DOI: 10. 11835/j. issn. 2096-6717. 2024. 124

开放科学(资源服务)标识码 OSID:

FURobot: A software control platform for construction robots for large-scale construction

LU Ming, *Philip F. YUAN*

(College of Architecture and Urban Planning, Tongji University, Shanghai 200092, P. R. China)

Abstract: The advent of parametric design has resulted in a marked increase in the complexity of building. Unfortunately, traditional construction methods make it difficult to meet the needs. Therefore, construction robots have become a pivotal production tool in this context. Since the arm span of a single robot usually does not exceed 3 meters, it is not competent for producing large-scale building components. Accordingly, the extension of the robot's working range is often achieved by external axes. Nevertheless, the coupling control of external axes and robots and their kinematic solution have become key challenges. The primary technical difficulties include customized construction robots, automatic solutions for external axes, fixed axis joints, and specific motion mode control. This paper proposes solutions to these difficulties, introduces the relevant basic concepts and algorithms in detail, and encapsulates these robotics principles and algorithm processes into the Grasshopper plug-in commonly used by architects to form the FURobot software platform. This platform effectively solves the above problems, lowers the threshold for architects, and improves production efficiency. The effectiveness of the algorithm and software in this paper is verified through simulation experiments.

Keywords: construction robots; customization; construction; robotics; kinematics; software

FURobot:面向大尺度建造的建筑机器人软件控制平台

陆明,袁烽

(同济大学 建筑与城市规划学院,上海 200092)

摘　要:随着参数化设计的兴起,建筑构件的复杂性不断增加,传统的建造方法难以满足需求,在此背景下,建筑机器人成为重要的生产工具。由于单个机器人的臂展通常不超过3 m,无法胜任大尺度建筑构件的生产任务,因此,常通过外部轴来扩展机器人的工作范围。然而外部轴与机器人的耦合控制及其运动学解算成了关键挑战,主要技术难点包括:定制化建筑机器人、自动求解外部轴、固定轴关节以及特定的运动模式控制。提出针对这些难点的解决方案,详细介绍相关的基本概念和算法,并将这些机器人学原理与算法流程封装到建筑师常用的Grasshopper插件中,形成FURobot软件平台。该平台不仅有效解决了上述问题,还降低了建筑师的使用门槛,提升了生产效率。最后,通过仿真试验验证了算法和软件的有效性。

关键词:建筑机器人;定制化;建造;机器人学;运动学;软件

中图分类号:TU689;TU17　　**文献标志码:**A　　**文章编号:**2096-6717(2025)05-0001-11

Received: 2024-09-29

Foundation items: National Key R & D Program of China (Nos. 2023YFC3806900, 2022YFE0141400)

Author brief: LU Ming (1984-), PhD candidate, main research interest: intelligent construction, E-mail: 2110282@tongji.edu.cn.

Philip F. YUAN (corresponding author), PhD, professor, E-mail: philipyuan007@tongji.edu.cn.

1 Introduction

The concept of mass customization originated in the 1970s with the aim of improving the efficiency of mass production. However, due to the different shapes of construction components, it is impossible to rely on traditional molds for production. Upon entering the 1980s, CAD(Computer-Aided Design) was gradually applied to the field of architectural design[1]. Notably, irregular and personalized design needs promoted the development of customized building components. CNC(Computer Numerical Control) machine tools and laser cutting technology were introduced from the manufacturing industry to the construction field[2], technically solving the problem of mass-customized production. However, due to the lagging level of automation and information construction, the actual realization of mass-customized construction is still in the conceptual stage[3]. Recently, industrial robots have been gradually applied to the production line of customized components, dramatically improving the flexibility and adaptability of the processing process. Moreover, it made the factory production of non-standard components possible and promoted the application of intelligent construction equipment in on-site construction[4].

1.1 Features of construction robots

We call intelligent equipment related to construction work construction robots. Construction robots for mass-customized component production have the following characteristics:

(1) Large stroke

The primary demand of the construction industry is to develop building components. The size of these components is usually between 1 meter and 10 meters, so the robot is required to have an extensive range of motion. Typically, a robot arm is used to process the components, and the choice of robot arm should be adjusted according to the size of the component. Since the arm span of most robots does not exceed 3 meters, monorail external axes or mobile robots are often used in larger construction tasks to extend the working range[5].

(2) Heavy load

Construction robots are usually equipped with corresponding end-effectors at their ends when performing different processes[6]. Even the smallest plastic printing end-effector weighs about 10 kg, while the end-effector of a concrete printing robot usually weighs about 20 kg. Additionally, the robot itself usually weighs between 1 and 2 tons. Therefore, its external axis tracks must be able to carry heavy loads.

(3) High precision

In the production of large-scale components, the accuracy requirement is usually plus or minus 0.5 mm. That is, the error must be controlled within 1 mm. Especially in the field of wood processing[7-8], the accuracy requirements of mortise and tenon structures are more stringent.

This paper focuses on how to better control the movement of construction robots under large strokes.

1.2 External axis equipment

External axes are widely used in the production of customized mass-produced building components (Fig. 1). Compared with mobile platform robots, external axes have higher operating and positioning accuracy. The robot can also be fixed on the external axis in different installation poses, such as upside-down or horizontal installation. Thus, the combination of industrial robots and external axes is often used to produce large-scale prefabricated building parts, such as curved facade panels for 3D printed buildings, customized furniture, curved wooden beams for subtractive processing, and the construction of curved brick walls[9] and concrete 3D printed beams[10].

Mobile robots have also been widely used in fields such as bricklaying and wooden construction[11]. Mobile robots have a larger range of motion and broader applicability than external axis devices. They are also particularly suitable for use in construction sites with complex terrain. However, the stability and accuracy of mobile robots are not as ideal as those of external axes. Usually, mobile platforms need to be equipped with motion capture positioning systems and SLAM(Simultaneous Localization and Mapping) algorithms to ensure their positioning accuracy.

By using monorail external axes or gantry external axes, robots can print building components and entire buildings[12]. The size and number of external

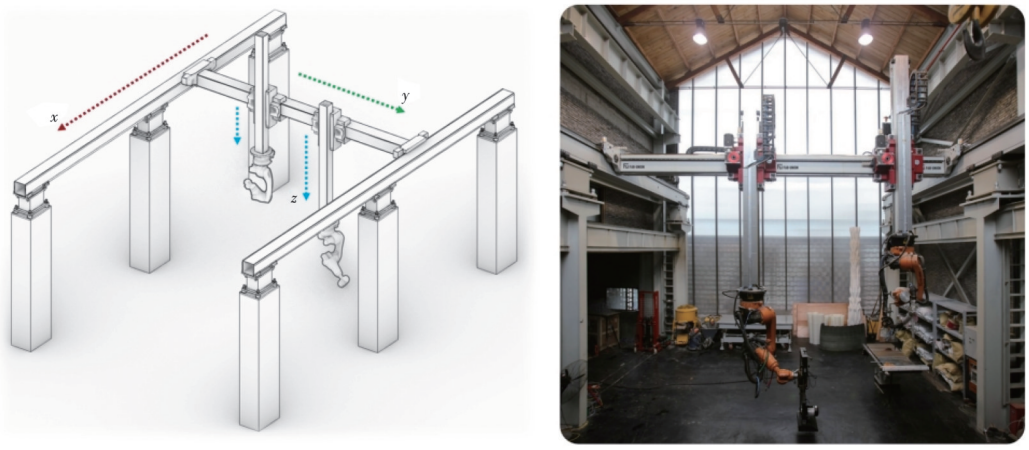


Fig. 1　The gantry external axis system for batch prefabricated components (Shanghai FabUnion Technology Co., Ltd., 2015)

axes required will also vary depending on the construction site. Similar to external axes, the principle of the mobile platform is to expand its range of motion by changing the robot's root coordinate frame position.

After the robot hardware is ready, the building components designed by the architect (such as facade wall panels or curved wooden beams, etc.) must be generated into robot executable programs through software. Finally, the robot should process the product to complete the process. The whole process is shown in Fig. 2

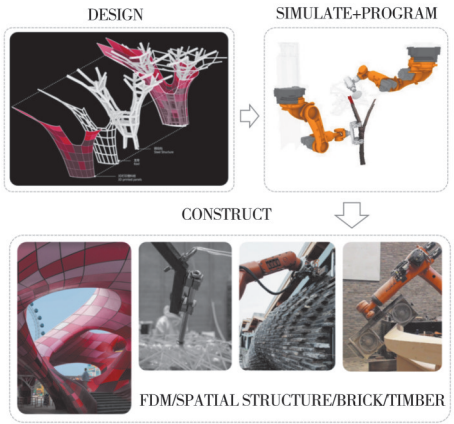


Fig. 2　Building components from design to construction

2　Related work

The simulation and offline program generation software was called construction robot control software, which combined automation technology, robotics, and architectural knowledge to manage and control construction robot operations. At present, robot software control platforms are mainly divided into two categories: original platform and general robot platform.

Original platforms, such as ABB RobotStudio and KUKA SimPro, provide the most complete support for their brand. General robot platforms, such as RoboDK, although supporting a variety of robot types, usually only offer basic functions.

To fill the gap between this software and the needs of architectural designers, software for architects has emerged, such as KUKA|prc[13], Robots, and Machina[14]. Notably, these tools enable architects to complete most construction tasks efficiently. They also generate motion instructions that robots can execute through the building component models and customized process flows provided by the architects. These instructions can drive the robot in an offline program or real-time control mode. At present, the mainstream construction robot control software includes KUKA|prc and HAL.

Most existing software does not integrate process packages. Thus, architects often need to create Grasshopper files from scratch to complete the robot's motion path, which is a time-consuming and complicated task for architects[15]. Due to the disciplinary background of architects, the principles of robotics often become an obstacle to using these devices. As such, FURobot[16] solves this problem by integrating commonly used process packages and programming the methods introduced in this article, which are applicable to almost all construction processes.

The differences between the software are listed in table 1.

Table 1 Existing robotic arm programming software

Name	Obstacle avoiding	Communication	Tech. package	Robot brand
KUKA|prc		KUKA		KUKA
HAL			Hot-wire	KUKA/ABB/UR/Staubli/Fanuc
FURobot	Has	KUKA/ABB/UR	FDM/Spatial structure/Timber/Fiber	KUKA/ABB/UR/Step/Fanuc/Staubli
Robots				KUKA/ABB/UR/Staubli/Frank Emika
Machina		KUKA/ABB/UR		KUKA/ABB/UR
Robot-Components		ABB		ABB

2.1 Difficulties in robot simulation

From the above construction workflow, the software simulation and program file generation stages face the following difficulties:

(1) Quickly establish the series structure

The URDF(Unified Robot Description Format) and DH(Devavit-Hartenberg)[17] parameter were used to define robot kinematic models. However, these have a high threshold for architects to use. From the production tasks of mass-customized building components, there are many combinations of external axes and robotic arms, including single-axis, multi-axis, rotary external axes, and linear track external axes. In this case, architects need to build Grasshopper components for each external axis configuration for motion simulation.

(2) Solving for external axis motion

When using external axes to produce large building components, writing offline programs to control the movement of robots is one of the most time-consuming steps for architects. Currently, the software available on the market cannot automatically generate motion programs for external axes, and architects need to set them manually. This manual setting makes it challenging to ensure continuous and smooth movement of external axes and may cause instantaneous acceleration or deceleration. Consequently, this increases labor costs and may cause damage to equipment.

(3) Lock the robot or external axis joints

During use, the user may need to fix the joint of a certain external axis at a certain value to reduce the movement of the external axis. In addition, the user may need to lock a robot arm joint. For example, they may fix the specific position of the external axis during certain operations to avoid collisions.

(4) Adapt to different task requirements

Different tasks have different requirements for the TCP(Tool Center Point) target frame. For example, in bricklaying tasks, the bricks need to be placed in the exact target pose in addition to reaching the target position. In wood milling, only the position and orientation of the milling cutter need to be aligned. If all poses are still required to be fully aligned, it will cause unnecessary joint movements and add unnecessary complexity.

The research direction of this paper focuses on how to generate motion paths that are more suitable for construct component processing. Importantly, this research is at the intersection of robotics and specific component processing technology. A more optimized motion path is a common requirement for most processes, but most of the current building robot control software for mass-customized components is lacking.

3 Twist-based methods

The existing software platform divides external axes and robot joints into two categories. The robot joints can be solved by inverse kinematics. At the same time, the movement of the external axes requires the user to manually set it under specific instructions (such as linear instructions). Finally, it can perform forward kinematics simulation through simple linear interpolation. Note that this method cannot achieve the automatic solution of external axis motion.

This paper first adopts the kinematic iterative solution method based on the twist[18-20]. The iterative solution has the following advantages: (1) It can solve any serial structure's forward and inverse kinematics problems; (2) Modifying the Jacobian matrix can flexibly achieve target control, which is challenging in an analytical solution. We combined the external axis with the industrial robot into a multi-axis serial robot. We used the iterative IK(inverse kinematics)

method to obtain all robot and external axis joint values. By improving the Jacobian matrix in the iterative algorithm, the locking of specific joints was achieved, and the target TCP(tool center coordinate plane) was aligned according to different task requirements. Finally, this method successfully solves the problem of solving external axis motion.

Before introducing the method in detail, this article will briefly introduce the relevant basic concepts.

3.1 Basic concepts of robot kinematics

This article defines the robot joints by using rotation. The use of rotation does not require the robot's joint coordinate frame to be set in advance, making it easier to define the robot's joints. The most basic robot joints are revolute joints and linear joints[21]. Furthermore, industrial robots are generally composed of revolute joints, and external axes are usually composed of translation joints.

Whether rotating or moving a joint, it can be described by a screw $\boldsymbol{\mathcal{V}}$, where $\boldsymbol{\omega}$ represents the angular velocity of the rotational joint (rad/s), and $\boldsymbol{v}$ is the velocity of the virtual point that coincides with the origin of the reference coordinate frame {s} in the reference coordinate frame {s} when the rotational joint rotates (m/s).

$$\boldsymbol{\omega}=\left[\omega_x,\omega_y,\omega_z\right]^{\mathrm{T}}, \tag{1}$$

$$\boldsymbol{v}=\left[v_x,v_y,v_z\right]^{\mathrm{T}}, \tag{2}$$

$$\boldsymbol{\mathcal{V}}=[\boldsymbol{\omega},\boldsymbol{v}]^{\mathrm{T}} \tag{3}$$

(1) Prismatic joint

Because it is a linear motion, $\boldsymbol{\omega}$ is $[0,0,0]^{\mathrm{T}}$, and $\boldsymbol{v}$ represents the unit vector of the linear motion joint. For example, $\boldsymbol{v}=[1,0,0]^{\mathrm{T}}$ represents a joint moving along the x-axis.

(2) Revolute joint

$\boldsymbol{\omega}$ represents the angular velocity of the revolute joint, and $\boldsymbol{v}$ is the velocity of a virtual point in the reference coordinate frame {s} that coincides with the origin of the reference coordinate frame {s} when the revolute joint rotates. For example, defining a single-track external axis and a 6-axis robot requires a total of 7 joints. In addition to defining the robot joints, you also need to define the robot's root coordinates, flange coordinate frame, and end-effector coordinate frame.

(3) Root coordinate frame

The root coordinate frame determines the position and pose of the entire robot in space[22]. When merging robots, you need to set the root coordinates for all except the first to the world coordinate frame.

(4) Flange coordinate frame

The flange coordinate frame is the coordinate frame on the end joint of the robot when all joint values are zero. It is described relative to the world coordinate frame. The origin of this coordinate frame is generally the center point of the flange face connected to the end effector.

(5) End-effector coordinate frame

The coordinate frame of the end-effector is described relative to the flange coordinate frame.

3.2 Connecting external axes to the robot

Before connecting multiple robots in series to form a new single robot, it is necessary to perform a rigid body transformation on the rods of all robots except the first one. The goal is to place the combined robot in the new correct position. Starting from the second robot, all its links need to implement the rigid body transformation T_{01}. T_{01} is the rigid body transformation from the base of the first robot to its flange coordinate frame. Furthermore, T_{12} is the rigid body transformation from the base of the second robot to its flange coordinate frame, and so on.

In addition to the robot links, all joints except the first robot need to be transformed. For example, the first joint rotation of the second robot is $\boldsymbol{\mathcal{V}}_1$. Through the adjoint matrix $\boldsymbol{A}_{\mathrm{d}}(T_{01})$, a new joint screw $\boldsymbol{\mathcal{V}}_1'$ is generated.

$$\boldsymbol{\mathcal{V}}_1'=\boldsymbol{A}_{\mathrm{d}}(T_{01})\boldsymbol{\mathcal{V}}_1 \tag{4}$$

$$T_{01}=\begin{bmatrix}\boldsymbol{R} & \boldsymbol{p}\\ 0 & 1\end{bmatrix},\boldsymbol{R}\in S_{\mathrm{O3}},\boldsymbol{p}\in\mathbb{R}^3 \tag{5}$$

Where S_{O3} is a special orthogonal group in three dimensions. $\mathbb{R}^3$ is the vector space of three real values. The definition of the adjoint matrix is

$$\boldsymbol{A}_{\mathrm{d}}(T_{01})=\begin{bmatrix}\boldsymbol{R} & 0\\ [\boldsymbol{p}]\boldsymbol{R} & \boldsymbol{R}\end{bmatrix} \tag{6}$$

Where $\boldsymbol{R}$ is the rotation matrix, $\boldsymbol{p}$ is the displacement vector, and $[\boldsymbol{p}]$ is the skew-symmetric matrix of $\boldsymbol{p}$. In this way, all joints of the robot that are to be combined can be transformed into the same world coordinate frame for description.

In addition to joints and links, each robot has a fixed base. The base is the only part of all parts that does not move with the joints. When connecting

multiple robots, the robot's flange part and the next robot's fixed base are combined into one.

Each robot has a flange coordinate frame at its end, which is calculated by formula (7).

$$T_{\text{flange}} = T_{\text{flange1}} T_{\text{flange2}} \cdots T_{\text{flange}N} \tag{7}$$

All flange coordinate frames are merged into the final robot flange coordinate frame T_{flange}.

3.3 Classification of positioner

A positioner is a two-degree-of-freedom fixture that can rotate horizontally and vertically (such as KUKA DKP-400) and is usually used to cooperate with robots for milling operations. In general, a positioner is also an external axis device. However, in FURobot, all robots are currently serial structure robots. Through the combined external axes, their movement will affect the other robots or be affected by the movement of different robots. Consequently, the movement of the external axis will directly affect the end position of the robot. However, the positioner is not affected by the robot's movement, and the positioner's movement will not affect other robots. Therefore, in this case, the positioner is classified as an independent robot in the FURobot, not an external axis device.

3.4 Inverse kinematics solution

Inverse kinematics is usually solved by analytical solution or numerical iteration. The analytical solution is more commonly used for mature industrial 6-axis robots. Its advantages are fast calculation speed and accurate results. Its disadvantage is that it needs to find a solution formula[23]. As long as the robot structure changes, the formula needs to be re-derived. The advantage of the numerical iteration method is that it can be solved on a serial robot of any structure. The disadvantage is that the speed is slightly slower. Since this study is aimed at general serial structure robots, it is impossible to obtain the kinematic analytical solution of each robot structure. Thus, the numerical iteration method is selected for a solution, usually using the Newton-Raphson method.

$$\mathrm{d}t = \boldsymbol{J}_{\text{b}}^{-1}(t)\,\mathrm{d}x \tag{8}$$

Where $\mathrm{d}t$ is the increment of the joint value calculated after each iteration, $\boldsymbol{J}_{\text{b}}$ is the body Jacobian matrix of the robot (with the end-effector as the reference frame). $\mathrm{d}x$ is the difference between the target pose and the current pose of the robot end (described in a twist).

Since we define the joint using the spatial Jacobian matrix (with the world coordinate frame as the reference frame) instead of the body Jacobian, considering the transformation T_{root} of the base coordinate frame (i. e. the root coordinate frame), we get the columns $\boldsymbol{J}_{\text{si}}$ of the Jacobian of the joint velocity to the spatial velocity in the world coordinate frame.

$$\boldsymbol{J}_{si} = \boldsymbol{A}_{\text{d}}\left(T_{\text{root}} \mathrm{e}^{[\boldsymbol{S}_1]\theta_1} \cdots \mathrm{e}^{([\boldsymbol{S}_{i-1}])\theta_{i-1}} \right) \boldsymbol{S}_i \tag{9}$$

Here $\boldsymbol{J}_{si}$ is the i-th column of the spatial Jacobian, $\boldsymbol{S}_i$ is the twisted form of the i-th joint, and θ_i is the value of the i-th joint. We calculate the spatial Jacobian and convert it to the body Jacobian:

$$\boldsymbol{J}_{\text{b}} = \boldsymbol{A}_{\text{d}}\left(T_{\text{s}}^{\text{b}}\right)\boldsymbol{J}_{\text{s}} \tag{10}$$

Where T_{s}^{b} represents the rigid body transformation from the robot's end (flange or end-effector) coordinate frame to the robot's root coordinate frame at the current robot joint values.

Substituting $\boldsymbol{J}_{\text{b}}$ in formula (8) yields

$$\mathrm{d}t = \left(\boldsymbol{A}_{\text{d}}\left(T_{\text{s}}^{\text{b}}\right)\boldsymbol{J}_{\text{s}}\right)^{-1}\mathrm{d}x \tag{11}$$

The above method can also be used to obtain the increment of joint value and perform iterative calculations.

A singular pose is a pose of the robot when the value of the determinant of the Jacobian of the robot is close to 0. When the robot reaches this pose or is close to this pose, the Newton-Raphson method cannot guarantee convergence[24]. Therefore, we use two methods to improve it:

(1) Segmented approximation solution

When the robot executes a straight-line instruction, the robot often reaches a singular pose during the entire operation, especially a long-distance straight-line instruction. We use the dichotomy method to find the non-convergent point near the singular pose, the closest and convergent pose to this pose, and calculate the inverse kinematics solution. Increasing the sampling points can ensure the stability of the numerical iteration. In contrast, the position that will not produce divergence is skipped in the numerical iterative calculation to improve the calculation efficiency.

(2) More stable methods

For example, the LM(Levenberg-Marquardt) method[25] can be improved. After testing, the LM method can significantly improve the stability and accuracy of numerical calculations.

3.5 Fixed joint values

In some cases, fixing a joint axis in a specific position is necessary while allowing other joints to continue to move. For example, the robot is usually placed on a monorail external axis when printing large building components. In this case, it is common to move the robot to a specific position on the external axis and then fix the position of the external axis before starting printing. Doing so may avoid errors and jitter caused by the movement of the external axis. Notably, the way to achieve a fixed external axis is to move the external axis to the predetermined position and then perform an inverse kinematics solution on the robot. However, the limitation of this method is that the movement of the external axis must be determined manually, reducing the work efficiency. Furthermore, this method cannot fix the robot joints (not the external axis), limiting the flexibility of operation.

Since this paper uses a numerical iteration method to solve inverse kinematics, when a joint value needs to be fixed (such as a moving joint of an external axis), it can be achieved by modifying the Jacobian of the robot. Specifically, all the Jacobian matrix column elements corresponding to the "fixed joint" were set to 0. This is because the i-th column of the Jacobian matrix represents the mapping of the i-th joint velocity to the spatial velocity. If all the elements in this column are 0, it means that the joint no longer contributes to the overall spatial velocity. Using this modified Jacobian, the value of the i-th joint will remain unchanged during the iterative process of solving inverse kinematics, thereby achieving joint fixation.

For example, for a 6-axis robot, the third joint can be fixed by setting the third column of its Jacobian matrix to 0 and keeping the other columns unchanged.

$$\boldsymbol{J}=\begin{bmatrix} j_{11} & j_{12} & 0 & \cdots & j_{16} \\ j_{21} & j_{22} & 0 & \cdots & j_{26} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ j_{61} & j_{62} & 0 & \cdots & j_{66} \end{bmatrix} \tag{12}$$

Therefore, the effect of controlling the joint motion range can also be achieved by scaling the values of the columns of the Jacobian matrix.

3.6 Align end-effector mode

When printing large building components and tilting is required, the end-effector of the 6-axis robot does not need to be wholly aligned with the target frame. It is only necessary to ensure that the normal of the end-effector is aligned with a specified axis of the target frame (Fig. 3). This not only simplifies motion control but also reduces unnecessary rotation adjustments while improving printing efficiency.

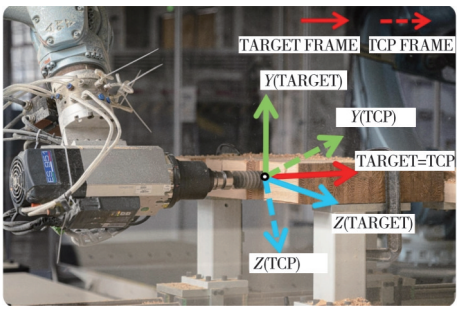


Fig. 3　Most woodworking operations only require aligning the normal of the end-effector frame with the normal of the target frame

The above alignment requirements can be achieved by modifying the inverse or pseudo-inverse of the Jacobian. For example, in a printing task, if the normal direction of the end-effector is aligned with the x-axis of the TCP coordinate frame and the normal vector of the target frame is also defined as the x-axis direction, we first calculate the Jacobian. When it is necessary to maintain the position and only align the normal vector, the corresponding screw velocity row needs to be set to zero. For example, when the normal direction is the x-axis, set the first row of the Jacobian matrix to zero; if the normal direction is the y-axis, set the second row to zero; if the normal direction is the z-axis, set the third row to zero. If only the position needs to be aligned, set all the first three rows of the Jacobian matrix to zero.

$$\boldsymbol{J}=\begin{bmatrix} 0 & 0 & 0 & \cdots & 0 \\ 0 & 0 & 0 & \cdots & 0 \\ 0 & 0 & 0 & \cdots & 0 \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ j_{51} & j_{52} & j_{53} & \cdots & j_{56} \\ j_{61} & j_{62} & j_{63} & \cdots & j_{66} \end{bmatrix} \tag{13}$$

When using the Newton-Raphson method, it is sometimes necessary to calculate the pseudo-inverse of the Jacobian, or you can directly set the corresponding columns in the pseudo-inverse to zero[26].

For example, when the normal direction is the x-axis, set the first column of the pseudo-inverse to zero; if the normal direction is the y-axis, set the second column to zero; when the normal direction is the z-axis, set the third column to zero.

During the production of building components by a robotic arm, the coordinate frame of the end-effector does not have to be completely aligned with the target frame. This is especially true in additive and subtractive applications such as 3D printing and wood processing, which are usually sufficient to align a particular axis (such as the normal of the end-effector). As shown in Fig. 4, position alignment can be selected when there is no strict requirement for the pose. This method reduces the range of motion of the robot joints and increases the redundant degrees of freedom, which can better realize the obstacle avoidance function.

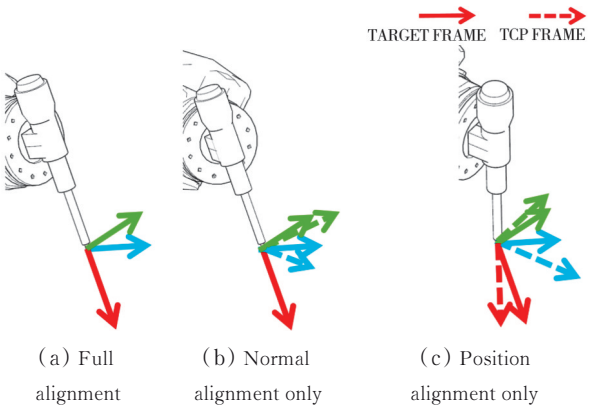


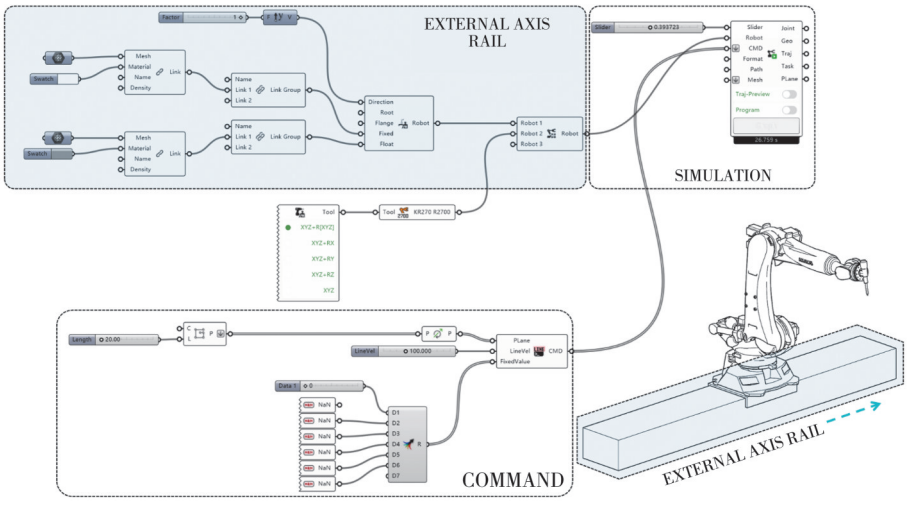
Fig. 4 Three alignment modes

4 External axis experiment

We tested the above method using the FURobot software and performed a simple simulation motion demonstration by integrating a monorail external axis and a 6-axis robot.

In this example, we used the KUKA 6-axis robot KR6-R900 for demonstration. The KR6-R900 robot component has already been built into FURobot. Thus, there is no need to re-customize the robot.

Since the robot is connected to the external axis to form a series structure, we used the robot series function of FURobot to combine the monorail external axis with the 6-axis robot. The creation of the external axis is similar to that of the custom robot. We also used the preset component to create the external axis, as shown in Fig 5.



Note: External axis rail group: Defining external axes; Command group: Define the list of instructions that the robot needs to complete; Simulation group: Simulate robot motion and generate offline programs.

Fig. 5 FURobot combines external axes with 6-axis KUKA robots

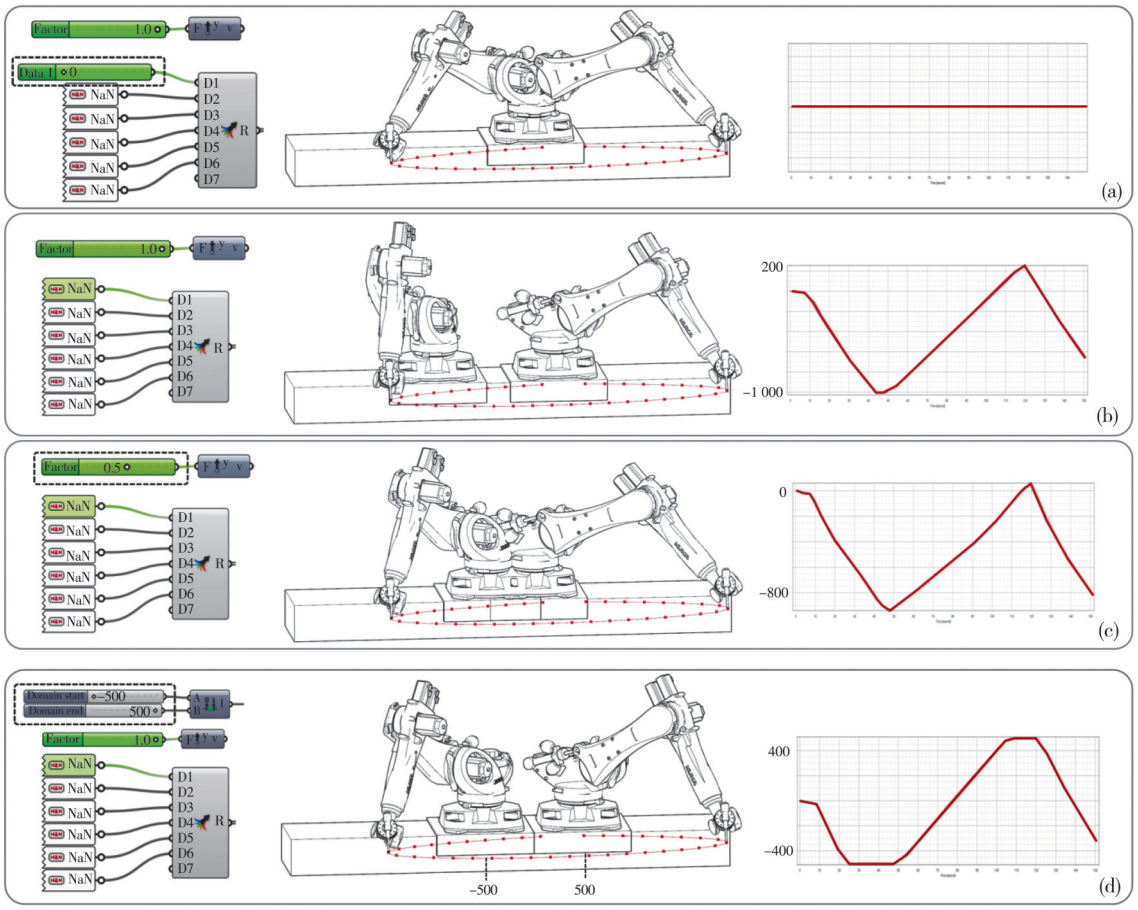
External axes can be set to free motion or fixed to a specific value. As shown in Fig 6, when joint 1 (i. e., the single-track external axis) is set to a fixed value of 0, the linear motion of the external axis is locked. In contrast, if joint 1 is set to NaN(Not a Number), which means that its position limitation is not restricted, the joint of the external axis will automatically calculate the appropriate position based on the target frame.

Similarly, other joint values (including rotational joints) can be locked. Since locking a joint reduces the degree of freedom, verifying that the inverse kine-

matics solution still meets the requirements is necessary.

We define the external axis joint and specify its velocity, which defaults to 1 m/s. This velocity value affects the robot Jacobian, which in turn affects the inverse kinematics solution through iterative calculations. Notably, the larger the velocity, the greater the impact on the end velocity when the robot moves at unit velocity on this joint. When the velocity value is large, the iterative algorithm increases this joint's motion due to the high velocity, and the optimization algorithm tends to prioritize adjusting more effective joint values.

It can be clearly seen in Fig. 6 that when the speed is 0.5 m/s, the movement range of the single track outer axis is significantly smaller than when the speed is 1 m/s. Similarly, changing the speed of other joints also affected their movement range. With this function, we can effectively limit the range of motion of the joints.



Notes: (a) indicates that the value of the external axis is fixed, and in automatic mode; (b) the value of the external axis is not locked (set to NaN as shown in the figure); (c) in automatic mode, the speed of the linear joint is scaled to reduce (set to 0.5 as shown in the figure) or increase the movement amplitude of the linear joint; (d) the joint motion range is set on the linear motion joint of the external axis. When the range is exceeded, the joint is fixed at the maximum or minimum value as in the method of adjusting the Jacobian matrix in this paper. In doing so, the robot can move strictly within the joint setting range.

Fig. 6　Lock mode

In addition, by adjusting the alignment mode of the end-effector coordinate frame and the target coordinate frame, the overall movement range of the robot can be further reduced or increased.

5　Conclusion and outlook

This paper introduces the complete workflow of construction robot design and processing and discusses the relevant technical difficulties. These difficulties include defining robot joints in a simulation environment, assembling robots, and the basic principles of kinematic iterative solutions. In response to these challenges, this paper introduces in detail the methods of path generation in different motion modes, as

well as the principles and techniques of how to lock joint values and adjust the range of joint motion. The generation process of robot motion programs is simplified through these methods, and the effects that other software cannot achieve are also achieved. This platform can help architects quickly control construction robots to process building components because there is no need to specify external axes, significantly improving manual work efficiency. Simultaneously, the new kinematic algorithm reduces the range of motion of robot joints, making the robot motion smoother. Likewise, because this software platform is based on Grasshopper and Rhino, it is more friendly to architectural designers. Thus, the above functions are of practical significance for large-scale external axis robots in construction sites.

Finally, we demonstrated the effect of the algorithm in a simulation environment through the self-developed FURobot robot software platform.

Looking to the future, research can further combine mobile platform robot technology, such as wheeled robots and drones, for the construction of an extensive range of building components. These mobile platforms can also provide greater freedom of movement, especially in complex and large-scale construction tasks. By combining fixed robots with mobile platforms, future construction robot systems will be able to respond to on-site environments more flexibly and expand their scope of application. As such, this will push the construction industry's automation and intelligence level to a new level.

References

[1] EASTMAN C, TEICHOLZ P, SACKS R, et al. BIM handbook: A guide to building information modeling for owners, managers, designers, engineers, and contractors [M]. Wiley Publishing, 2008.

[2] FABRICATE G, KOHLER M, LANGENBERG S, et al. FABRICATE: Negotiating design & making [M]. London: UCL Press, 2017.

[3] LARSEN M S S, LINDHARD S M, BRUNOE T D, et al. Mass customization in the house building industry: Literature review and research directions [J]. Frontiers in Built Environment, 2019, 5: 115.

[4] YUAN P F. Launch editorial [J]. Architectural Intelligence, 2022, 1: 1.

[5] PUZATOVA A, SHAKOR P, LAGHI V, et al. Large-scale 3D printing for construction application by means of robotic arm and gantry 3D printer: A review [J]. Buildings, 2022, 12(11): 2023.

[6] FIRTH C, DUNN K, HAEUSLER M H, et al. Anthropomorphic soft robotic end-effector for use with collaborative robots in the construction industry [J]. Automation in Construction, 2022, 138: 104218.

[7] CHAI H, SO C, YUAN P F. Manufacturing double-curved glulam with robotic band saw cutting technique [J]. Automation in Construction, 2021, 124: 103571.

[8] CHAI H, WAGNER H J, GUO Z X, et al. Computational design and on-site mobile robotic construction of an adaptive reinforcement beam network for cross-laminated timber slab panels [J]. Automation in Construction, 2022, 142: 104536.

[9] KHOSHNEVIS B. Automated construction by contour crafting: Related robotics and information technologies [J]. Automation in Construction, 2004, 13(1): 5-19.

[10] WU H, LI Y, XIE X J, et al. Structural performance-based 3D concrete printing for an efficient concrete beam [M]//Design for Rethinking Resources. Cham: Springer International Publishing, 2023: 343-354.

[11] TIRYAKI M E, ZHANG X, PHAM Q C. Printing-while-moving: A new paradigm for large-scale robotic 3D printing [C]//2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). November 3-8, 2019. Macau, China. IEEE, 2019: 2286-2291.

[12] NEMATOLLAHI B, XIA M, SANJAYAN J. Current progress of 3D concrete printing technologies [C]//Proceedings of the International Symposium on Automation and Robotics in Construction (IAARC), IAARC Publications, June 28-July 1, 2017.

[13] MUNZ H, BRAUMANN J, BRELL-COKCAN S. Direct robot control with mxAutomation: A new approach to simple software integration of robots in production machinery, automation systems, and new parametric environments [M]//Robotic Fabrication in Architecture, Art and Design 2016. Cham: Springer International Publishing, 2016: 440-447.

[14] Jose Luis García del Castillo y López. Machina.NET: A library for programming and real-time control of industrial robots [J]. Journal of Open Research Software, 2019, 7(1): 27.

[15] MENGES A. Material computation: Higher integration in morphogenetic design [J]. Architectural Design, 2012, 82(2): 14-21.

[16] LU M, ZHU W R, YUAN P F. Toward a collaborative robotic platform: FUROBOT [M]//Architectural Intelligence: Selected Papers from the 1st International Conference on Computational Design and Robotic Fabrication (CDRF 2019), 2020: 87-101.

[17] DENAVIT J, HARTENBERG R S. A kinematic notation for lower-pair mechanisms based on matrices [J]. Journal of Applied Mechanics, 1955, 22(2): 215-221.

[18] BROCKETT R W. Robotic manipulators and the product of exponentials formula [C]//Mathematical Theory of Networks and Systems: Proceedings of the MTNS-83 International Symposium Beer Sheva, Israel, June 20-24, 1983: 120-129.

[19] BROCKETT R W. Asymptotic stability and feedback stabilization [J]. Differential Geometric Control Theory, 1983, 27(1): 181-191.

[20] LYNCH K M, PARK F C. Modern robotics [M]. Cambridge, UK: Cambridge University Press, 2017.

[21] SPONG M W, HUTCHINSON S, VIDYASAGAR M. Robot modeling and control [M]. Second Edition. Wiley, 2020.

[22] CRAIG J J. Introduction to robotics: Mechanics and control [M]. Boston, MA, United States: Addison-Wesley Longman Publishing Co., Inc., 1989.

[23] LENARČIČ J. An efficient numerical approach for calculating the inverse kinematics for robot manipulators [J]. Robotica, 1985, 3(1): 21-26.

[24] NAKAMURA Y, HANAFUSA H. Inverse kinematic solutions with singularity robustness for robot manipulator control [J]. Journal of Dynamic Systems, Measurement, and Control, 1986, 108(3): 163-171.

[25] LEVENBERG K. A method for the solution of certain non-linear problems in least squares [J]. Quarterly of Applied Mathematics, 1944, 2(2): 164-168.

[26] SICILIANO B, KHATIB O. Springer handbook of robotics [M]. Springer-Verlag, 2008, 2: 15-35.

（编辑　胡英奎）