its Jacobian matrix, respectively, which are defined as follows:  $\mathbf{F} = [e_1, e_2, e_3]^T$  and

$$\mathbf{J} = \begin{bmatrix} \frac{\partial e_1}{\partial \alpha} & \frac{\partial e_1}{\partial \beta} & \frac{\partial e_1}{\partial z} \\ \frac{\partial e_2}{\partial \alpha} & \frac{\partial e_2}{\partial \beta} & \frac{\partial e_2}{\partial z} \\ \frac{\partial e_3}{\partial \alpha} & \frac{\partial e_3}{\partial \beta} & \frac{\partial e_3}{\partial z} \end{bmatrix}$$
(40)

where  $e_i$  are functions of  $\mathbf{s} = [\alpha, \beta, z]^T$ , defined in Eqs. (1)-(3). Since  $\mathbf{s}^*$  will change little during Newton iterations, matrix  $\mathbf{J}^{-1}$  may be computed only once before refining each solution  $\mathbf{s}^*$  (right before line 11) and kept constant in line 12, which saves computations without impeding fast convergence.



Figure 5: The two closed constraint curves for the example of Section 5. The region enclosed by each curve has been colored to better distinguish these curves under wrapping.

#### 5 EXAMPLE AND DISCUSSION

Next, the efficiency and validity of the proposed method will be demonstrated through an example with the following parameters:  $\mathbf{a_1} = [0.7, 2.45, 0]^T$ ,  $\mathbf{a_2} = [2.676, -1.379, 0]^T$ ,  $\mathbf{a_3} = [-2.161, 2.627, 0]^T$ ,  $\mathbf{b_1} = [-2.255, 1.099, 2.728]^T$ ,  $\mathbf{b_2} = [0.675, -2.347, 0.532]^T$ ,  $\mathbf{b_3} = [-1.935, -0.966, -1.953]^T$ ,  $\theta = 0$ ,  $\rho_1 = 5$ ,  $\rho_2 = 4.5$ , and  $\rho_3 = 4.631$  (assume arbitrary units of length). Our method will be run configured as follows:  $\Delta \alpha = \Delta \beta = 0.0315738$  rad,  $r_{\alpha} = r_{\beta} = 0.0347312$  rad,  $\varepsilon = 0.000001$  (threshold to stop Newton refinement).

For these parameters, the constraint curves obtained by the proposed method are shown in Figure 5 in thick black lines. As this figure shows, this example yields 18 real solutions, which are identified by numbers 1-18 and small blue circles along these curves. Note that, considering the wrapping of these curves along the lines  $\alpha = \pm \pi$  and  $\beta = \pm \pi$ , there are two closed constraint curves. The sampling method proposed in Section 4.1 is robust to the frequent case in which these curves have multiple connected components, which may pose troubles to alternative methods based on continuation for tracking and rendering these curves. In that case, continuation methods should guarantee that a continuation starts from one seed point belonging to each connected component, in order to track all components (Gomes, 2014).

The values of  $(\alpha, \beta, z)$  for these 18 solutions are shown in Table 1. To check the correctness of these solutions, these were back-substituted into Eqs. (1)-(3) to solve  $\rho_i$  from them, obtaining the original values ( $\rho_1 = 5$ ,  $\rho_2 = 4.5$ , and  $\rho_3 = 4.631$ ) with errors smaller than  $10^{-9}$  in all cases. Also, to check if the proposed method obtained all real solutions (without missing solutions), these were also obtained by other reliable methods for obtaining all solutions, which are available as software packages: CUIK (Porta et al., 2014) and Bertini (Bates et al., 2013), which use branch-and-prune and homotopy methods, respectively. Our method was coded in Java programming language, and all three methods (ours, and the CUIK and Bertini methods) were tested on a single-core 2GB RAM Ubuntu 32bit virtual machine with CUIK preinstalled on it, running on a Mac Pro with a 3 GHz 8-Core Intel Xeon E5 processor and 16 GB RAM. All three methods obtained 18 real solutions, and the solutions obtained by our method coincide with those of the other two methods up to the eighth decimal position (CUIK and Bertini work with the cosine and sine of angles intead of the angles themselves, so this comparison was made after computing the sine and cosine of  $\alpha$  and  $\beta$  for all 18 solutions displayed in Table 1).

Comparing times, our method took only 0.01 seconds to generate the curves and obtain all 18 solutions (steps 1 to 3), whereas CUIK and Bertini took 1.11 and 0.89 seconds to obtain these solutions, respectively (CUIK and Bertini were run using their default parameters). These times are averages of 10 runs for each method, with standard deviations of 0.002, 0.02 and 0.08 seconds for our method, CUIK, and Bertini, respectively. The times measured for the CUIK and Bertini methods were extracted from the output files generated by these packages, which are identified as "User time in process" in CUIK, and "Track Time" in Bertini. According to this comparison, our method is roughly 100 times faster than other methods.

Nevertheless, CUIK and Bertini have other advantages over our proposed method. For example, Bertini can obtain also all nonreal solutions, which are not physically possible but still are useful for learning how they become real when approaching singularities (Peidró et al., 2019). In particular, the example considered in this section has 10 nonreal solutions besides the 18 real ones obtained by the proposed method, and Bertini obtains all 28 solutions (real and nonreal). Regarding CUIK, it can handle degenerate cases which are otherwise elusive or very difficult to solve, e.g., when a constraint curve degenerates into a point solution, which would be overlooked by the sampling method described in Section 4.1. Finally, both CUIK and Bertini work with any parallel manipulator, whereas our method is ad-hoc for 3UPS-PU manipulators. However, we believe that this method can be easily generalized and extended to more complex manipulators, which is the topic of future work.

# Solution	α (rad)	$\beta$ (rad)	z
1	-3.074015668	2.096303267	-1.560581389
2	-2.598732611	2.977271815	-1.218233476
3	2.259863227	2.298271789	1.836517445
4	2.296596123	1.825026909	-0.312209976
5	-3.076668574	-0.285479858	1.695349818
6	-2.521795906	-0.498199503	0.138552682
7	-0.944416244	1.279997012	-5.636730120
8	-0.618751656	0.447473831	2.571815016
9	-0.573215460	0.056916560	2.345009820
10	-0.139016511	-0.862859797	2.372701275
11	0.684167157	-1.425432943	4.033720688
12	-0.056524769	2.706168583	-2.505056890
13	-1.325219478	3.042199401	0.924688594
14	2.483381960	-1.670214061	0.751169173
15	1.065465274	-0.595531432	-2.109833757
16	0.950424577	-0.014348808	-2.409395862
17	1.203800574	1.274843283	-1.554851753
18	2.911141509	0.085737211	2.937707838

Table 1: 18 real solutions for the example of Section 5.

#### 6 CONCLUSIONS

In this paper, we have presented a new method for solving the forward kinematics of 3UPS-PU parallel manipulators. The proposed method eliminates one unknown, obtaining an equation that defines a constraint curve in the plane of the remaining unknowns. This constraint curve is densely sampled and then efficiently searched to find all real solutions of the forward kinematics. This method makes use of kd trees and requires solving quartic polynomials at most. Furthermore, explicit symbolic expressions of the coefficients of these polynomials have been derived, which are parameterized in terms of all kinematic parameters. This allows us to change the value of any kinematic parameter and instantaneously recompute and update all solutions. This is not feasible in general for elimination methods, which yield highdegree polynomials that involve too large and complicated expressions to be obtained symbolically.

The proposed method is about 100 times faster than other methods, but has also a much more limited scope, since it works only for 3UPS-PU robots. However, we are currently working on the extension of this method to broader families of manipulators with more degrees of freedom, succeeding to find "constraint manifolds" (analogous to the constraint curves here analyzed) that can be efficiently sampled and searched for all real solutions of the forward kinematics. Extension of this method to the complex domain will also be explored, to find also nonreal solutions.

#### ACKNOWLEDGEMENTS

This research was funded by the Spanish government through the project DPI2016-78361-R (AEI/FEDER, UE): "Creación de mapas mediante métodos de apariencia visual para la navegación de robots", and by the Generalitat Valenciana through the project AICO/2019/031: "Creación de modelos jerárquicos y localización robusta de robots móviles en entornos sociales".

#### REFERENCES

- Bates, D. J., Hauenstein, J. D., Sommese, A. J., and Wampler, C. W. (2013). Numerically Solving Polynomial Systems with Bertini. SIAM.
- Caccavale, F., Siciliano, B., and Villani, L. (2003). The tricept robot: dynamics and impedance control. *IEEE/ASME Transactions on Mechatronics*, 8(2):263–268.
- Fattah, A. and Kasaei, G. (2000). Kinematics and dynamics of a parallel manipulator with a new architecture. *Robotica*, 18(5):535–543.
- Gomes, A. J. (2014). A continuation algorithm for planar implicit curves with singularities. *Computers & Graphics*, 38:365 – 373.
- Hosseini, M. A. and Daniali, H.-R. M. (2011). Kinematic analysis of tricept parallel manipulator. *IIUM Engineering Journal*, 12(5):7–16.
- Hu, B. (2014). Complete kinematics of a serial-parallel manipulator formed by two tricept parallel manip-

ulators connected in serials. *Nonlinear Dynamics*, 78(4):2685–2698.

- Innocenti, C. and Wenger, P. (2006). Position analysis of the RRP-3(SS) multi-loop spatial structure. J. Mech. Design, 128(1):272–278.
- Joshi, S. and Tsai, L.-W. (2003). The kinematics of a class of 3-dof, 4-legged parallel manipulators. J. Mech. Design, 125(1):52–60.
- Kolláth, L., Kureková, E., Ploskuňáková, L., and Beniak, J. (2009). Non-conventional production machines. Scientific Proceedings. SjF, STU Bratislava, pages 69– 75.
- Liu, C. H. and Hsu, F.-K. (2007). Direct singular positions of the parallel manipulator tricept. *Proceedings of the Institution of Mechanical Engineers, Part C: Journal* of Mechanical Engineering Science, 221(1):109–117.
- Lu, Y., Hu, B., and Liu, P. (2007). Kinematics and dynamics analyses of a parallel manipulator with three active legs and one passive leg by a virtual serial mechanism. *Multibody System Dynamics*, 17(4):229–241.
- Neumann, K. (1988). Robot, US4732525A.
- Peidró, A., Gil, A., Marín, J. M., and Reinoso, Ó. (2016). A web-based tool to analyze the kinematics and singularities of parallel robots. *Journal of Intelligent & Robotic Systems*, 81(1):145–163.
- Peidró, A., Marín, J. M., Reinoso, Ó., Payá, L., and Gil, A. (2019). Parallelisms between planar and spatial tricept-like parallel robots. In Arakelian, V. and Wenger, P., editors, *ROMANSY 22 – Robot Design, Dynamics and Control*, pages 155–162, Cham. Springer International Publishing.
- Peidró, A., Reinoso, O., Gil, A., Marín, J. M., and Payá, L. (2018). A method based on the vanishing of self-motion manifolds to determine the collision-free workspace of redundant robots. *Mechanism and Machine Theory*, 128:84 – 109.
- Pendar, H., Roozbehani, H., Sadeghian, H., and Zohoor, H. (2008). Singularity analysis of a 3dof parallel manipulator using infinite constraint plane method. *Journal* of Intelligent and Robotic Systems, 53(1):21–34.
- Porta, J. M., Ros, L., Bohigas, O., Manubens, M., Rosales, C., and Jaillet, L. (2014). The cuik suite: Analyzing the motion closed-chain multibody systems. *IEEE Robotics Automation Magazine*, 21(3):105–114.
- Press, W. H., Teukolsky, S. A., Vetterling, W. T., and Flannery, B. P. (2007). Numerical Recipes: The Art of Scientific Computing (3rd Edition). Cambridge University Press.
- Siciliano, B. (1999). The tricept robot: Inverse kinematics, manipulability analysis and closed-loop direct kinematics algorithm. *Robotica*, 17(4):437–445.
- Simionescu, P. A. (2014). Computer-aided graphing and simulation tools for AutoCAD users. CRC Press.