Original article

IJMEE

International Journal of Mechanical Engineering Education
2020, Vol. 48(3) 197–220
© The Author(s) 2018
Article reuse guidelines:
sagepub.com/journals-permissions
DOI: 10.1177/0306419018807376
journals.sagepub.com/home/ijj
SAGE

Robotic kinematics teaching system with virtual reality, remote control and an on–site laboratory

Xiao Xu, Peisen Guo, Jingmei Zhai and Xianwen Zeng

Abstract
Given the ever-growing need for robotics learning, an experimental teaching model was developed in combination with virtual reality, remote control and an on-site laboratory to support robotic kinematics teaching. Based on the Browser/Server mode, a kinematics simulation platform of a 6-DOF articulated robot and a 4-DOF SCARA robot was established. Furthermore, a Client/Server mode SCARA robot's remote-control system was developed along with a simulation with the same functionality and interface. Experiments with virtual reality simulation can eliminate the safety concerns associated with real-world robot operations. Test resources can be web-shared for wider audiences. Remote-controlled experiments could operate the robot off-site and achieve the same results as on-site operations. Virtual reality and remote-controlled experiments could also alleviate the shortages of lab time, space, equipment, staff and teaching time. Students could choose one of the three modes to examine the effectiveness of their study, or they could first use simulations and later operate robots on-site or remotely and obtain real-world experimental results. The various testing combinations available to learners would improve the flexibility, effectiveness and safety of experiments and would meet various learners' needs.

Keywords
Robotic kinematics, experiment teaching, virtual reality, remote control, on-site laboratory

School of Mechanical and Automotive Engineering, South China University of Technology, Guangzhou, China

Corresponding author:
Jingmei Zhai, School of Mechanical and Automotive Engineering, South China University of Technology, China.
Email: mejmzhai@scut.edu.cn

Introduction

China has become the largest global market for industrial robots and urgently needs robotics professionals. Universities and technology schools have expanded their robotics training and education, such as by offering more robotics classes and adding new robotics majors or departments. Kinematic control of robotic manipulators is one of the core subject areas for robotics. This control includes mechanics, forward and inverse kinematic model, path planning, trajectory interpolation and collision-free algorithm. Applying theoretically learned knowledge for actual robotic operation is an important practical requirement for students. Owing to the high cost of robots, it is difficult to purchase sufficient equipment to meet growing teaching needs. Furthermore, improper operation could easily cause damage to the robots and accidents to personnel. Concurrently, university laboratories usually face a shortage of lab time, space and staff.

To resolve the above issues effectively, based on our traditional on-site lab practice, this study explored robot virtual reality (VR) experimental simulation and robot remote-controlled experiments having web-based access.

VR experiments use software that imitates real experiments on real infrastructure without any restrictions of cost, space, time and schedule. Remote control (RC) experiments are a remote means of performing real physical experiments, which enables the setup of equipment and continuous communication with students and provides real experimental data to students conducting experiments inexpensively. Compared with hands-on laboratories, the students access remote experiments over the internet. Fewer people are required to administer laboratories for large student population; students have more flexibility to arrange their experimental timetable.

Depending on students' needs or course requirements, three types of VR experimental simulation, remote-controlled experiments, and on-site experiments can be conducted independently or as a combination of any two or three of them. The experiments could proceed from virtual online to reality. Students could first finish tests on the virtual platform and later move to actual laboratory operations or web-based remote-controlled operations. This new teaching methodology could improve equipment usage efficiency and safety, reduce students' stress and inspire in them the spirit of exploration.

Related works

Robot virtual experimental simulation platform

Currently, there are several studies on robotic simulation platforms, with the following four solutions

1. Robotic simulation software:

Such software as ROBCAD,[1] USARSim,[2] Webots,[3] Simbad,[4] Workspace,[5] MRDS[6] and others[7] can design robotic models quickly and simulate real kinematics, possess realistic 3D effects and can control the robots through script programs. However, these software packages have certain weaknesses: they are expensive, are not open source, or are highly difficult to develop; therefore, their applications are highly restricted.

In addition, robot manufacturing companies have developed simulation platforms, such as ABB RobotStudio.[8] In this study, the simulation objects are only limited to each company's specific robots. Their interface and functionality are not designed to meet a customer's special needs.

2. Software with professional kinematics and dynamics analysis:

ADAMS, MATLAB and others fall into this category. ADAMS/View[9] can set up a robotic visualization model and simulate system kinematic and dynamics characteristics in real environments. Being a commercial software, ADAMS has many limitations. Its underlying algorithm is not open source and its embedding algorithm is difficult to implement.[10] MATLAB[11,12] has high computing power. Its Simulink toolbox can build models, simulate and analyze a dynamic system. Additionally, its Simulink/Robotics toolbox can provide several very important modeling and simulation functions in robotics, which are very easy to use. However, MATLAB and its related toolboxes have weak 3D displays, low operating efficiency and poor algorithm code portability.[13] In some studies,[14,15] MATLAB and ADAMS have been integrated.

3. Integration of 3D modeling software and other programming tools:

3D modeling software such as SolidWorks,[16–19] Pro/E[13] and VR languages such as VRML[20] and Unity3D[21] have very powerful 3D modeling capabilities. Combined with programming languages such as MATLAB,[13,16,20] C++,[17] C#,[18] LabVIEW[19] and Python,[21] they can establish experimental simulation platforms, set up flexible user interfaces and implement various functionalities.

4. Software development based on graphic databases such as Open Graphics Library (OpenGL) and Web Graphics Library (WebGL):

As a cross-platform graphic interface, OpenGL[10,22–25] can present 3D graphics without third-party software or plug-ins; therefore, a simulation platform developed on OpenGL could solve the above problem suitably. However, OpenGL cannot run as a web-based application. WebGL[26–28] can create web-based 3D graphics using JavaScript, which is embedded into an internet browser without installing other software or plug-ins.

Table 1 lists examples of robotic simulation programs.

Most applications of robot virtual simulation[1–10,13,14,16–20,25] are designed for academic or engineering purposes (there are several similar studies that are not listed in this paper). In educational applications,[2,11,12,15,22–24,26–28] the robotic

Table 1. Examples of robot simulation programs.

Programming language	Available robots	Simulation scenarios	Interactive	Educational
ROBCAD	Robot welding station[1]	• Robots and environment 3D simulation; • Welding process simulation of Car-Body-in-White Floor Assembly.	Keyboard Mouse	N
Webots	A wheel-leg hybrid robot[3]	• Wheel-leg hybrid robot prototype; • A set of basic robot motions in structured terrain with varied friction.	Keyboard listener	N
MRDS	Robot-fish[6]	• Robot-fish and environment simulation. • Robot fish straight swimming, turning, backward and collision simulation.	Keyboard Mouse	N
ADAMS	A six-leg-wheel hybrid mobile robot[9]	• Kinematic and dynamic parameters simulated curves in various contact conditions between wheel and ground. • Climbing motion simulation. • Analyze climbing obstacle capability.	Keyboard Mouse	N
MATLAB	A 5-DOF robot manipulator[11]	• Demonstrate position and orientation of end-effectors by inputting joints angle. • Demonstrate joints angle by inputting position and orientation of end-effectors.	Keyboard Mouse	Y
ADAMS MATLAB	A 2-DOF robot arm[15]	• Cosimulation of kinematics and dynamics by setting torques, angle and velocity. • Verify trajectory tracking control.	Keyboard Mouse	Y
SolidWorks MATLAB	A redundant SCARA robot[16]	• Observe dynamic performance velocity & torque, and manipulator movement by modifying arm length and fore length of robotic.	Keyboard Mouse	N
SolidWorks C#	A testing robot[18]	• Real-time motion simulation of pressure vessel detection process	Keyboard Mouse	N

(continued)

Table 1. Continued.

Programming language	Available robots	Simulation scenarios	Interactive	Educational
VRML MATLAB	A humanoid robot[20]	• Simulation of walking and arms coordinated operation by controlling shoulder and hip joints.	Program	N
Unity3D Python	Robot manipulators[21]	• Build any type of serial-link manipulator. • Change position and orientation of joints of the robot through sliders. • Set position and shape of obstacles, then design serial-link manipulators to reach an objective without collision.	Keyboard Mouse Program	Y
OpenGL	A 6-DOF industrial robot[22]	• Demonstrate robot motion by changing joint parameters or end-effector's position for virtual kinematic teaching.	Keyboard Mouse	Y
	Industrial robots[24]	• Create industrial robots and grippers 3D model by inputting geometrical variables. • Simulate kinematic scheme of robot through position and orientation of each joint.	Keyboard mouse	Y
WebGL	A 5-DOF manipulator[26]	• Demonstrate position and orientation of end-effector through sliders to change angles of joints. • Demonstrate joint variables through sliders to change position and orientation of end-effector.	Keyboard Mouse	Y
	A 6-DOF PUMA560 robot[28]	• Observe movement of joint or end-effector through sliders for explaining forward or inverse kinematics.	Keyboard Mouse	Y

models have been established by developers and the learners just observe the motion of end-effectors or joints using a keyboard and/or mouse. These simulations are almost for demonstration of the basic kinematics or dynamics knowledge. In the simulated platform,[21] the programming in Unity3D is for teachers who provide the student files describing the environment of the robot and the robotic tasks. The students use a built-in algorithm to try various robot configurations to find an obstacle-free path to the final position via keyboard and mouse. Although obstacles are introduced, no collision-free method is involved.

Robot remote-controlled platform

With the development of robotics and network technology, robotic RC has been developed and applied in important areas, such as space exploration, underwater operations, dangerous environment operations and web-based medicine. This technology has considerable advantages in terms of laboratory space, the safety of operators and users sharing devices in different locations. This technology has also been used in robotics teaching.

Several studies[29–31] provide practical examples of applying RC on robotics teaching. Lai et al.[29] used the Browser/Server (B/S) architecture model with three layers for RC and remote supervision. The user can control the remote arm robot and observe effects in various control algorithms and parameters. Lopes et al.[30] proposed a remote laboratory for robotics that allows experiments with a mobile robot to be performed, tested and validated remotely. Students can control and manipulate the robot with simple commands, or use the C/C++ programming language to develop complex programs. Radhamani et al.[31] developed a remotely controlled robotics labs as a new pedagogy for augmenting laboratory education. Feedback analysis among students suggested the use of an online-based bioinspired robotics labs in augmenting the conceptual knowledge of working with a robotic arm, forward and inverse kinematics and studying artificial neural network patterns beyond classroom hours.

Shen[32] developed a 3D simulation and RC software based on Client/Server (C/S) structure software for a 5-DOF education robot. The simulation control software can take the place of a conventional teach pendant unit. Safaric et al.[33] integrated remote hands-on experiments into control and/or robotics courses to minimize the gap between theory and practice, teach students the use of remote rapid control prototyping and decrease educational costs. Another study[34] proposed the development of a low-cost educational framework for the RC of a PUMA robot, which is divided into two parts: the PUMA support platform provides basic control and monitoring functions to remote users that are connected to the internet via a local computer; the additional modules include on-line programming, off-line programming and simulation. An RLab[35] is developed that provides students with online access to the real-world hardware for remote experimentation, and a VLab[36] is also developed to simulate robot activity in a nonimmersive VR environment before execution on the actual device.

Other works[29–31] effectively solved the issues of teaching resource sharing and distance learning. However, these methods directly control real robots; thus, any mistakes in control algorithms or operations could damage the robotic equipment. Some studies[32–36] have added robotic 3D virtual simulation into an RC system. Compared to operating directly on real robots, remote-controlled operation with 3D simulation could considerably improve the safety and practicality of robotic teaching. However, the two systems are totally separate in the above studies, which results in users taking more time for different interactions, operations and actions.

Robotic kinematic experiment platform based on B/S

Introduction to the platform

SCARA and 6-DOF articulated robots are widely used in industrial applications. Our platform selected a 4-DOF GOOGOL GRB 3014 SCARA robot and a 6-DOF ABB IRB 120 robot as simulation targets.

A B/S pattern is developed to share resources of experimental education among a broader student audience, and WebGL technology is used to build a 3D model that would make effective realization of the network sharing possible. The students can access the experiment platform through internet anytime from anywhere. Meanwhile, multiple control modes are developed. In addition to the simplest demonstration of robot motion, the students need to set up the kinematics model of certain types of robots by themselves and write algorithms based on robotic tasks in familiar programming languages to control robot activity, so they can validate the effectiveness of their method in the platform.

The functionalities of the robotic kinematic experiment platform are as follows:

1. Observing robots' structural synthesis and motions by parameters;
2. Robot end-effector position and orientation control;
3. Robot trajectory planning;
4. Collision avoidance path planning among robots and obstacles (static or dynamic);
5. Coordination of dual robots under complex environments.

Platform design

Robotic kinematic simulation platform consists of four modules: motion control, motion status display, obstacles or operational targets generation and VR simulation. Among them, the motion control module includes three control modes of shortcut commands control, robot programming language control and interpolation points control. The motion control module realizes robotic or target motion planning in VR simulation module. Furthermore, this module could also calculate the real-time status information for the motion display module and the robot end-effector trajectory for the VR module. In the environments of robot workspace,

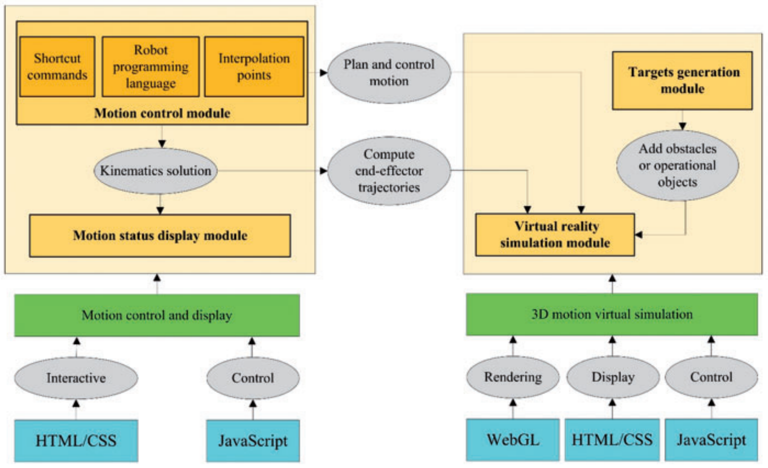

Shortcut commands
Robot programming language
Interpolation points
Motion control module
Kinematics solution
Motion status display module
Plan and control motion
Compute end-effector trajectories
Targets generation module
Add obstacles or operational objects
Virtual reality simulation module
Motion control and display
3D motion virtual simulation
Interactive
Control
Rendering
Display
Control
HTML/CSS
JavaScript
WebGL
HTML/CSS
JavaScript

Figure 1. Module structure and realization of simulation platform.

there will be operating objects and obstacles, namely, the target collectively referred to herein, which may be stationary or moving. The target generation module supplies obstacles or operation objects for the VR simulation module. The motion status display module provides real-time dynamic information of robots for each axis. The virtual simulation module shows the 3D working environment, the robot's movement and its end-effector trajectories.

Figure 1 shows the relationship between the modules and the developing method.

Motion control and display have been mostly implemented using Hyper Text Markup Language (HTML), Cascading Style Sheets (CSS) and JavaScript. HTML/CSS is used to create Human Machine Interface (HMI); JavaScript is used to implement motion control. 3D motion simulation is mostly carried out by HTML, CSS, JavaScript and WebGL graphics database. Among them, WebGL graphics database provides 3D motion simulation module rendering; HTML and CSS provides control display; the JavaScript language is mostly responsible for controlling the behavior of each entity in the 3D motion simulation module.

The HMI of the robot kinematics simulation platform is shown in Figure 2. It consists of: ① motion simulation display window; ② targets generation panel; ③ motion status display panel; ④ shortcut commands panel; ⑤ code editor; ⑥ robot position adjustment panel and ⑦ Help.

The simulation experiment platform has the following characteristics:

1. In the simulation window, one or two robots can be selected and displayed. Their position is adjustable. A target with regular shape can be generated by setting. Models with complex shape can be imported;

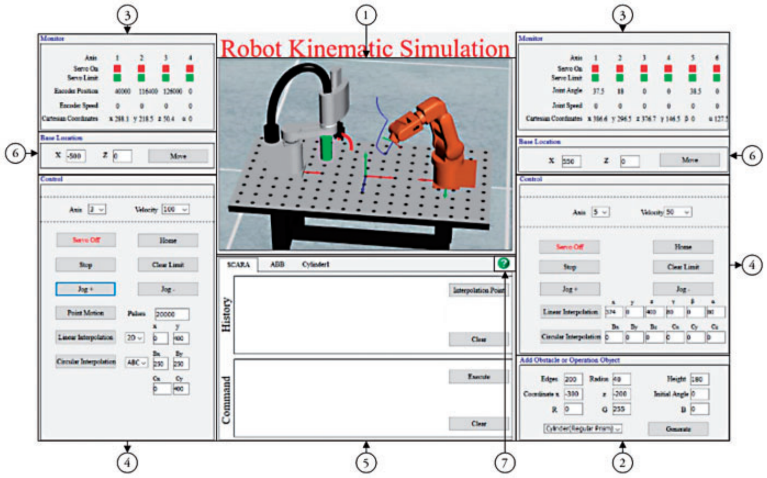


Figure 2. Human Machine Interface (HMI) of robot kinematics simulation platform.

2. A single robot can be controlled independently, and two robots can be controlled to move at the same time. The movement of the target can be controlled through programming;
3. In the simulation window, all objects can be viewed from various directions and sizes.
4. Robotic motion status and the end-effector movement trajectories can be viewed in real-time from the simulation window.
5. Three robotic motion control methods are provided including shortcut commands, robot programming language and interpolation points. The autonomous research algorithm can be tested and analyzed.
6. Software is open and scalable.

Interactive design

Coordinate systems in simulation window. As shown in Figure 3, there are four coordinate systems in the simulation window. According to the D-H coordinate system generally used in robotics, the base coordinate system of the SCARA and the ABB robot are separately established as $\{S\}$ and $\{A\}$, which describes the robot base location in the workspace. The end-effector coordinate system of the ABB robot is also established as $\{E\}$ to enable users to observe the position and orientation of the robot end-effector more clearly. To represent the location of robots, obstacles and targets in a working environment, a world coordinate system $\{W\}$ is established in the center of the workbench surface according to the default coordinate system of WebGL in 3D virtual space.

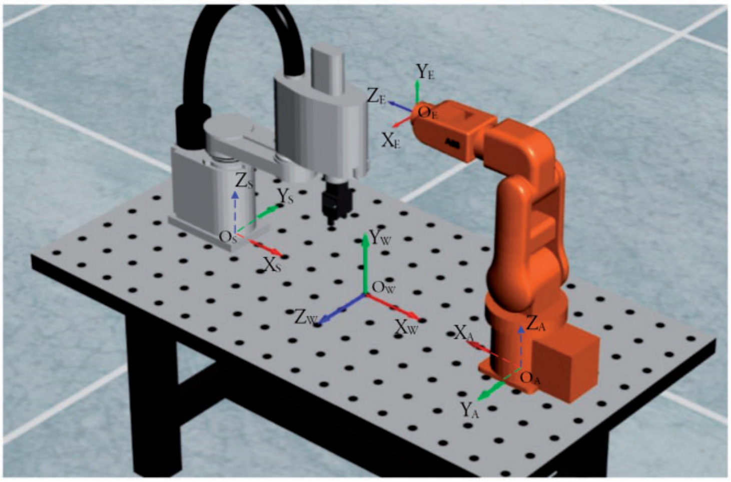


Figure 3. Coordinate systems in simulation window.

The homogeneous transformation matrix from $\{W\}$ to $\{S\}$ and from $\{W\}$ to $\{A\}$ are separately expressed as ${}^{W}_{S}T$ and ${}^{W}_{A}T$

$${}^{W}_{S}T = \begin{bmatrix} 1 & 0 & 0 & {}^{W}x_S \\ 0 & 0 & 1 & 0 \\ 0 & -1 & 0 & {}^{W}z_S \\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad {}^{W}_{A}T = \begin{bmatrix} -1 & 0 & 0 & {}^{W}x_A \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & {}^{W}z_A \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

where $\left({}^{W}x_S, 0, {}^{W}z_S\right)$ and $\left({}^{W}x_A, 0, {}^{W}z_A\right)$ denote the position of the O_S (the origin of coordinate system $\{S\}$) and O_A (the origin of coordinate system $\{A\}$) in the coordinate system $\{W\}$ separately.

The homogeneous transformation matrix ${}^{A}_{E}T$ from $\{A\}$ to $\{E\}$ can be obtained according to robotics forward kinematics. Furthermore, the homogeneous transformation matrix ${}^{W}_{E}T$ from $\{W\}$ to $\{E\}$ can be calculated as follows

$${}^{W}_{E}T = {}^{W}_{A}T\,{}^{A}_{E}T$$

Motion simulation. As shown in Figure 2, the movement of the robot and the target can be observed in real-time in the simulation window. In the HMI, the end-effector trajectory of the SCARA robot is shown in red as a thick curve, while the ABB robot is blue and is a thin curve.

The real-time movement status of each joint of the robot can be displayed on the movement status display panel where it is located, as shown in section ③ in Figure 2. The first line of the panel indicates the servo-enabled status of each axis

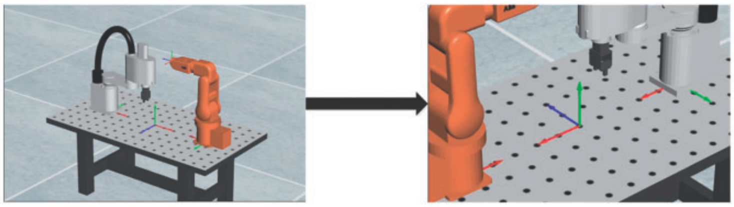


Figure 4. Rotation and magnification of the simulation scene.

of the robot. Red indicates that the servo motor has been powered on, while green indicates that the servo motor is off. The second line indicates the limit of each joint. Red shows robot in a limit position now, while green shows robot in its working area. The third line shows the positions of joints for the robot. The SCARA robot in the left uses the number of encoder pulses as the unit, and the ABB robot in the right uses the angle (°) as the unit, which are the same as their actual application. The fourth line shows the movement speed of each joint.

When the cursor is in the simulation window, scrolling the middle button can scale the scene. Pressing and holding the middle button can rotate the scene to change the viewing angle. Figure 4 shows the rotation and magnification of the simulation scene.

Motion control.

1. Shortcut command control

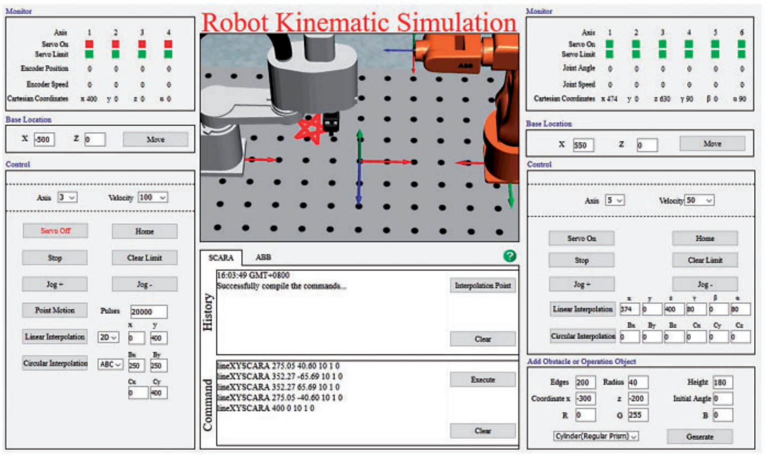
The shortcut command control panel of the SCARA robot and the ABB robot is shown as in section ④ in Figure 2. The "Servo On/Off", "Home", "Stop" and "Clear Limit" buttons control, respectively, the robot servo control on/off, the robot return to the initial position, the robot stop motion and clearing the robot limit alarm.

To observe the robot movement, the program sets up the robots "Jog +", "Jog −", "Point Motion" for each joint and "Linear Interpolation" and "Circular Interpolation" for robot end-effector.

2. Robot programming language control

Simulating motion control functions from the original manufacturer of the robot, robot programming language control mode can control robot motion as real robots actually do. Under the "Motion Command" control mode in the simulation platform ⑤ area, the commands to control robot or target movements can be entered.

A motion command consists of a command function name and parameters sequence. The command function name and every parameter are separated by spaces. The format of a legal motion command is "command function name (space) parameter 1 (space) parameter 2 (space) paremeter3 ....



```
16:03:49 GMT+0800
Successfully compile the commands...
```

```
lineXYSCARA 275.05 40.60 10 1 0
lineXYSCARA 352.27 -65.69 10 1 0
lineXYSCARA 352.27 65.69 10 1 0
lineXYSCARA 275.05 -40.60 10 1 0
lineXYSCARA 400 0 10 1 0
```

Figure 5. SCARA robot drawing a five-pointed star by robot programming language.

In the code editor window, a button is set to switch between two modes: in robot programming language, these are named "Motion Command" and "Interpolation Point". If it is currently in "Interpolation Point" mode, clicking "Motion Command" switches the mode to "Motion Command" control mode and vice versa.

As shown in Figure 5, robot programming language controls the movement of the SCARA robot, and its trajectory is five-pointed star, visible in red thick line on the monitor.

Figure 6 shows the process of detecting the legitimacy of the command format and compiling the commands. If there are errors in writing the code, an alert should be shown where the error happens. If the format of all commands is correct, the algorithm will compile the command list to the commands in the form of JavaScript, which can be executed to control the motion of the robot or obstacle in the virtual simulation environment.

3. Interpolation point control

The robot motion control ultimately involves controlling each joint so that the robot passes through each interpolation point on the trajectory in the proper order. To master and understand the robot motion control method more deeply, the interpolation point mode has been set up. The interpolation point control mode allows users to model and implement algorithms in a familiar programming language (e.g. MATLAB, VB, C++ or LabVIEW) and control the robot following a series of interpolation control instructions, such as ABB robot interpolation point control format "joint angle 1 + space + joint angle 2 + space + joint angle 3 + space + joint angle 4 + space + joint angle 5 + space + joint angle 6". Figure 7 is the space helix trajectory obtained by the interpolation point control method.

```
Algorithm 1 Check and Compile Command
Require: the command list of all robots and obstacles CommandListOfAll, the library of command function name
CommandName, the dictionary of the count of command parameters ParametersCount
procedure CheckAndCompileCommand()
  Initialize ExecuteCommandList
  for CommandList in CommandListOfAll do
    for i←0 to Length(CommandList)-1 do
      SplitCommand←CommandList[i].Split(' ')
      /*Check if the command function name is correct*/
      if SplitCommand[0] in CommandName
      then
        /*Check if the count of command parameters is correct*/
          if Length(SplitCommand)-1 = ParametersCount [SplitCommand[0]]
          then
            /*Combine command function names and parameters into executable JavaScript statements*/
            ExecuteCommand←SplitCommand[0]+"("
            for j←1 to Length(CommandList)-2 do
              ExecuteCommand+=SplitCommand[j]+","
            end for
            ExecuteCommand+=SplitCommand[Length(CommandList)-1]+")"
            ExecuteCommandList.Add(ExecuteCommand)
          else
            print("\nThe number of line "+(i+1).toString()+" parameters is incorrect.\n")
            return 1
          end if
        else
          print("\nFunction name of line"+(i+1).toString()+"is error.\n")
          return 1
        end if
      end for
    end for
    print("\nSuccessfully compile the commands...\n")
    return 0
  end procedure
```

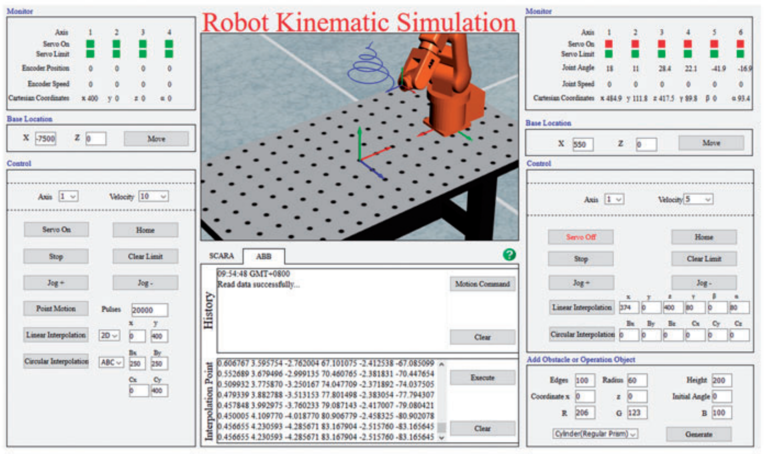
Figure 6. Detect the legitimacy of the command format and compile the commands.

Robot and its working location settings. X and Z values are set in base location of section ⑥ and “move” is clicked, as shown in Figure 2, to change the relative position of the robot base center to the world coordinate system, i.e. to change the position of robot in the workbench. When $X(\text{or } Z) \leq -7500$ or $X(\text{or } Z) \geq 7500$, the robot can be removed from the simulation scene. The SCARA robot has been removed from the scene in Figure 7.

Target setting.

1. Target generation

In the Settings panel shown in Figure 2 of section ②, the drop-down menu at the bottom left provides the choice of three shapes for cylinders, cones and spheres, as well as the import of the 3D model. “Load STL File” is selected, and the path of the STL file to be imported is input in the “Path” text box. The “Generate” button is clicked to import the STL model into the simulation scenario. The model file point cloud data can be downloaded from the same website. The first line of the panel sets the size of the target body, the second line sets the position of the target



Axis	1	2	3	4
Servo On	🟩	🟩	🟩	🟩
Servo Limit	🟩	🟩	🟩	🟩
Encoder Position	0	0	0	0
Encoder Speed	0	0	0	0
Cartesian Coordinates	x 400	y 0	z 0	α 0

X -7500 Z 0

Axis 1 Velocity 10

Pulses 20000

2D x 0 y 400

ABC Bx 250 By 250 Cx 0 Cy 400

```
09:54:48 GMT+0800
Read data successfully...
```

```
0.606767 3.595754 -2.762004 67.101075 -2.412538 -67.085099
0.552689 3.679496 -2.999135 70.460765 -2.381831 -70.447654
0.509932 3.775870 -3.250167 74.047709 -2.371892 -74.037505
0.479339 3.882788 -3.513153 77.801498 -2.383054 -77.794307
0.457848 3.992975 -3.760233 79.087143 -2.417007 -79.080421
0.450005 4.109770 -4.018770 80.906779 -2.458325 -80.902078
0.456655 4.230593 -4.285671 83.167904 -2.515760 -83.165645
0.456655 4.230593 -4.285671 83.167904 -2.515760 -83.165645
```

Axis	1	2	3	4	5	6
Servo On	🟥	🟥	🟥	🟥	🟥	🟥
Servo Limit	🟩	🟩	🟩	🟩	🟩	🟩
Joint Angle	18	11	28.4	22.1	-41.9	-16.9
Joint Speed	0	0	0	0	0	0
Cartesian Coordinates	x 484.9	y 111.8	z 417.5	γ 89.8	β 0	α 93.4

X 550 Z 0

Axis 1 Velocity 5

Linear Interpolation: x 374, y 0, z 400, γ 80, β 0, α 80

Circular Interpolation: Bx 0, By 0, Bz 0, Cx 0, Cy 0, Cz 0

Edges 100 Radius 60 Height 200
Coordinate x 0 z 0 Initial Angle 0
R 206 G 123 B 100
Cylinder(Regular Prism)

Figure 7. ABB robot drawing a spiral trajectory by interpolation points.

body relative to the world coordinate system and the third line sets the color of the target body. A green cylinder object example is shown in Figure 2.

2. Target motion control

After the target is generated, it appears in the tab of the control window. By clicking on the icon, the user can exercise control in the code editor. Similarly, there are two forms of control, robot programming language and interpolation points.

For example, under "Motion Command", the instruction "lineObstacle -100 0 400 0 0 0 10" controls the target and moves it from its current position and orientation at 10 mm/s to the target position and orientation of $x = -100$, $y = 0$, $z = 400$, $\gamma = 0$, $\beta = 0$, $\alpha = 0$.

Interpolation point control can not only realize the control of an arbitrary trajectory of the target but also control the orientation of each position and the running speed.

As shown in Figure 8, the interpolation points command that control object from $x = 0$, $y = 0$, $z = 200$, $\gamma = 0$, $\beta = 0$, $\alpha = 0$ to do circular motion with a radius of 200 mm and point $(0, 0, 0)$ as the center, and the object accelerates from the standstill with 0.52 rad/s angular acceleration at 0–2 s, then decelerates to the standstill with -0.52 rad/s angular acceleration at 2–4 s.

3. Target deletion

Positioning the cursor to a target body, its operating menu is selected by right-clicking, and then clicking "Delete" deletes the target body.

Interpolation points command to control an object motion
0 0 200 0 0 0
0.0145444104204664 0 199.99999947115 0 0 0
0.0581776409126849 0 199.999991538405 0 0 0
0.13089968455403 0 199.999957163177 0 0 0
0.232710514423179 0 199.999864614495 0 0 0
……
173.321318766813 0 -99.7983990900294 0 0 0
173.270493501388 0 -99.8866161294172 0 0 0
173.234162249388 0 -99.9496124542398 0 0 0
173.212352504101 0 -99.9874039066679 0 0 0
173.205080756888 0 -100 0 0 0

Figure 8. Interpolation points command example.

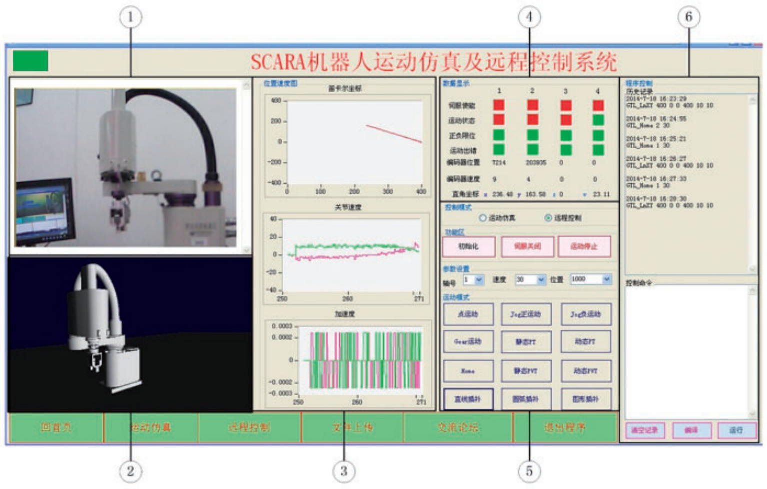


Figure 9. Client interactive interface of SCARA robot remote control and virtual simulation.

SCARA robot RC experiment platform

Another SCARA robot RC/virtual simulation teaching platform based on C/S mode is developed that is available to students of some specific majors. The students can control the simulated robots first, the move to real robots in the laboratory after the expected kinematics results are achieved. The client software system takes a Windows system as the development environment, Microsoft Visual Studio 2010 as the software development platform and C# as the software development language. The client HMI is shown in Figure 9 (shown in Chinese). Unlike other designs, the virtual simulation and the RC are integrated into one consistent application. They share a control platform so that students need not spend more time in learning interfaces and functions. The interface design includes the following: ① robot live video in real time; ② robot VR motion simulation; ③ robot end-effector position, joint speed and acceleration curve; ④ robot joints status and motion information ⑤ shortcut command control panel and ⑥ code editor and historical operation records.

RC/virtual simulation experiment teaching system is divided into client layer, service layer and physical layer. The system logical structure and function are shown in Figure 10.

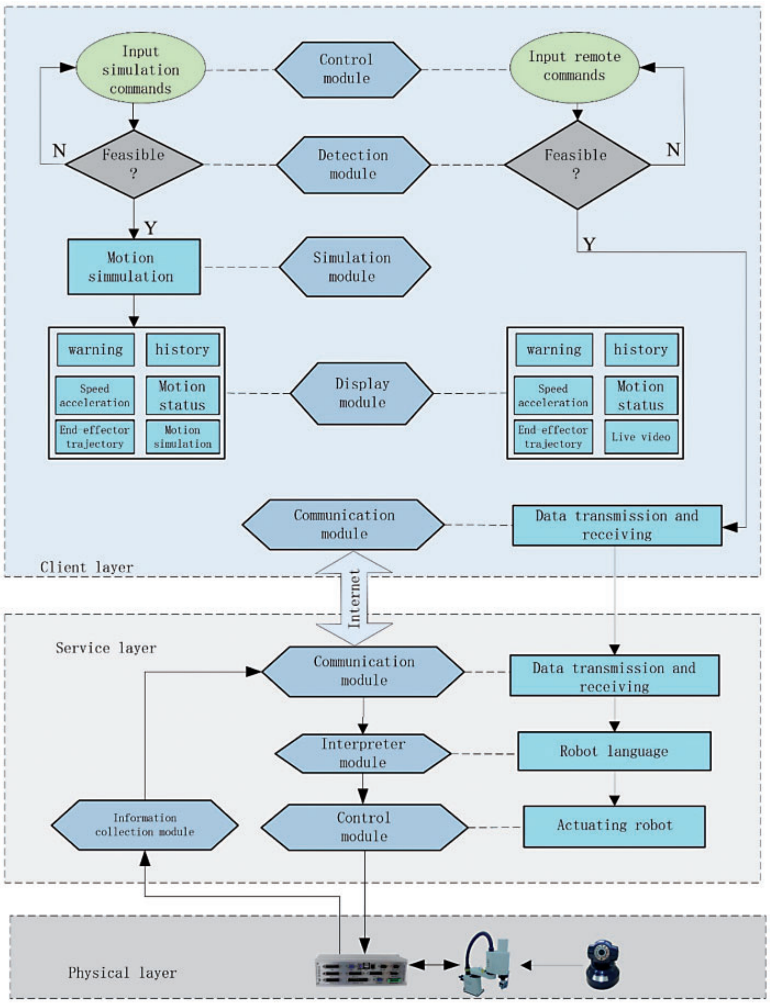


Figure 10. Remote control and virtual simulation experimental teaching system framework.

SCARA robot kinematics simulation based on OpenGL

The client-side control module generates the motion simulation command. The detection module detects whether the command satisfies the current robot behaviors and, if so, sends the command to the simulation module, and the simulation module performs the motion simulation and feeds back the motion state to the display module.

The simulation module adopts OpenGL technology to construct the VR environment, which includes 3D modeling and virtual motion simulation. The virtual control commands or functions are programmed based on the actual motion control library of the robot with the same calling and control effects as the actual robot.

SCARA robot RC based on C/S

The RC system adopts the C/S mode to realize network communication based on WinSock technology of Transmission Control Protocol/Internet Protocol (TCP/IP) protocol. The server establishes the monitoring thread and continuously monitors requests from the client. A connection between client and server is established to achieve two-way data transmission. The server receives the client's control commands and feeds back the robot's motion status to the client.

Control information is passed from the client to the robot. To avoid blocking and losing information in the transmission process, multiple motion control instructions are packaged and sent as a single packet.

The client-side control module generates an RC command. The detection module detects whether the command satisfies the current robot behavior and, if yes, sends the control commands to the client communication module. The client communication module encodes the control commands and then transmits them to the server via the internet. The server communication module receives and decodes the client's remote-control commands, and then the interpreter module converts the decoded commands into an executable robot driver. The server control module drives the robot to realize RC according to the driver executable program.

Remote monitoring status information is obtained and transmitted in real time from the real device. The status display includes robot live video, robot joint servo, limits, position and speed of the encoder, end-effector position and joints speed and acceleration curve.

Applications and evaluation

Applications

The robot virtual simulation platform can be found by logging in to mechanical basic experimental teaching website (http://222.16.42.167/jixie/index.aspx). The experimental platform is open to all students. Students can log in to the website anytime and anywhere, select tasks of varying difficulty levels according to their learning conditions, and validate the learning outcomes. RC and on-site experiments operate by online appointments. They are mainly for students in Robotics class, students in Mechanical Engineering and Automation taking Mechanical Design courses, as well as graduate students in related fields.

Figure 11 shows the knowledge cells and their corresponding experiment items. The experiment makes it easy to begin and to advance progressively into more complex contents. The simplest task is observing joints movements by shortcut command. The most complex experiment can be designed as a coordinated motion control algorithm of two robots' cooperation, and can include a robot avoiding obstacles as well as the other robot, priority design of two robot paths, sequential coordination and other tasks that may involve a collision avoidance algorithm, cooperation strategy, optimization design or some machine learning methods

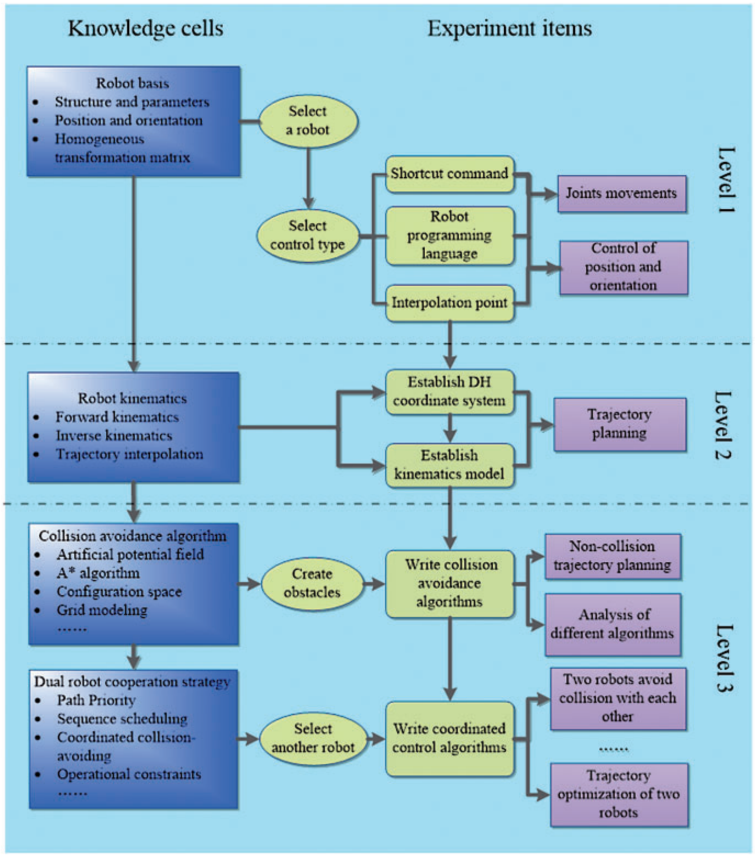

Knowledge cells
Experiment items
Robot basis
Structure and parameters
Position and orientation
Homogeneous transformation matrix
Select a robot
Select control type
Shortcut command
Robot programming language
Interpolation point
Joints movements
Control of position and orientation
Level 1
Robot kinematics
Forward kinematics
Inverse kinematics
Trajectory interpolation
Establish DH coordinate system
Establish kinematics model
Trajectory planning
Level 2
Collision avoidance algorithm
Artificial potential field
A* algorithm
Configuration space
Grid modeling
......
Create obstacles
Write collision avoidance algorithms
Non-collision trajectory planning
Analysis of different algorithms
Dual robot cooperation strategy
Path Priority
Sequence scheduling
Coordinated collision-avoiding
Operational constraints
......
Select another robot
Write coordinated control algorithms
Two robots avoid collision with each other
......
Trajectory optimization of two robots
Level 3

Figure 11. Execution experiments step-by-step.

except robot kinematics. Even for the same experiment item, such as trajectory planning, there are large differences, such as that it can be a straight line in a plane or a complicated curve and geometry in 3D space.

Three specific examples are described briefly below.

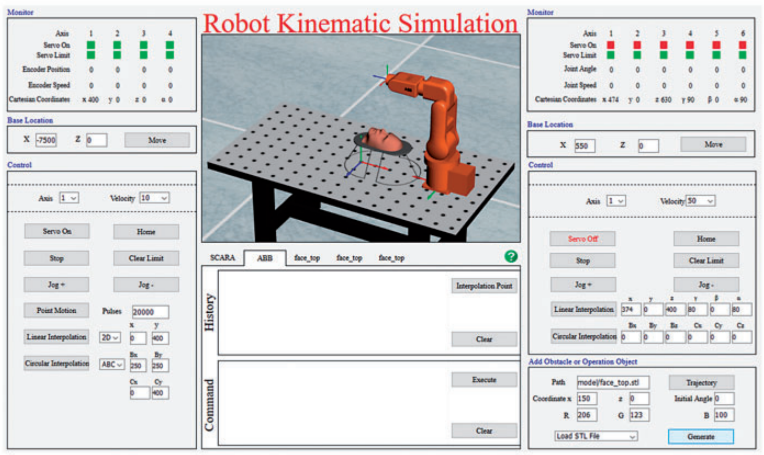
1. VR application examples: facial trajectory planning and orientation control

The operation requires the end-effector of the robot to touch the face and move in a predetermined trajectory while ensuring that the mid-axis (Z-axis) of the end-effector is always collinear with the normal vector of the facial current contact point. A facial STL model can be imported into the simulation scenario. Through given triangulated surfaces data, students perform orientation planning and control. Figure 12 shows the simulation.

2. VR + on-site Application Example: Dual-robot collision-free trajectory planning

To verify the validity of the robot collision-free trajectory planning algorithm, a dual-robot coordinated trajectory planning experiment was designed. The robots ABB and SCARA each move from the starting point to the target point with overlapping work spaces, while there is a stationary obstacle and a moving obstacle in the environment. The moving obstacle moves linearly along the Z direction at a speed of 20 mm/s.

Figures 13 and 14 show part of the virtual simulation and the actual field experiment, respectively. In the figures, images (a) and (b) show that two robots move to their respective target points when no obstacle is encountered; images (c) and (d) show that the SCARA robot is avoiding the stationary obstacle, while the



Axis	1	2	3	4
Servo On	🟩	🟩	🟩	🟩
Servo Limit	🟩	🟩	🟩	🟩
Encoder Position	0	0	0	0
Encoder Speed	0	0	0	0
Cartesian Coordinates	x 400	y 0	z 0	α 0

Base Location: X -7500 Z 0

Axis 1 Velocity 10

Pulses 20000

Linear Interpolation: 2D, x 0, y 400

Circular Interpolation: ABC, Bx 250, By 250, Cx 0, Cy 400

Axis	1	2	3	4	5	6
Servo On	🟥	🟥	🟥	🟥	🟥	🟥
Servo Limit	🟩	🟩	🟩	🟩	🟩	🟩
Joint Angle	0	0	0	0	0	0
Joint Speed	0	0	0	0	0	0
Cartesian Coordinates	x 474	y 0	z 630	γ 90	β 0	α 90

Base Location: X 550 Z 0

Axis 1 Velocity 50

Linear Interpolation: x 374, y 0, z 400, γ 80, β 0, α 80

Circular Interpolation: Bx 0, By 0, Bz 0, Cx 0, Cy 0, Cz 0

Path model/face_top.stl

Coordinate x 150 z 0 Initial Angle 0

R 206 G 123 B 100

Figure 12. Robot facial trajectory planning and orientation control.

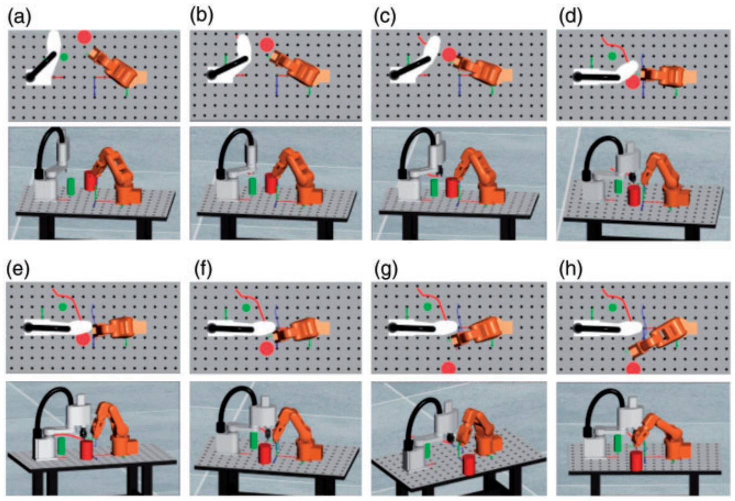


Figure 13. Virtual simulation experiment of double-robot's avoiding collision.

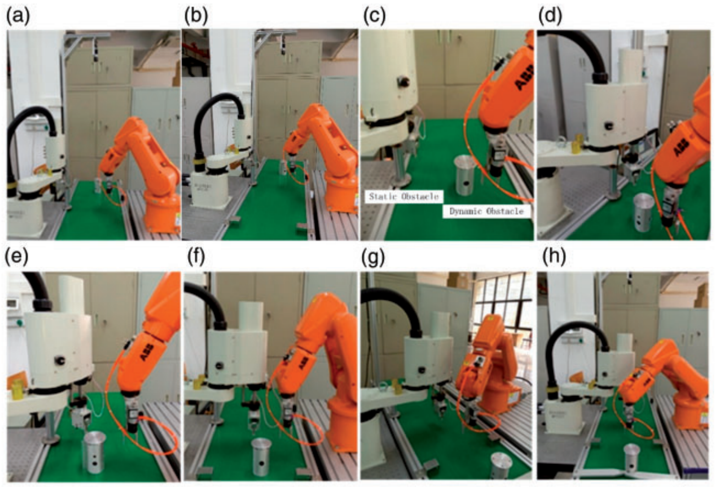


Figure 14. Actual experiment of double-robot's collision avoidance.

ABB robot is avoiding the moving obstacle; images (e) and (g) show that the two robots avoid each other; image (h) shows that both robots move to the respective target point.

3. VR + RC application example: robot trajectory optimization

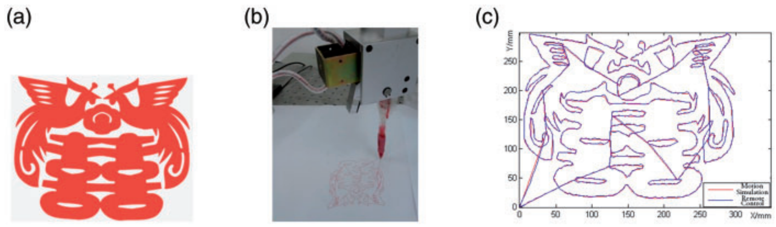


Figure 15. SCARA robot trajectory optimization. (a) Paper-cut pattern. (b) Remote-hand on the robot. (c). Trajectory comparison of VR and RC. VR: virtual reality; RC: remote control.

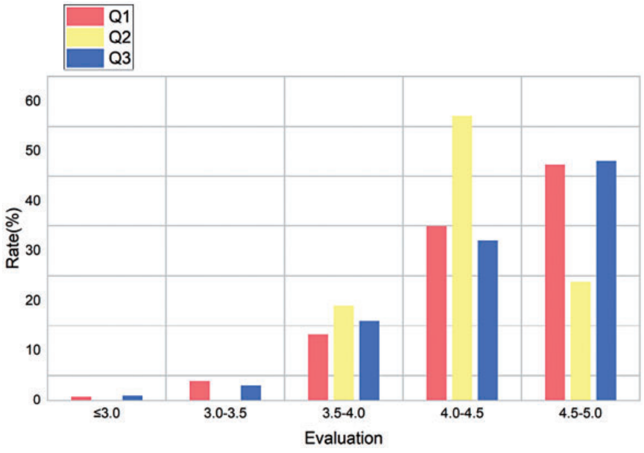


Figure 16. Questionnaire statistical results.

Figure 15(a) shows a paper-cut pattern. Based on the profile data of the pattern, the trajectory of the robot is planned and optimized based on the shortest path. Figure 15(b) shows the robot working by RC and Figure 15(c) shows the results by virtual simulation and RC.

Evaluation

Questionnaires on new experimental mode are distributed and rated by 1 (lowest) to 5 (highest). Specific evaluation of statistical results from approximately 150 undergraduate students in the last three years are shown in Figure 16. For question 1 (Q1) that asked whether VR and RC experiments are flexible or convenient for their experimental schedules, sites or contents, 82.16% of the students gave a more than 4.0 evaluation. For question 2 (Q2) that asked whether Robot Kinematic Simulation is effective or helpful for their studies in the relevant knowledge of robots, no student gave a score of 3.5 or less, 57.14% gave a score between 4.0 and

4.5 points, while 23.81% gave a score between 4.5 and 5.0. In the question 3 (Q3) of HMI design, 80% of students gave more than 4.0 points.

Conclusions

Robotics courses have been provided for undergraduate and graduate students majoring in mechanical engineering in universities or colleges, and an increasing number of robotics practitioners and enthusiasts wish to join the courses. The current laboratory robotic equipment and staffing cannot meet the rising need. In this study, a robot experiment teaching system integrating a variety of experimental modes has been established to meet the needs of learners studying at different levels. Robot VR simulation experiments and robot RC experiments are conducted as alternatives or complements to traditional on-site experiments.

Online virtual simulation experiment resources benefit a wide range of students and effectively solve such issues as expensive experimental equipment and potential security risks. RC experiments have solved the bottleneck problem associated with experiment teaching of distance education. Both experimental modes can effectively alleviate the shortage of laboratory personnel and laboratory space. The combination of multiple experimental modes usually performs simulation verification in a virtual environment and further manipulates the robot through RC or online experiment. Evaluations by students verify the flexibility and effectiveness of VR and RC experiments.

At present, the virtual simulation platform is limited to the motion simulation of two kinds of robots. In future, the simulation objects and contents can be further broadened. For example, a robot library can be built to provide more robots of varied types and sizes for analysis and comparison. In addition, the current platform cannot achieve collision detection between robots and between robots and obstacles, which needs to be addressed in future.

Acknowledgement

The authors sincerely thank Dr Ningning Zhou for her assistance in writing this article in English.

Declaration of conflicting interests

The author(s) declared no potential conflicts of interest with respect to the research, authorship, and/or publication of this article.

Funding

The author(s) disclosed receipt of the following financial support for the research, authorship, and/or publication of this article: This work was supported by funding from the Undergraduate Deep Reform Project of the Ministry of Education of China, Guangdong Province, Higher Education Teaching Reform Project and the SCUT Teaching Research Project.

References

1. Zhang SF, Guo JG, Ji CJ, et al. *Visual simulation of the robot welding station of car-body-in-white floor assembly*. Berlin, Heidelberg: Springer, 2012, pp. 151–155.
2. Carpin S, Lewis M, Wang J, et al. USARSim: a robot simulator for research and education. 2007: 1400–1405.
3. Vargas GA, Gómez DJ, Mur O, et al. Simulation of a wheel-leg hybrid robot in Webots. *Robotics and Automation* 2017; 1–5.
4. Zhang R, Zou Y, Xu Z, et al. The review and implementation of multi-robot system simulation software. *Control and Decision Conference* 2014; pp. 3869–3874. IEEE.
5. Owens J. WORKSPACE-a microcomputer-based industrial robot simulator and off-line programming system. In: *Next Steps for Industrial Robotics, IEE Colloquium on*. 1994, pp. 4/1–4/4.
6. Zhu Y, Chen P and Cai Q. Design and implementation of the robot-fish simulation system. In: *Electrical & Electronics Engineering*. 2012; pp. 539–542. IEEE.
7. Su H. Design and Realization of Industrial Robot 3D Simulator Based on ODE. MASTER Thesis, Chongqing University, Chongqing 2012.
8. Connolly C. Technology and applications of ABB RobotStudio. *Industr Robot* 2009; 36: 540–545.
9. Li YJ and Wu ZW. Modeling and simulation of a six-leg-wheel hybrid mobile robot based on adams. *AMM* 2010; 26-28: 194–197.
10. Fu B, Hu FF, Chen L, et al. The multi-robot simulation platform research and realize based on Open GL. *Combined Machine Tool and Automatic Machining Technology*. 2016; 1: 20–23.
11. Yuan S, Liu Z and Li X. Modeling and simulation of robot based on Matlab/SimMechanics. In: *China Control Conference*. 2008, pp. 161–165. IEEE.
12. Xie B and Cai ZX. Robot simulation experiment teaching based on Matlab robotics toolbox. *Computer Education* 2010; 19: 140–143.
13. Gan YH and Dai XZ. An efficient method of constructing open 3D simulation environment for articulated robot. *Robot* 2012; 34: 628–633.
14. Cheraghpour F, Vaezi M, Jazeh HES, et al. Dynamic modeling and kinematic simulation of Stäubli© TX40 robot using MATLAB/ADAMS co-simulation. In: *IEEE international conference on mechatronics*. 2011, pp. 386–391. IEEE.
15. Luo H, Liu Y, Chen Z, et al. Co-Simulation control of robot arm dynamics in ADAMS and MATLAB. *Res J Appl Sci Eng Technol* 2013; 6: 3778–3883.
16. Mariappan SM and Veerabathiran A. Modelling and simulation of multi spindle drilling redundant SCARA robot using SolidWorks and MATLAB/SimMechanics. *Rev Fac Ing Univ Antioquia* 2016; 81: 63–72.
17. Li DW and Xu X. Design of 3D Simulation and Off—line Programming System for Multi—robot System. *Industr Control Comput* 2014; 27: 39–40.
18. Huang XC, Zhang ML, Li MH, et al. 3D real-time motion simulation method based on SolidWorks secondary development. Journal of Machine Design 2014; 31: 12–15.
19. Zhu F, Diao Y and Wu YD. Minimally invasive surgical robot motion simulation based on Lab VIEW and SoildWorks. *Machinery Design & Manufacture* 2013; 3: 136–138.
20. Zheng-Wen LI, Zhang GL and Zhang WP, JB. A simulation platform design of humanoid robot based on SimMechanics and VRML. *Proc Eng* 2011; 15: 215–219.
21. Arnay R, Hernández-Aceituno J, González E, et al. Teaching kinematics with interactive schematics and 3D models. *Comput Appl Eng Educ* 2017; 25: n/a–n/a.

22. Wang YS, Gai YX and Wu FY. A robot kinematics simulation system based on OpenGL. In: Robotics, Automation and Mechatronics. 2011, pp. 158–161. IEEE.
23. Jiang DY and Li HL. Application of computer virtual simulation on robot teaching. *Combined Machine Tool and Automatic Machining Technology* 2003; 11: 89–90.
24. Marcu C, Lazea G and Robotin R. *An OpenGL application for industrial robots simulation*. In: IEEE International Conference on Automation, Quality and Testing, Robotics, 2006, pp. 254–259. IEEE.
25. Zhang Z, Ma S, Li B, et al. *Development of an OpenGL based multi-robot simulating platform*. International Conference on Control, Automation, Robotics and Vision Kunming, 2004, pp. 1400–1405. China.
26. Demin A, Dai YX, Li X, et al. 3D VR kinematic robot simulation based on HTML5 and WebGL. *J Northeast Univ* 2014; 35: 564–568.
27. Zhou SY. *Design and implication of robotics virtual experiment system*. Beijing University of Posts and Telecommunications, 2014.
28. Dey U, Jana PK and Kumar CS. Modeling and Kinematic Analysis of Industrial Robots in WebGL Interface. In: *IEEE eighth international conference on technology for education*. 2017, pp. 256–257. IEEE.
29. Lai XZ, Yang YH and Wu M. Server design and development of arm robot network teaching experiment system. *J Hannan Univ Technol* 2004; 18: 37–41.
30. Lopes M, Gomes I, Trindade R, et al. Web environment for programming and control of mobile robot in a remote laboratory. In: *IEEE Transactions on Learning Technologies*, 2017, pp. (99).1–1.
31. Radhamani R, Nizar N, Kumar D, et al. Low cost neuro-inspired robots for sustainable laboratory education. In: *International Conference on Robotics and Automation for Humanitarian Applications*. 2017, pp. 1–6. IEEE.
32. Shen J. *3D Visual Off-line Programming and Remote Control for EduRobot-III*. Shanghai Jiao Tong University, 2006.
33. Safaric R, Trunti M, Hercog D, et al. Control and robotics remote laboratory for engineering education. *International Journal of Online Engineering* 2005; 1: 1.
34. Nakao EY, Eguti CCA, Silva CEO, et al. An educational framework for remote control of a PUMA robot. In: *International Congress of Mechanical Engineering*, 2009.
35. Safaric R, Hedrih I, Klobucar R, et al. Remote controlled robot arm. *IEEE Int Conf Industr Technol* 2003; 2: 1202–1207.
36. Safaric R, Sinjur S, ZAlik B, et al. Control of robot arm with virtual environment via the Internet. *Proc IEEE* 2003; 91: 422–429.