A parallel robotic attachment and its remote manipulation

Lihui Wang\textsuperscript{a,*}, Fengfeng Xi\textsuperscript{b}, Dan Zhang\textsuperscript{c}

\textsuperscript{a}Integrated Manufacturing Technologies Institute, National Research Council of Canada, 800 Collip Circle, London, Ontario N6G 4X8, Canada
\textsuperscript{b}Department of Aerospace Engineering, Ryerson University, Toronto, Ontario M5B 2K3, Canada
\textsuperscript{c}Faculty of Engineering and Applied Science, University of Ontario Institute of Technology, Oshawa, Ontario L1H 7K4, Canada

Received 7 October 2005; accepted 25 November 2005

Abstract

This paper discusses a 3-dof (degree of freedom) parallel robotic attachment and its remote manipulation. This attachment is designed as a tripod that provides two rotary motions and one linear motion. The attachment can be mounted onto a variety of machines for different applications, including CNC milling machines, industrial robots, and CMM. Java technologies are used to develop a remote manipulation system for the parallel robotic attachment, including remote monitoring and control. The main difference of this system from the existing web-based or internet-based remote systems is the way to control the motion of the machine from a remote site. Instead of using a camera for monitoring, the tripod is modeled using 3D computer graphics with behavioral control nodes embedded. Compared with camera-based solutions, network traffic is largely reduced, thereby making real-time remote device manipulation practical on the web. Our parallel robotic attachment is one type of parallel kinematic mechanisms (PKM). With PKM emerging as a new way of building flexible systems or agile machines, its advantage over serial mechanism is also presented.

\textcopyright 2006 Elsevier Ltd. All rights reserved.

Keywords: Parallel kinematic machines; Stiffness mapping; Inverse kinematics; Remote real-time manipulation

1. Introduction

Parallel kinematic mechanisms have significant advantages over serial mechanisms. In particular, they possess reduced moving mass and higher stiffness. Thus, parallel mechanisms can work at higher velocities, and yet maintain sufficient rigidity to deliver high levels of accuracy. This research represents our continuous efforts of PKM design, analysis and control. Parallel mechanisms generally comprise two platforms that are connected by joints or legs acting in parallel [1]. Over the past decades, parallel mechanisms have received much attention from researchers and industrial practitioners. They can be found in several practical applications, including aircraft simulators [2], position tracker [3], telescopes [4], and micro-motion device [5]. More recently, they have been used in the development of high precision machine tools [6–8] by many companies such as Giddings & Lewis, Ingersoll, Hexel, Geodetic and Toyoda. Among others, hexapod machine tools are one type of the successful applications.

With the growing manufacturing decentralization, design and fabrication functions of most original equipment manufacturers (OEM) have been distributed geographically. Following the trend, manufacturing operations have been shifted rapidly from intra-corporation to global networks. How to keep all activities under control is a challenging issue. In such a distributed environment, controllability and predictability of factory operations are two essential requirements for web-based collaborative manufacturing. In this paper, this challenge is addressed through a parallel robotic attachment by applying the latest Web and Java technologies. Since 1993 shortly after its debut, the Web has gained a wide acceptance for developing business applications. It has been widely used by many as a medium to share information, knowledge and business decisions [9–11]. Due to the decentralization of today’s manufacturing, web-based solutions have become the...
norm for distributed collaborative manufacturing because of their platform, network and operating system transparency, and their easy-to-use user interface—the web browser. In addition to the Web, Java technology has brought about a fundamental change in the way that applications are designed and deployed. With Java, the browser paradigm has emerged as a compelling way to produce manufacturing applications over the Internet. In this paper, we introduce a Java-based monitoring and control approach that demonstrates a great potential to support dispersed engineers in a collaborative environment. Powered by Java’s “write once, run anywhere” concept, our Java-based approach can largely reduce the complexity and cost traditionally associated with software development on multiple distinct hardware platforms. A parallel robotic attachment is introduced as a testing device for system validation. This 3-dof (degrees of freedom) attachment can be combined with milling machines, industrial robots, or coordinate measuring machines (CMM) to provide fine tuning for motion control.

This paper reports on new R&D findings and is arranged as follows. The principles of the Java-based monitoring and control approach are presented in Section 2. The mechanism and kinematics of a parallel robotic attachment (a tripod) are revealed in Section 3. For the purpose of system validation, the parallel tripod is attached to a gantry system for light polishing. It is used as a case study in Section 4 to demonstrate the Java-based monitoring and control. Finally, conclusions and future work are summarized in Section 5.

2. Java-based monitoring and control

2.1. Concept and approach

Traditionally, remote monitoring and control solutions have been following two approaches: signals only or bundled with cameras for visualization. With the emergence of the Internet and Web, more and more companies are seeking compelling ways for web-based solutions. Although a camera-based monitoring and control system can be ported to the Internet as a web-based application, the bandwidth conservation of a network prevents it from being practical as a real-time solution. Targeting the problem, we propose a Java-based approach to combining sensor data with computer graphics. It utilizes the latest Java technologies including Java 3D and Java Servlet for system implementation, and allows clients to monitor and control a distant device with visual helps enabled by a Java 3D model. The 3D model is a replica of the physical device being rendered off-site for visualization, and its behavior is driven directly by real-time sensor signals of its physical counterpart. Once downloaded from an application server, the 3D model works on behalf of its counterpart at a client side, but remains alive by connecting with the physical device through low-volume sensor data passing. In other words, the hybrid solutions for meeting user requirements demanding rich data sharing and the real-time constraints over the Internet are: (1) using interactive scene-graph based Java 3D models instead of bandwidth-consuming camera images for off-site visualization; (2) transmitting only the sensor data and control commands between models and device controllers for monitoring and control; (3) providing clients with a platform-transparent graphical user interface for navigation; and (4) deploying major control logics in a secure application server for property protection. The goal is to significantly reduce network traffic, while still providing end users with an intuitive environment. It hence makes real-time monitoring and control practical over the Internet for users on relatively slow hook-ups.

This Java-based monitoring and control approach is supported by a Wise-ShopFloor framework [12], and shown as modules in the grey boxes in Fig. 1. Within the framework, the mid-tier application server handles major security concerns, such as session control, user registration, sensor data collection/distribution, and real device manipulation. A central Session Manager is designed to look after the issues of user authentication, session control, session synchronization, and sensitive data logging. All initial transactions need to pass through the Session Manager for access authorization. As a server-side module, the Signal Collector is responsible for sensor data collection from networked physical devices. The collected data are then passed to another server-side module Signal Publisher who in turn multicasts the sensor data to the registered subscribers (clients) through applet-servlet communication. A Registrar is designed to maintain a list of subscribers with the requested sensor data. A physical device is controllable only by the Control Commander that resides in the same application server.

The concept of the Java-based monitoring and control is designed to use the popular client-server architecture and VCM (view-control-model) design pattern. The control-tier is an application server that hosts major manufacturing control logics. The model-tier is a database that stores needed data, knowledge and 3D models for monitoring and control. At client side, a Java applet user interface (fitting in the view-tier) provides clients with a model-based and sensor-driven environment. In a multi-client environment, different clients may require different sets of sensor data for device manipulation. It is not efficient to have multiple clients who share the same model talk with the same device at the same time. Instead, a publish-subscribe design pattern is adopted, based on which two server-side modules (Signal Collector and Signal Publisher) are responsible for sensor data collection and distribution to the right clients, efficiently. HTTP Streaming is chosen as the communication protocol between applets and servlets. Fig. 2 illustrates how a tripod is monitored and control remotely according to our Java 3D enabled concept.
Although the global behavior of the Java 3D model is controlled by the application server based on real-time sensor signals, a remote user still has the flexibility of monitoring the model from different perspectives, such as changing viewpoint location (translation, rotation, orbiting, zooming) or orientation (panning, tilting). An authorized user can submit control commands to the application server. The server then takes over the control for real device manipulations. Although this approach is proposed as an alternative of camera-based monitoring systems, an off-the-shelf web-ready camera can easily be switched on remotely to capture unpredictable real scenes for diagnosis purposes, whenever it is needed. Further details on architecture design can be found in the previous publication [12].
2.2. Security measures

According to an NCMS report [13], there is a growing consensus that linking physical devices to the Internet will become the backbone technology for collaborative manufacturing. However, a major concern of implementing Internet or Web-based collaborative manufacturing systems is the assurance that proprietary information about the intellectual property owned by the organization or information about the company’s operations is available only to those authorized individuals. Any web-based collaborative systems must accommodate privacy of the individuals and organizations involved in collaborative activities. In a highly competitive manufacturing environment, the information about the operations of or the information provided by individuals or organizations should only be shared by those involved. Clearly, it is even more critical to avoid security breeches with physical devices. Java-based remote monitoring and control typically involve sharing information in the form of detailed run-time operations, as well as real-time and mission-critical device controls. For general acceptance of our approach, the secrecy of the proprietary information must be properly maintained. To meet the security requirements, our approach depends on a security infrastructure built into the Java platform. This security infrastructure consists of byte-code verification, security policies, permissions, and protection domains. In addition to the security infrastructure, other security and privacy issues are considered for implementation, including digital rights management for information access and sharing, data encryption, and process confidentiality protection.

Fig. 3 shows how a remote user can get access indirectly to the real device without violating the security policy. All data communication between the end user and a device goes through the application server, and is processed by a server-side module before passing the data to its receiver. As mentioned earlier, only the server-side modules are allowed to collect sensor data or manipulate devices within their limits. On the other hand, all end users are physically separated from the real devices by using segmented networks (Intranet/Internet, and Factory Network) with the application server as a gateway.

3. Parallel robotic attachment

3.1. Tripod kinematics

The parallel robotic attachment—a tripod, as shown in Fig. 2, is one type of parallel kinematic machines (PKMs). Realizing that parallel structure is stiffer, researchers and engineers started developing PKMs in the past decade, which is now considered another milestone after NC in machine tool history. As mentioned earlier, one of the advantages of PKM is that no motor has to carry another. This feature opens up new potentials in machine tool industry to develop lightweight machines running at high speeds with high accuracy.

The tripod used as a testing device in this paper was developed as an attachment at National Research Council of Canada [14]. The tripod is based on the fixed-length leg configuration. A tool can be mounted on the moving platform (the top plate) of the tripod. The movement of the moving platform is controlled by sliding the three fixed-length legs along their respective guide ways. The mobility requirement of this device can be examined by the Chebychev–Grübler–Kutzbach Criterion [15]:

\[ M = d(n - g - 1) + \sum_{i=1}^{g} f_i, \]  

(1)
where $M$ denotes the mobility or the system’s degrees of freedom (dof), $d$ is the order of the system ($d = 3$ for planar motion, and $d = 6$ for spatial motion), $n$ is the number of the links including the frames, $g$ is the number of joints, and $f_i$ is the number of degrees of freedom for the $i$th joint. For the device shown in Fig. 2, $M = 3$ and $d = 6$. The total number of links $n$ is 8 (three sliding-legs, three guideways, and two platforms). The total number of joints $g$ is 9, three for each sliding-leg/guideway system. By substituting $M = 3$, $d = 6$, $n = 8$, and $g = 9$ into Eq. (1), in order to satisfy $M = 3$, the dof for each sliding-leg/guideway system should be five. In our design, one end of a sliding-leg is connected to a guideway by a revolute joint (1 dof) and the other end is connected to the moving platform by a spherical joint (3 dof). Each sliding-leg is driven by a DC servomotor with lead screws (1 dof).

Although the tripod under study can rotate about and move along the $x$, $y$ and $z$ axes, only three variables are independent and the other three are dependent [16]. In our design, the tripod rotation $\theta_{xt}$ about $x$-axis, the rotation $\theta_{yt}$ about $y$-axis, and the movement $z_t$ along $z$-axis are the independent variables. The other three, the movement $x_t$ along $x$-axis, the movement $y_t$ along $y$-axis, and the rotation $\theta_{zt}$ about $z$-axis are dependent variables. The three dependent variables are related to the three independent variables by the following equations:

$$x_t = \frac{\sqrt{3}}{3} L_p \cos \theta_{yt} \sin \theta_{zt},$$  

(2a) $$y_t = \frac{\sqrt{3}}{6} L_p (\cos \theta_{yt} \cos \theta_{zt} + \cos \theta_{xt} \sin \theta_{zt} - \sin \theta_{xt} \sin \theta_{yt} \sin \theta_{zt}),$$  

(2b) $$\theta_{zt} = \arctan \left( -\frac{\sin \theta_{xt} \sin \theta_{yt}}{\cos \theta_{xt} + \cos \theta_{yt}} \right),$$  

(2c)

where $L_p$ is the side length of the moving platform triangle. When a tool is mounted on the platform of the tripod, the position of the tool tip is expressed

$$x_{tt} = R_t [0, 0, L_t]^T + x_t,$$

(3)

where $x_{tt} = [x_{tt}, y_{tt}, z_{tt}]^T$ is the vector representing the tool tip; $x_t = [x_t, y_t, z_t]^T$ is the vector representing the center of the moving platform; $L_t$ is the tool length; and $R_t$ is the rotation matrix given as

$$R_t = \begin{bmatrix}
\cos \theta_{yt} \cos \theta_{zt} & -\cos \theta_{yt} \sin \theta_{zt} & \sin \theta_{yt} \\
\sin \theta_{xt} \sin \theta_{yt} \cos \theta_{zt} + \cos \theta_{xt} \sin \theta_{zt} & -\sin \theta_{xt} \sin \theta_{yt} \sin \theta_{zt} + \cos \theta_{xt} \cos \theta_{zt} & -\sin \theta_{xt} \cos \theta_{yt} \\
-\cos \theta_{xt} \sin \theta_{yt} \cos \theta_{zt} + \sin \theta_{xt} \sin \theta_{zt} & \cos \theta_{xt} \sin \theta_{yt} \sin \theta_{zt} + \sin \theta_{xt} \cos \theta_{zt} & \cos \theta_{xt} \cos \theta_{yt}
\end{bmatrix}. $$

(4)

Although $z_t$ does not appear in Eqs. (2a)–(2c), it does not mean that $x_t$, $y_t$, and $\theta_{zt}$ are completely independent of $z_t$. In fact, the allowable angles of $\theta_{xt}$ and $\theta_{yt}$ are functions of $z_t$.

Taking the derivative of Eqs. (2a)–(2c) with respect to time yields expressions for the velocities as

$$\dot{x}_t = -\frac{\sqrt{3} L_p}{3} (-\sin \theta_{yt} \sin \theta_{zt} \dot{\theta}_{zt} + \cos \theta_{yt} \cos \theta_{zt} \dot{\theta}_{zt}),$$  

(5a) $$\dot{y}_t = \frac{\sqrt{3} L_p}{6} \left( \sin \theta_{yt} \cos \theta_{zt} \dot{\theta}_{zt} + \cos \theta_{yt} \sin \theta_{zt} \dot{\theta}_{zt} - \sin \theta_{xt} \cos \theta_{zt} \dot{\theta}_{zt} - \cos \theta_{xt} \sin \theta_{zt} \dot{\theta}_{zt} \\
- \cos \theta_{xt} \sin \theta_{yt} \dot{\theta}_{zt} - \sin \theta_{xt} \cos \theta_{yt} \dot{\theta}_{zt} - \sin \theta_{xt} \sin \theta_{yt} \dot{\theta}_{zt} \right),$$  

(5b) $$\dot{\theta}_{zt} = -\cos^2 \theta_{zt} \left( 1 + \cos \theta_{xt} \cos \theta_{yt} \right) \frac{(\sin \theta_{yt} \dot{\theta}_{zt} + \sin \theta_{zt} \dot{\theta}_{yt})}{(\cos \theta_{xt} + \cos \theta_{yt})^2}. $$

(5c)

Similarly, expressions for accelerations can be obtained from Eqs. (5a)–(5c) as follows

$$\ddot{x}_t = -\frac{\sqrt{3} L_p}{3} \left( -\cos \theta_{yt} \sin \theta_{zt} \ddot{\theta}_{zt} - 2 \sin \theta_{yt} \cos \theta_{zt} \dot{\theta}_{zt} \dot{\theta}_{zt} \\
- \sin \theta_{yt} \sin \theta_{zt} \dot{\theta}_{yt} - \cos \theta_{zt} \sin \theta_{zt} \dot{\theta}_{zt} \ddot{\theta}_{zt} + \cos \theta_{zt} \cos \theta_{zt} \dot{\theta}_{zt} \dot{\theta}_{zt} \right),$$  

(6a)
\[
\ddot{y}_{ct} = \frac{\sqrt{3}L_c}{6} \left( \cos \theta_{ct} \cos \theta_{st} \dot{\theta}_{ct}^2 - \sin \theta_{ct} \sin \theta_{st} \dot{\theta}_{ct} - \sin \theta_{ct} \cos \theta_{ct} \dot{\theta}_{st} - \sin \theta_{ct} \sin \theta_{st} \dot{\theta}_{ct} \dot{\theta}_{st} + \sin \theta_{ct} \cos \theta_{ct} \dot{\theta}_{st} - \sin \theta_{ct} \sin \theta_{st} \dot{\theta}_{ct} \dot{\theta}_{st} + \sin \theta_{ct} \cos \theta_{ct} \dot{\theta}_{st} + \sin \theta_{ct} \sin \theta_{st} \dot{\theta}_{ct} \dot{\theta}_{st} + \sin \theta_{ct} \cos \theta_{ct} \dot{\theta}_{st} \right)
\]

\[
\ddot{\theta}_{ct} = -2 \tan \theta_{ct} \dot{\theta}_{ct}^2 - \cos^2 \theta_{ct} \left[ \frac{- \sin \theta_{ct} \cos \theta_{ct} \dot{\theta}_{ct} - \cos \theta_{ct} \sin \theta_{ct} \dot{\theta}_{ct}}{\cos^2 \theta_{ct}} \right] \frac{\sin \theta_{ct} \dot{\theta}_{ct} + \sin \theta_{ct} \dot{\theta}_{ct}}{\cos \theta_{ct} + \cos \theta_{ct}}
\]

\[
+ \frac{(1 + \cos \theta_{ct} \cos \theta_{ct})^2 (\sin \theta_{ct} \dot{\theta}_{ct} + \sin \theta_{ct} \dot{\theta}_{ct})}{(\cos \theta_{ct} + \cos \theta_{ct})^2}
\]

\[
+ 2 \frac{(1 + \cos \theta_{ct} \cos \theta_{ct}) (\sin \theta_{ct} \dot{\theta}_{ct} + \sin \theta_{ct} \dot{\theta}_{ct}) (\sin \theta_{ct} \dot{\theta}_{ct} + \sin \theta_{ct} \dot{\theta}_{ct})}{(\cos \theta_{ct} + \cos \theta_{ct})^2}.
\]

3.2. Tripod dynamics

Lagrange’s formulation is used for dynamic modeling of the 3-dof parallel robotic attachment. First, dynamic equations of the moving platform and the sliding-legs are formulated and then are assembled.

Assuming that the center of mass is coincident with the geometric center of the moving platform, the Lagrangian of the moving platform can be written as

\[
L_P = \frac{1}{2} \dot{x}_p^T M_P \dot{x}_p - m_p g^T h.
\]

where \( \dot{x}_p \) is a 6 \times 1 vector comprised of the linear velocity of the center of mass and the angular velocity of the moving platform, \( g \) is a 3 \times 1 gravitational acceleration vector, \( h \) is a 3 \times 1 vector representing the position of center of mass of the moving platform, and \( M_P \) is a 6 \times 6 inertia matrix defined by the moment of inertia of the moving platform \( I_p \) and its mass \( m_p \) as,

\[
M_P = \begin{bmatrix}
I_p & 0_{3 \times 3} \\
0_{3 \times 3} & m_p I
\end{bmatrix}.
\]

Here, \( I_p \) is the 3 \times 3 inertia matrix, \( 0_{3 \times 3} \) denotes a 3 \times 3 null matrix, and \( I \) denotes a 3 \times 3 identity matrix. The equations of motion of the moving platform can be obtained as

\[
M_P \ddot{x}_p + \mathbf{c}_P (\dot{x}_p, x_p) + \mathbf{g}_P = \mathbf{w}_P + \mathbf{w}_a
\]

where \( \mathbf{c}_P \) is a 6 \times 1 vector containing the centrifugal and Coriolis terms, \( \mathbf{g}_P \) is a 6 \times 1 vector representing the gravitational term, \( \mathbf{w}_a = [f_a^T, m_a^T]^T \), \( f_a \) and \( m_a \) are 3 \times 1 vectors representing the applied force and moment at the center of mass, respectively, and \( \mathbf{w}_P \) is a 6 \times 1 vector representing the constraint force term defined as

\[
\mathbf{w}_P = \begin{bmatrix}
\sum_{i=1}^{3} f_{bi}^c \\
\sum_{i=1}^{3} R_{bi} \times f_{bi}^c
\end{bmatrix}
\]

in which, \( f_{bi}^c \) is a 3 \times 1 vector representing the constraint force at the \( i \)th ball joint on the moving platform.

As for the sliding-legs, their Lagrangian can be written as

\[
L_s = \sum_{i=1}^{3} \left( \frac{1}{2} \dot{x}_i^T M_i \dot{x}_i - m_i g^T h \right),
\]
where $\mathbf{h}_i$ is a $3 \times 1$ vector representing the center of mass of $i$th leg, and $\mathbf{M}_i$ is the inertia matrix in terms of the moment of inertia $I_i$ of $i$th leg about its rotational direction and mass $m_i$ of $i$th leg,

$$\mathbf{M}_i = \begin{bmatrix} I_i & 0 \\ 0 & m_i \end{bmatrix}.$$  \hfill (12)

The equations of motion of the legs can therefore be obtained as

$$\mathbf{M}_S \ddot{\mathbf{x}}_S + c_S(\dot{\mathbf{x}}_S, \mathbf{x}_S) + \mathbf{g}_S = \mathbf{w}_S$$  \hfill (13)

where $\mathbf{x}_S$ is a $6 \times 1$ vector, $\mathbf{M}_S = \text{diag}(\mathbf{M}_i)$ is a $6 \times 6$ matrix, $c_S$ is a $6 \times 1$ vector containing the centrifugal and Coriolis terms, $\mathbf{g}_S$ is a $6 \times 1$ vector containing the gravitational term, $\mathbf{w}_S = [\mathbf{w}_1^T \mathbf{w}_2^T \mathbf{w}_3^T]^T$ is a $6 \times 1$ vector and $\mathbf{w}_i$ is a $2 \times 1$ vector defined by $\mathbf{f}_{1i}$ and $\mathbf{f}_{2i}$. $\mathbf{f}_i = f_i \mathbf{u}_i$ is the driving force of the $i$th actuator acting on the $i$th through the $i$th revolute joint.

The equations of motion of the entire tripod can be obtained by assembling Eqs. (9) and (13) as

$$\mathbf{M} \ddot{\mathbf{x}} + \mathbf{c}(\dot{\mathbf{x}}, \mathbf{x}) + \mathbf{g} = \mathbf{w},$$  \hfill (14)

where $\mathbf{M}$, $\mathbf{x}$, $\mathbf{c}$, $\mathbf{g}$ and $\mathbf{w}$ are given as,

$$\mathbf{M} = \begin{bmatrix} \mathbf{M}_p & \mathbf{0}_{6 \times 6} \\ \mathbf{0}_{6 \times 6} & \mathbf{M}_S \end{bmatrix}$$  \hfill (15)

$$\mathbf{x} = \begin{bmatrix} \mathbf{x}_P^T \\ \mathbf{x}_S^T \end{bmatrix}^T$$  \hfill (16)

$$\mathbf{c} = \begin{bmatrix} \mathbf{c}_P^T \\ \mathbf{c}_S^T \end{bmatrix}^T$$  \hfill (17)

$$\mathbf{g} = \begin{bmatrix} \mathbf{g}_P^T \\ \mathbf{g}_S^T \end{bmatrix}^T$$  \hfill (18)

$$\mathbf{w} = \begin{bmatrix} \mathbf{w}_P^T + \mathbf{w}_a^T \\ \mathbf{w}_S^T \end{bmatrix}^T$$  \hfill (19)

It is noted that $\mathbf{0}_{6 \times 6}$ in Eq. (15) denotes a $6 \times 6$ null matrix.

In the equations of motion of the tripod system obtained above, the total number of equations is 12, which is equal to the total number of force, i.e., 3 × 3 constraint forces of the ball joints on the moving platform and 3 driving forces from the actuators. If required, method such as Lagrangian multipliers [15] could be used to eliminate the constraint forces of the ball joints to reduce Eq. (14) to three in terms of the driving forces of the actuators. However, if both the driving forces of the actuators and the constraint forces of the joints are needed, Eq. (14) can be used to obtain a full set of solutions. In our research, the driving force is used for the control of the tripod, and study of the constraint forces is used for the joints design, machine structural design and machining ability design.

3.3. Tripod applications

The 3-dof tripod is designed as a programmable add-on device for the purpose of enhancing the capabilities of any machine by providing it with a more flexible range of motion. It can be attached to other machines or robots to not only add more degrees of freedom but also enable fine tuning for complex motion control, so as to upgrade low-end equipment for more sophisticated operations cost-effectively. Fig. 4 depicts one application when the tripod is attached to a three-axis CNC machine.

When the tripod is attached to a CNC machine as a spindle head, the actual movement by the CNC machine should take into consideration the movement of the tripod, which is

$$\mathbf{x}_d = \mathbf{x}_c + \mathbf{R}_c \mathbf{x}_{ji},$$  \hfill (20)

where $\mathbf{x}_d = [x_d, y_d, z_d]^T$ is the vector representing the desired position of the tool tip, and $\mathbf{x}_c = [x_c, y_c, z_c]^T$ is the vector representing the movement of the CNC machine.

Since the structure of a common CMM machine is similar to that of a common CNC machine, Eq. (20) is also valid for CMM machines. Fig. 5 demonstrates another tripod application as a work stage for CMM laser scanning. During the laser scanning, the part to be measured is fixed onto the moving platform of the tripod that can move the sample precisely toward different orientations.

Most industrial robots are designed with decoupled position and orientation kinematics. The position is usually controlled by the first three joints and the orientation is controlled by a wrist. Therefore, the equation for the robot position
may be expressed as
\[ x_r = f(\theta_1, \theta_2, \theta_3) \]  
where \( x_r = [x_r, y_r, z_r]^T \) is the vector representing the robot position, and the nonlinear equation \( f(\theta_1, \theta_2, \theta_3) \) can be readily derived as a function of three joints \( \theta_1, \theta_2, \theta_3 \) for given robot structures.

In terms of pitch \( \theta_xr \), roll \( \theta_yr \) and yaw \( \theta_zr \) angle representation, the robot orientation may be expressed as
\[
R_r = \begin{bmatrix}
\cos \theta_yr \cos \theta_zr & -\cos \theta_yr \sin \theta_zr & \sin \theta_yr \\
\sin \theta_xr \sin \theta_yr \cos \theta_zr + \cos \theta_xr \sin \theta_zr & -\sin \theta_xr \sin \theta_yr \sin \theta_zr + \cos \theta_xr \cos \theta_zr & -\sin \theta_xr \cos \theta_yr \\
-\cos \theta_xr \sin \theta_yr \cos \theta_zr + \sin \theta_xr \sin \theta_zr & \cos \theta_xr \sin \theta_yr \sin \theta_zr + \sin \theta_xr \cos \theta_zr & \cos \theta_xr \cos \theta_yr
\end{bmatrix}.
\]  
When the tripod is attached to an industrial robot, the entire tripod is treated as a tool for the robot. Therefore, the tip of the tool attached to the tripod is expressed as
\[ x_d = x_r + R_r x_{tt}. \]

4. Case study

The test bed used for this case study is a gantry system, which consists of an \( x \)-table to hold a workpiece and the tripod unit mounted on a \( y \)-table for a polishing application. It is first modeled by using the scene graph-based Java 3D as an alternative of camera images, with behavioral control nodes embedded. The tripod kinematics derived in Eqs. (2a)–(2c) is then coupled with the control nodes. During normal polishing operation, remote monitoring and control are facilitated by sensor signals and commands passing between the test bed and an end user.
4.1. Java 3D model creation

Java 3D is designed to be a mid to high-level fourth-generation 3D API [17], using scene-graph architecture with a direct parent-to-child relationship among nodes. Enabled by the scene-graph architecture, Java 3D provides an abstract, interactive imaging model for behavior control of 3D objects. Because Java 3D is part of the Java pantheon, it assures users ready access to a wide array of Java applications and network support functionality [18]. Fig. 7 illustrates (a) the test bed setup and (b) the Java 3D scene-graph architecture of the tripod test bed.

The scene graph given in Fig. 7(b) contains a complete description of the entire scene including the geometry data, the attribute information, and the viewing information needed to render the scene from a particular point of view. The Virtual Universe object provides a grounding for the entire scene graph. All Java 3D scene graphs must connect to a virtual universe object to be displayed. A scene graph itself starts with the BranchGroup (BG) nodes (although only one BG node in this case). A branch group node serves as the root of a sub-graph, or branch graph, of the scene graph. The TransformGroup nodes inside of a branch graph specify the position, the orientation, and the scale of the geometric objects in the virtual universe. Each geometric object consists of a Geometry object, an Appearance object, or both. The geometry object describes the geometric shape of a 3D object. The appearance object describes the appearance of the geometry (color, texture, material reflection characteristics, etc.). The behavior of the tripod model is controlled by Behavior nodes, which contain user-defined control codes and state variables. Sensor signal processing can be embedded into the codes as well for remote monitoring. Once applied to a transform group node, the so-defined behavior control affects all the descending nodes. In this example, the Moving Platform can be controlled by using a Moving Platform Control node, for
both on-line monitoring and control. As the 3D model is connected with the test bed through the control node by low-
volume message passing (real-time sensor signals and control commands transmissions, etc.), it becomes possible to
remotely manipulate the real tripod through its Java 3D model.

4.2. Tool trajectory planning

Tool trajectory planning for polishing is to generate discrete set of points for the tool mounting on the moving platform
of the tripod. The time interval between two commands is the control period of the tripod. The goal of the trajectory
planning is to plan a smooth trajectory for the tool to accurately follow the contour of a workpiece surface with full
coverage. To achieve this goal, the following global strategy is adopted. The tool follows a zig-zag path, and is driven along
the surface line by line in x direction. The stepover between two lines is chosen as 75 percent diameter of the tool-bit to
ensure the full coverage of the surface. At each line, the tool speeds up from zero to a given speed then remains constant
speed. At the end of the line, the tool-bit slow down and the final speed reaches zero at the last point. After each line, the
tool-bit moves to another line. The procedure repeats until the full coverage of the surface is achieved.

For the trajectory planning, a motion profile is selected first and the tool locations are determined accordingly based on
the workpiece surface to improve the accuracy. Since the motion profile is selected first, starting from the first point, the
distance of each control interval that the tool is supposed to travel is known as \( d_{\text{next}} \). Therefore, the next point could be
calculated by solving the equation below:

\[
\begin{align*}
    z &= f(x_0, y_i), \\
    (z_{\text{next}} - z_{\text{now}})^2 + (x_{\text{next}} - x_{\text{now}})^2 &= d_{\text{next}}^2.
\end{align*}
\]  

(24)

Since the motion is along \( x \)-axis, \( y_i \) remains constant. Iteratively solving the equation leads the positions for all the
points. Furthermore, when planning the trajectory for the five-axis test bed, the direction of the tool at each point could be
calculated as the norm of the surface at that point. More details can be found in [19].

4.3. Remote tripod manipulation

As shown in Fig. 2, our Java-based remote monitoring and control are conducted through a web user interface, which
communicates indirectly with the device controller via an application server. Whereas all registered users can do
monitoring simultaneously, only one user is authorized at a given time to manipulate the tripod. Currently, the same data
format is used in monitoring and control. A typical data packet is shown below:

\[
\begin{array}{cccccccc}
S_1 & S_2 & S_3 & x & y & z_t & \theta_{xt} & \theta_{yt} \\
\end{array}
\]

where \( S_{i=1,2,3} \) is the displacement of a sliding leg along its guideway; \( x \) and \( y \) are the coordinates of an \( x-y \) table; \( \theta_{xt} \) and \( \theta_{yt} \)
are the pitch and roll angles of the moving platform; and \( z_t \) is the center position of the moving platform along \( z \)-axis. This
packet format includes all information needed by both the Java 3D model and its physical counterpart, the tripod. The 3D
model ignores the first three numbers, while the tripod ignores the last three numbers. Compared with an 8-bit camera
image of 600 × 800 (480 KB) in one frame, our packet size is only 80 bytes—a significant size reduction suitable for web-
based real-time applications.

Fig. 8(a) shows a close-up of the test bed system. The tripod is mounted upside-down on a \( y \)-table above, whereas a
workpiece is placed on an independent \( x \)-table below. In our design, the three sliding-legs of the tripod are driven by three
24 V DC servomotors combined with three lead screws. Each actuator has a digital encoder (1.25 \( \mu \)m/count) for position
feedback. Fig. 8(b) depicts one snapshot of the Java-based and sensor-driven prototype system for remote monitoring and
control. A sampling rate of 100 Hz is used in this case study, and an average time delay for data communication and
kinematics calculation is less than 100 ms in our lab environment. This Java-based approach can also be applied to other
operations as shown in Figs. 4–6, showing promise of its application to distributed manufacturing.

5. Conclusions and future work

This paper presents a Java-based approach to remote real-time monitoring and control of a parallel robotic attachment.
To overcome the bottleneck of unpredictable network traffic, a real device is represented as a Java 3D model that can be
rendered off-site in an end user’s computer. The behavior of the 3D model is completely driven by the real sensor signals,
and can thus enable a Java model-based and sensor-driven monitoring and control. The significantly reduced network
traffic makes this approach practical as a web-based real-time application. Although Java 3D models are used to replace
camera images, an off-the-shelf web-ready camera can easily be switched on remotely to capture any abnormal scenes during an operation (polishing, deburring, etc.).

A special parallel robotic attachment—a tripod is designed as a toolhead that can be mounted onto other CNC machines, industrial robots or CMMs. The advantage of such a combination is the increased degrees of freedom and the fine tuning for complex motion control. Three tripod applications are introduced in the paper to show its flexibility as a toolhead. In addition, it can be used for standalone operations, such as polishing, after adding an independent x-y table. The results of the case study demonstrate the feasibility of our approach as a web-based real-time solution and its potential to distributed manufacturing.

In addition to real-time monitoring and control, it is envisioned that the Java-based approach can be extended to collaborative design, off-site inspection and trouble shooting, as well as virtual training and facility touring. In a web-based environment, it will find new ground and potentials in the future.

References