Proc IMechE Part B:
J Engineering Manufacture
1–17
© IMechE 2014
Reprints and permissions:
sagepub.co.uk/journalsPermissions.nav
DOI: 10.1177/0954405414534642
pib.sagepub.com
SAGE

Augmented reality–based programming, planning and simulation of a robotic work cell

Yun Suen Pai, Hwa Jen Yap and Ramesh Singh

Abstract

In this article, the development of an augmented reality–based robotic work cell is presented, consisting of a virtual robot arm, conveyor belt, pallet and computer numerical control machine that simulates an actual manufacturing plant environment. The kinematics of the robot arm is realized using Denavit–Hartenberg's theorem, which enables complete manipulation of the end-effector in three-dimensional space when interacting with other virtual machines. Collision detection is implemented in two areas, namely, modifiable marker–based detection for the robot arm, which detects nearby obstacles as well as integration with object manipulation to pick and place a virtual object around the environment. In addition, an augmented heads-up display overlay displays live information of the current system. The case studies suggest that the proposed system can simulate a collision-free operation while displaying the coordinates of the virtual object, current tool equipped and speed of the conveyor belt, with a percentage error of less than 5%.

Keywords

Augmented reality, robotic work cell, kinematics, collision detection, heads-up display, robot simulation

Date received: 22 January 2014; accepted: 14 April 2014

Introduction

Current implementation of robotics in engineering is not a novel approach as numerous industries have been utilizing automation to design and develop their products. However, the primary concern of robotics lies in the complexity in integrating and fully utilizing the capabilities of robots due to the fact that numerous aspects need to be considered such as programming methods, algorithms, path planning, display systems and kinematics. Owing to these requirements, small and medium enterprises (SMEs) tend to avoid incorporating robotics into their production processes, even though they are fully aware of the benefits offered by automated systems. Furthermore, SMEs avoid implementing robotics due to cost and time constraints as well as safety issues when programming robots on site. Augmented reality (AR) is a field of research, which is rapidly growing following the introduction of virtual reality (VR) that aims to fully integrate virtual and real environments to solve the above issues. VR utilizes a fully computer-generated environment, which is more costly and requires a higher degree of skills compared to AR. Even though AR has been present since the early 1990s, it is only recently that AR emerges as one of the forefront technologies due to the rise of popularity in smartphones and tablets. This indicates that AR can be applied in various research fields and consumer products. Engineers can simulate a manufacturing environment effectively by implementing AR in robotics, in which a scaled simulation of a robotic arm is possible. The integration of AR in robotics enhances a user's perception and visual senses while reducing costs associated with expensive prototype fabrication. In path planning, it is complex yet necessary to avoid collision and accidents. In addition, there is a lack of information feedback to the operators, which in turn increases the chances of errors as well as time consumption. AR solves these problems by clear visualization of collision detection and two-dimensional (2D) information overlay. Knowing the benefits of AR in robotics, the main objective of this study is to simulate

Department of Mechanical Engineering, Faculty of Engineering, University of Malaya, Kuala Lumpur, Malaysia

Corresponding author:
Hwa Jen Yap, Department of Mechanical Engineering, Faculty of Engineering, University of Malaya, 50603 Lembah Pantai, Kuala Lumpur, Malaysia.
Email: hjyap737@um.edu.my

an AR environment for use of a robotic work cell, apply marker-based collision detection as well as augment 2D information in a heads-up display (HUD) to enhance the user's visual, aural and proprioceptive senses.

Related studies

Manual work by humans is generally not as efficient and flexible in comparison to robots and is further inhibited by other limitations such as the randomness of walking worker activities[1] and fatigue. Hence, ergonomic studies must be taken into consideration.[2] However, robots require careful calibration as well as numerous testing and programming, which are achieved through simulations. There are two main programming methods currently available for industrial robots, namely, online and offline programming. Online programming basically requires the operator to be within the working envelope of the robot, which is undesirable due to safety reasons. Conversely, offline programming uses a remote computer console, in which the input program is translated into robotic language. An AR-assisted robot simulation is considered to be a form of offline programming, which requires functions and precise positional values.[3] Positioning and path planning involve computing a continuous order of configurations between a starting point and goal point. Motion or trajectory planning can be clearly distinguished from path planning since the path in motion planning is constrained by time. This means that computing a collision-free trajectory between the initial and end points requires both precise trajectory and path planning. Planning involves actions taken by the robot in order to avoid obstacles and complete the programmed path. All actions taught to the robot are then transferred to the controller for execution.

Kinematic modelling of an industrial robot is a vital part of this study as the main aim is to manipulate a robotic arm effectively in AR environment. Kinematic analysis can be classified as direct kinematics and inverse kinematics. Direct kinematics is the process of searching and calculating the position and orientation of the end-effector from the given joint position whereas inverse kinematics gives the final position of the end-effector whereby the position for each joint needs to be determined. The Denavit–Hartenberg (D-H)-based method is the most popular approach for kinematic analysis[4] and is used for the KUKA robotic arm manipulator in this study. The mathematics and notations of the robot's forward kinematics can be best determined through D-H method regardless of its sequence or complexity, and the location of the joint is dependent on the previous joint's location, which can be calculated using transformation matrices. Transformation can be represented by Cartesian, cylindrical, spherical, Euler or roll–pitch–Yaw (RPY) coordinates. Another kinematic analysis method was developed in 1876 and is known as the screw theory.[5] Screw represents two kinds of motion, that is, translation and rotation, which can be converted from one another. The theory is expressed in Plücker coordinates, which comprise angular and linear velocities. Another kinematic study carried out by Rocha is differential kinematics, which has a similar approach and utilizes Jacobian matrix to map joint velocities to the end-effector velocity in order to implement both D-H and screw theory methods. The screw theory offers an advantage over the D-H method in modelling and analysis such that it is more flexible. However, the screw theory is more complex and difficult to implement.

AR is a technology capable of blending three-dimensional (3D) virtual objects to the real world seamlessly in order to visualize the D-H model of the robot and observe its interaction with the environment. AR is a subject of intensive studies and perceived as one of the 10 emerging technologies.[6] The two main activities of AR are tracking and registration. Markers in the real environment relative to the camera (also known as fiducials) are tracked to obtain position and orientation data. The tracking values are utilized during registration to superimpose the 3D virtual object onto the real environment. ARToolKit is a marker-tracking library implemented for AR applications and was developed by Hirokazu Kato in 1999. ARToolKit is used as the tracking system in this study. One study was focused on the tangible user interface (TUI) aspect of ARToolkit via a prototype system used for interior design,[7] which includes a physical paddle that uses transparency cues in an AR interface for team collaboration work. Even though the prototype is relatively simple, it exhibits a potential, which can be further expanded such as its implementation in this study through full 3D object manipulation. In fact, the marker-tracking system supports robot programming in an interesting manner. Robot programming was developed recently using the AR (robot programming using augmented reality (RPAR)) system, which enables movement of a virtual robot either by the number of start-goal paths or enabling the end-effector to follow a pre-defined path.[8] A collision-free volume (CFV) aids the system and constrains the movement of the robot simultaneously. Although these methodologies are proven to be reliable for their intended tasks, a user-defined path is comparatively more flexible. A user-defined path enables the user to move an object, which generates a path automatically for the robot to follow and this approach is adopted in this study. However, it shall be noted that ARToolKit is incapable of handling 3D computer graphic models, and thus, a third party computer graphics rendering software needs to be employed.

OpenGL is a 3D graphics and modelling library that has a wide application programmer's interface (API) and includes a number of commands and functions. The programmer can create 3D graphics consisting of points, lines and primitive polygons, create inputs for display as well as manipulate graphics with OpenGL.[9] In addition, OpenGL is capable of handling complex modelling transformation techniques, and therefore, an interactive 3D simulation system can be built for robot path planning. A recent study demonstrated a robot collision avoidance simulation achieved via OpenGL.[10] The fully realized 3D virtual environment displays the robot's capability to detect collision in an axis-aligned bounding box (AABB) collision, which is quite promising. Implementing such a technology with AR will indeed create a more immersive system, which is what ARToolKit is capable of achieving. Furthermore, OpenGL can be supported by all platforms such as Windows and Macintosh operating systems.[11] This study utilizes an OpenGL application interface written in C++ programming language supported by the OpenGL library to generate virtual contents in the AR scene with collision detection.

The final key component in the AR simulation system is a display, which enables the user to view the scene. Display systems are available in various shapes and sizes, and a head-mounted display (HMD) is a hardware, which is typically utilized to exploit an AR generated environment. An early study focused on a development of a method for calibrating see-through HMDs using a dynamic framework that is also applicable for stereo HMDs.[12] However, there are several limitations associated with the use of see-through HMD such as constrained transformations and increased errors due to camera movement. A stochastic solution was presented to calibrate a see-through HMD for the application of AR and it was proven that the stochastic model of commonly applied mathematical models is stochastically incorrect.[13] Conversely, stochastically consistent solutions are possible. Flock of Birds (FOB) has been used for the tracking system, and it is a standalone system consisting of a one-foot cube transmitting antenna, an extended range controller (ERC) and two receivers. Virtual points on the HMD screen are assigned to points in the real environment during the calibration process, while taking into account pixel accuracy and coordinate systems. It shall be highlighted that there are no studies available in the literature that provide a systematic comparison between various calibration methods of an AR system through see-through HMDs. Hence, it is evident that an intensive calibration is required to minimize errors for see-through HMDs. In addition, the ergonomics aspect of HMDs needs to be considered as they are typically bulky and uncomfortable to wear, even over a short period. In this study, a typical Windows laptop monitor is used as the display system, as it can be used to run the required applications. Since the system is used primarily for robot programming, the hardware involved is stationary, which eliminates unnecessarily complicated calibration problems associated with HMDs.

Recent advancements

Several studies were carried out in the past 2 years on the methods of implementing AR in robotics. A recent study showed that marker-tracking requires accurate pose estimation, which is dependent on camera distance and viewing angle, even though it is more accurate than markerless ones.[14] Pati et al.[14] proposed a method to model errors based on scaled unscented transform (SUT), which is a new function for estimating nonlinear transformation. This gives a more accurate calibration since the method is independent of the image sequences, and multiple markers are not required. In comparison, ARToolKit uses a calibration dot marking system with multiple images of 6×4 dot patterns captured at different angles. An accurate calibration leads to accurate robot simulation, and a recently developed prototype system is able to control a virtual robot in a real environment.[15] The system architecture used in Girbacia et al.[15] bares resemblance to the study in this article, with the exception that it does not cover a work cell interaction. The system is also marker-based which differs from another study that uses markerless gestures.[16] Markerless system is achieved through a 3D motion tracking system for depth perception, coupled with a 2D camera. Bare-hand manipulation is seemingly more natural especially in pick-and-place tasks due to the gripping gesture, and the data are transmitted to an industrial robot controller. Spatial robot programming is a form of online programming by demonstration (PbD), even though the AR module aids in defining the program. Although spatial robot programming is quite intuitive, it is rather costly due to the fact that depth sensors are expensive simulation tools. Operations such as welding shows potential for a system such as this, and the ability to verify the simulation system appropriately is vital to provide relevant information for further development.[17] The recently developed half-silvered glass works similarly to an optical see-through display, with the exception of its aspect ratio and size of the display monitor. Superimposition of virtual robots becomes much more intuitive and realistic. Even though software communication is still poor in the present, the benefits offered are promising if the system can be scaled to fit an entire work cell or placed on a computer numerical control (CNC) machine.

Path generation and manipulation in AR

A planned set-up is typically required to implement AR. A novel approach towards planning a collision-free path in an unprepared environment was presented in Ong et al.,[18] taking into account the advantages of utilizing AR in industrial robot programming. The end-effector is the main concern when an industrial robot is targeted for a simulation process, and therefore, its movements are constrained along a visible path and supported by a piecewise linear parameterization (PLP) algorithm. This algorithm is essentially used to parameterize the data points by adopting an interactive generated piecewise linear approximation. Furthermore, the data points are used to generate 3D parametric curves by applying Bayesian neural networks and reparameterization. Reparameterization is used for fitting problems, in which the parameters of the data points are updated after each iteration. A CFV is generated once the curve is obtained and is used to verify whether the end-effector collides with obstacles along the curve. The AR transformation matrix will accurately position the orientation of the end-effector of the robot. CFV is essentially generated from the swept volume of a sphere and is represented as a mass of virtual spheres intersecting one another. AR environment is primarily used to provide visual feedback, which improves interaction between the user and real world directly, which is also part of the PbD approach. Since a virtual industrial robot will be placed into the real environment through AR, proper planning is crucial in order to accurately manipulate the 3D movement path in an AR work cell.

A path editing method is required to evaluate the movement path of a 3D object in an AR scene.[19] In general, there are three forms of path manipulation, whereby the first one involves utilizing a third party commercial software and loading using an AR toolkit. The second form of path manipulation involves extending commercial software in which an AR plug-in module is connected to a conventional commercial software, whereas the third form involves using a TUI method. In the TUI method, the user can plan a path by hand movements and confirm the results in the AR environment. The third method is chosen in this study due to its interactive nature. However, this method may generate temporal errors such as trembling of the user' hand. In this method, the translation point of the object is first examined in order to select a point as the control point. Splines are then used to reconstruct the path into a smooth line. The Catmull–Rom spline is used since it has fewer extensions compared to other conventional splines, and it can be easily manipulated with control points. The control points are generated immediately as long as the user presses and holds down a button of the object or prop. A mouse pen is used as the input device in this study and the movement path is constructed until the button is released. This method differs from conventional graphical user interface (GUI)-based path manipulation that emphasizes the use of mouse and keyboard input. Since there are ongoing studies on the increasing interactiveness of path manipulation, collision-free-path programming is also studied in the same aspect. The utilization of heuristic beam search algorithm for path generation coupled with AR environment was analyzed by Chong et al.[20] The system architecture involves markers to input inverse kinematic modules that calculate the coordinates of the virtual robot. The forward kinematic module calculates the end-effector's reference point. OpenGL Renderer is used to portray images through a HMD for each video frame, which serves as feedback. The force module then calculates the work done by the robot and the value is used for path cost. They discovered that the size of the beam does not necessarily need to be large for a more robust result and the ARToolKit is sufficient to demonstrate the proposed methodology. The ability to snap the control point onto the physical prop is useful in robot path manipulation, and a similar feature is studied in this article.

Recent studies are similar in a way that the robot path manipulation system enables the user to create a list of control points and these points are used to generate a ruled surface.[21,22] The set-up involves a robot arm mounted with an end-effector, a computer, monitor, camera and an interactive device strapped to a marker cube. A fully AR-controlled environment with effective orientation planning is possible with the utilization of an ARToolKit-based tracking and interaction module, as well as path planning, path optimization and simulation modules. Although a series of control points within the CFV are created so that a path can be formed, there is an issue that the path may not be collision free. Therefore, a Euclidean distance-based method that computes the distance between the probe, and each control point is used to generate the path through interpolation, making it collision free in the joint-space. The end-effector orientation is represented relative to the robot's base frame by a unit vector. The path is then optimized using a convex optimization technique, which is the log-barrier method. The Newton–Euler algorithm is used to model the dynamic properties of the robot's behaviour and the trajectory is simulated using a discrete proportional–derivative (PD) control scheme. The developed system further proves that ARToolKit is more than capable for robot path planning. This paper also demonstrates that ARToolKit is excellent for robot path planning as verified by KUKA Sim Pro, which is a commercially available simulation software.

AR application in work cell

The concept, which renders virtual manufacturing very appealing, is the idea of having a factory sitting on your desk, with total control over all aspects of production.[23] Automotive companies such as Mercedes Benz have adopted virtual manufacturing as they are convinced that it is an effective engineering tool. Each Mercedes line has three types of machines, namely, 'Trallfa', 'Devilmat Spray Mate' and ESTA for interior painting, metallic painting and electrostatic painting, respectively. These are painting programs from the RobCad's virtual manufacturing software and have been proven to be highly efficient compared to importing data from computer-aided design (CAD) drawings into a PC. Such offline programming method results in 30%–40% gain in time, which increases productivity. A more streamlined approach can be achieved for communication and partnership between manufacturing organizations by implementing virtual manufacturing as the main production tool. Even though virtual factory layout planning (also known as digital manufacturing technology) may still be uncommon for most enterprises, those who have chosen to adopt virtual factory layout planning perceive it as a way to encourage parallel processing, decrease cycle time and improve precision. One of the methods that can be used to construct a mixed reality manufacturing environment is image-based tracking.[24] Image-based tracking does not involve the use of markers, and tracking is done by finding an arbitrary feature in the real environment such as a safety sign. The safety sign acts as a marker and acquires a fixture on the site. However, it shall be noted that not all kinds of feature are suitable for use as real-life reference markers and a circular contour shape, which is typical of safety signs, is analysed in Lee et al.[24] The Hough transform method, direct least square fitting and moment of inertia optimization method are among the methods used to analyse an ellipse, which pertains to the appearance of a circle at certain angles. The virtual manufacturing system software Delmia is used to apply 3D CAD geometry data of each equipment and machine for process planning. V-Collide library is used for collision detection such that a flag will be created at the collided parts in the event of a collision. However, accuracy is an issue because safety signs are not designed specifically for camera tracking and recognition, unlike fiducial markers.

Comparison with VR

Although AR application is the main focus of this study, a comparison with VR will not only produce valuable data for ongoing studies, but allows ample room for improvement in the field of visualization. VR replaces reality, in which the user is totally immersed in an artificial world and all surroundings are rendered through a computer. Meanwhile, AR enhances or modifies reality by adding minimum virtual objects and requires high accuracy in tracking scenery and environment. Both VR and AR have been growing rapidly over the past few years with continuous improvements in variety, functional capabilities and usability.[25] The multitude of technologies include computer assisted virtual environment (CAVE) environments, power walls, holographic workbenches, HMDs and sensors. The integration of VR into CAD (called VR-CAD/E technology) has garnered much attention in scientific research, in which the aim is to overcome issues such as integrating VR with commercialized CAD software tools, modelling and tooling applications. The challenges in integrating VR into CAD lie in the limitations in current technology. A number of studies have been devoted on addressing the above challenges, in which new AR approaches, advanced VR solutions and extensions of CAD/E systems are proposed. For instance, studies on AR are focused on product development process by using augmented technical drawing (ATD),[26] interfacing AR with custom-built 3D applications and immersive modelling system (IMMS) for interface design.[27] In contrast, studies on VR analysis are centred on VR tools for product life cycle management (PLM) systems,[28] haptic feedback system and simulation, SIMUCAL virtual simulator for analysis of footwear comfort[29] as well as VR-based modular fixtures. Studies that integrate VR into CAD/E systems stress on immersive interaction and haptic paradigms for CAD, bi-manual 3D input for CAD modelling and human overall performance in multi-modal CAD.[30] The common factor for all VR integration is that cost is substantially higher compared to AR, which results in the need for higher computational power.

Simulation has been proven to be a significant research tool towards the continuous development of various engineering industries, especially robotic systems, due to effective manipulation of the end-effector while accounting for limited floor availability as well as operator safety. Therefore, AR is the subject of much research due to its robust application in simulating a manufacturing environment, with its seamless interfacing between real and computer-generated content. However, there are still limitations of AR even with today's technology, especially with regard to fully integrating AR with robotics, display systems, sensors and path programming. It is evident that there are many areas for improvement in order to realize the goal of a fully digitalized factory with increased accuracy, registration and latency. One of the factors that need to be considered for robotic applications is the ability of the robot to recognize its path and effectively avoid collision. Collision avoidance is crucial since collision may result in injuries to nearby operators and increase maintenance costs for the robot arm. Studies have shown that there are several ways to prevent collision, either by developing a complex algorithm, generating a CFV or utilizing commercial software. These methods are challenging and may be inflexible for use in various environments. Hence, a marker-based method is devised in this study in order to develop a simpler recognition system. In this system, markers are placed at specific points, which will be recognized by ARToolKit as an obstruction in order to avoid collision effectively. Finally, a display system is mandatory to view the effects of AR on the system. However, a feature that portrays information will indeed be more useful for engineers. Even though various display systems can be used, there only a few systems available that can analyze and provide information feedback of current operations. Hence, this study is aimed to provide accurate 2D information using a heads-up display with proper calibration.

Methodology

In this study, C++ programming using ARToolKit is used to create a running program that generates AR content through a marker-based tracking method. The program includes path drawing, whereby a line can be drawn with two vertices. The marker probe acts as a teach pendant in order to determine the location of the vertex and needs to be fabricated out of paper and cardboard. The full kinematics of the robot is then studied. Forward kinematics is first studied to understand how the coordinates of the end-effector are determined, followed by inverse kinematics in which the joint variables are determined based on the location of the vertices. The entire formulation is carried out by programming in order to combine line drawing and kinematics effectively for simulation. A full work cell is added to the virtual environment when the AR robot functions as desired. The work cell consists of a conveyor belt, CNC machine and pallet, which are called by other static markers. HUD is added by programming in order to provide a more effective information feedback, which shows the speed of the conveyor, current tool equipped to the machine, as well as coordinates of the manipulated object. The object serves as a visual cue to show how a full operation is done in the work cell and requires collision detection and object manipulation algorithms in the code. The combination of these elements creates a fully functional AR robotic work cell with a pick-and-place system. However, the work cell is not merely constrained to perform this operation.

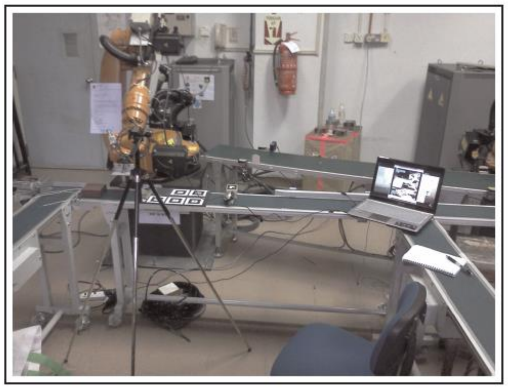


Figure 1. Experimental set-up at Robotics Laboratory.

The experimental set-up comprising hardware and software is relatively low cost and can be constructed virtually anywhere provided that there is proper lighting and sufficient space. The initial testing of the program is carried out by a simple set-up, which consists of a laptop, Webcam and 'Hiro' marker. The experimental set-up is then transferred to the Robotics Laboratory, Department of Mechanical Engineering, University of Malaya, after the program is compiled. A photograph of the experimental is shown in Figure 1. The main purpose is to position the virtual robot over the physical one in order to demonstrate the feasibility of running a real work cell in the laboratory. The layout plan is measured prior to the experimental set-up as this will influence the accuracy of the results.

Positioning of markers and camera is also crucial. The markers must be placed in a way that they are clearly visible to the camera under direct lighting. The camera must be positioned at a suitable height as higher locations will result in blurry images for the markers. An additional marker (labelled as 'Star') is added for the purpose of a case study. The 'Hiro' marker serves as a global origin, in which every other marker's position is calculated relative to the 'Hiro' marker. The 3D marker cube is used as the teach pendant because its function resembles that of an actual teach pendant of the KUKA robot, whereby it can move the robot arm around and save the coordinates of a point in space. This means that the position of the tip of the teach pendant is calculated relative to the 'Hiro' marker and is registered whenever one of the four faces are viewable. The primary reason the teach pendant is constructed as a marker cube is so that it is viewable to the camera regardless how it is held. The set of static markers as well as marker cube teach pendant are shown in Figure 2 and the overall system architecture is shown in Figure 3.

Kinematic modelling

Each compartment of the robot is first assigned a coordinate frame. The 6-DoF KUKA robot[31] and the

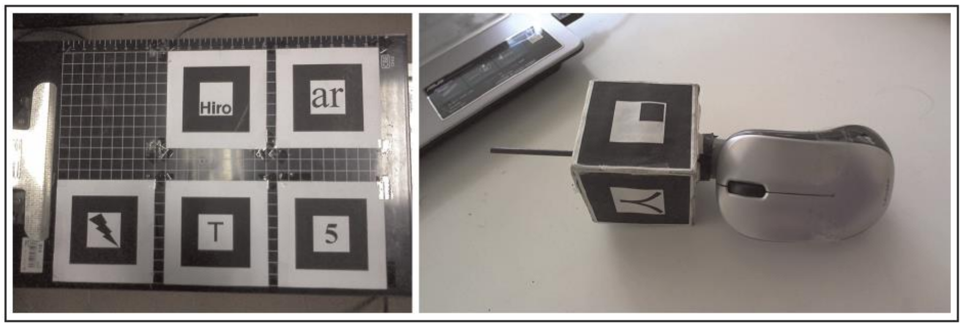
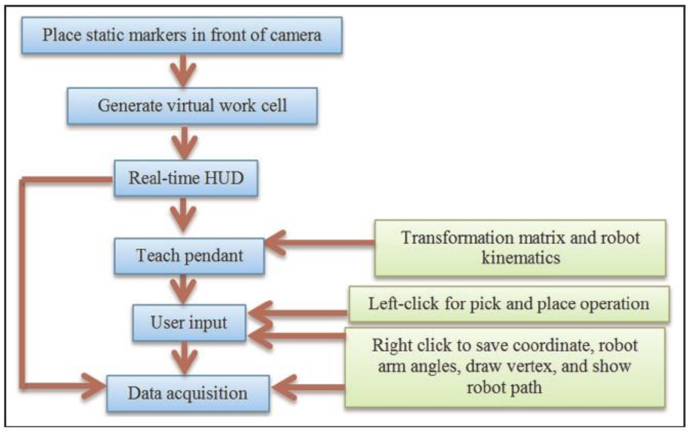
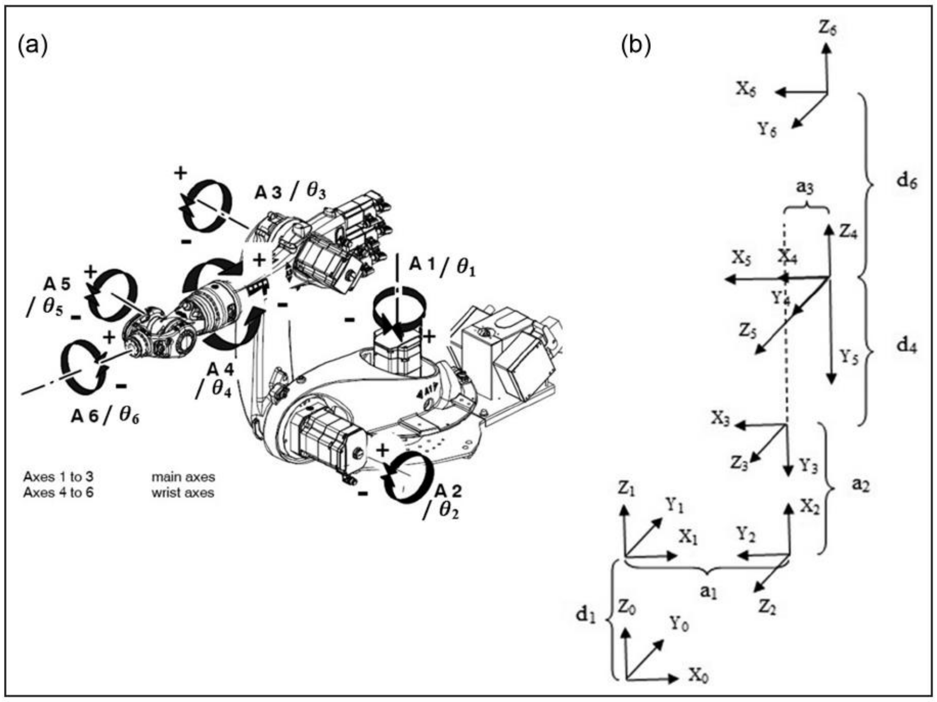


Figure 2. Set of static markers (left) and marker cube teach pendant (right).



Place static markers in front of camera
Generate virtual work cell
Real-time HUD
Teach pendant
Transformation matrix and robot kinematics
User input
Left-click for pick and place operation
Right click to save coordinate, robot arm angles, draw vertex, and show robot path
Data acquisition

Figure 3. Overall system architecture.



(a)
A 1 / θ_1
A 2 / θ_2
A 3 / θ_3
A 4 / θ_4
A 5 / θ_5
A 6 / θ_6
Axes 1 to 3 main axes
Axes 4 to 6 wrist axes
(b)
Z_6 X_6 Y_6 d_6 a_3 Z_4 X_5 X_4 Y_4 Z_5 Y_5 d_4 X_3 Z_3 Y_3 X_2 a_2 Z_1 Y_1 X_1 Y_2 a_1 Z_2 d_1 Z_0 Y_0 X_0

Figure 4. (a) KUKA robot 6-DoF. KUKA industrial robots, http://www.kuka-robotics.com/res/sps/e6c77545-9030-49b1-93f5-4d17c92173aa_Spez_KR_6_KS_en.pdf and (b) robot configuration with D-H frame.

assigned D-H frame are shown in Figure 4. The top surface of the pedestal, which is a solid cube onto which the robot is mounted, acts the world coordinate or point (0, 0, 0) of the robot. It shall be noted that angle A, B and C rotates about the *z*-axis, *y*-axis and *x*-axis, respectively.

The D-H coordinate frame consists of mainly four parameters (a, α, θ, d), which represents the link length, link twist, joint angle and link offset, respectively. The link length and link twist are the link parameters that describe the relative position between joints. The joint angle will be variable if the joint is revolute.

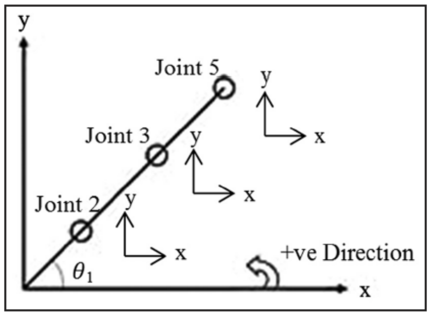


Figure 5. Calculation of θ_1 in X-Y plane, rotation about Z-axis.

Angle θ_1. Joint 1 is a twist joint, and therefore, it changes the orientation of its subsequent frame while the position of the subsequent frame is kept constant. In fact, there are three twist joints present, namely, Joints 1, 4 and 6. From calculations, only the *x*- and *y*-axes, as well as the length of the joints, are considered.

Angle θ_1 can be determined from the position of Joint 5, as shown in Figure 5. Frames 4 and 5 share the same position but different orientations, and hence, $\overrightarrow{p_{04}}$ and $\overrightarrow{p_{05}}$ have the same coordinates. It is assumed that the end of the arm will always face downwards, parallel to the floor. Therefore, the robot link will always form a straight line when viewed from the top. Thus, the position of $\overrightarrow{p_{04}}$ can be obtained using the following equation

$$\begin{pmatrix} p_{04,x} \\ p_{04,y} \\ p_{04,z} \end{pmatrix} = \begin{pmatrix} p_x \\ p_y \\ p_z \end{pmatrix} + \begin{pmatrix} 0 \\ 0 \\ d_6 \end{pmatrix} \tag{1}$$
$$\theta_1 = \arctan 2(p_{04,y}, p_{04,x})$$

The *arctan2* function was used in this case rather than *arctan* since *arctan2* represents angles in a different quadrant.

Angle θ_3. In this case, Link 4 is different from other links, as illustrated in Figure 6. This link is a special L-shaped link attached to Joint 3. An offset line parallel to Link 4 is required in order to determine θ_3. This angle can be solved from angle φ using the law of cosine. $|p_{24}|$ represents the opposite length of the new triangle formed and is obtained by determining the position of Joint 2 ($\overrightarrow{p_{02}}$) and Joint 5 ($\overrightarrow{p_{05}}$) with reference to the base. The position of Joint 2 is obtained from forward kinematic transformation (${}^0_2T$), whereas the position of Joint 5 ($\overrightarrow{p_{05}}$) shares the same position as that for Joint 4 ($\overrightarrow{p_{04}}$). The difference between these two positions will yield $\overrightarrow{p_{24}}$ and thus, the hypotenuse $|p_{24}|$ can be determined.

$$\begin{pmatrix} p_{24,x} \\ p_{24,y} \\ p_{24,z} \end{pmatrix} = \begin{pmatrix} p_{04,x} \\ p_{04,y} \\ p_{04,z} \end{pmatrix} - \begin{pmatrix} p_{02,x} = c_1 a_1 \\ p_{02,y} = s_1 a_1 \\ p_{02,z} = d_1 \end{pmatrix} \tag{2}$$

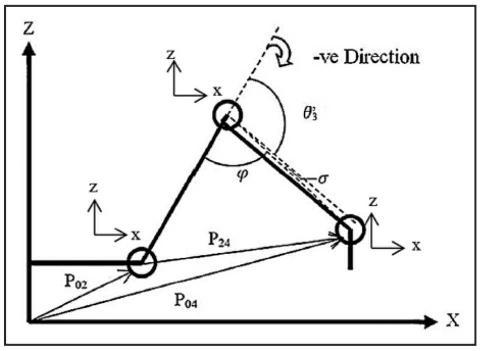


Figure 6. Calculation of θ_3 in X–Z plane, rotation about Y-axis.

$$|p_{24}|^2 = p_{24,x}^2 + p_{24,y}^2 + p_{24,z}^2 \tag{3}$$

The L-shaped link forms a small degree of difference and can be solved using the *arctan* function

$$\sigma = \arctan(a_3, d_4) \tag{4}$$

The link, which connects Joints 3 and 5, is not d_4. Rather, it is the hypotenuse of a right angled triangle (d'_4) with the edges d_4 and a_3. Hence, Pythagoras' theorem can be used to determine d'_4 as follows

$$d'_4 = \sqrt{a_3^2 + d_4^2} \tag{5}$$

Applying the cosine law gives

$$\varphi = \arccos\left(\frac{a_2^2 + (d'_4)^2 - |p_{24}|^2}{2 \cdot a_2 \cdot d'_4}\right) \tag{6}$$

$$\begin{aligned} \theta'_3 &= \pi - \varphi - \sigma \quad \text{(magnitude)} \\ \theta'_3 &= -(\pi - \varphi - \sigma) \quad \text{(direction)} \end{aligned} \tag{7}$$

$$\theta_3 = \theta'_3 + \frac{\pi}{2} \tag{8}$$

Angle θ_2. From Figure 7, it can be seen that angle θ_2 can be calculated using angles β_1 and β_2. Starting from angle β_1, $\overrightarrow{p_{24}^{(2)}}$ (which is the vector $\overrightarrow{p_{24}}$) is used referenced to Frame 2. From forward kinematics, it can be noted that $\overrightarrow{p_{24}^{(2)}}$ consists of only two components (*x* and *y*) as it is not a twist joint and will always form a right angled triangle. Hence, Pythagoras' theorem is used to solve for the edges of the triangle

$$\begin{pmatrix} \overrightarrow{p_{24,x}^{(2)}} \\ \overrightarrow{p_{24,y}^{(2)}} \\ \overrightarrow{p_{24,z}^{(2)}} \end{pmatrix} = \begin{pmatrix} p_{24,z} \\ \sqrt{p_{24,x}^2 + p_{24,y}^2} \\ 0 \end{pmatrix} \tag{9}$$

$$\beta_1 = \arctan\left(\overrightarrow{p_{24,x}^{(2)}}, \overrightarrow{p_{24,y}^{(2)}}\right) \tag{10}$$

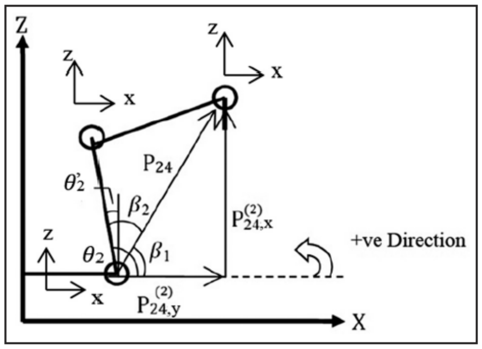


Figure 7. Calculation of θ_2 in X–Z plane, rotation about Y-axis.

The law of cosines is applied for the second triangle in order to obtain β_2

$$\beta_2 = \arccos\left(\frac{a_2^2 + \overrightarrow{|p_{24}|}^2 - (d'_4)^2}{2 \cdot a_2 \cdot \overrightarrow{|p_{24}|}^2}\right) \tag{11}$$

There are two solutions for θ_2 as follows:

Case 1. If β_1 is negative and β_2 is positive

The direction of 90° rotation is clockwise from home position and is therefore negative. If a part of Link 3 remains on top of *x*-axis of Joint 2 and the position of the end of arm moves downwards having negative β_1, only angle β_2 is required to define θ_2

$$\begin{aligned}\theta'_{2.1} &= \frac{\pi}{2} - \beta_2 \quad \text{(magnitude)}\\ \theta'_{2.1} &= -\left(\frac{\pi}{2} - \beta_2\right) \quad \text{(direction)}\\ \theta_{2.1} &= \frac{\pi}{2} + \theta'_{2.1}\end{aligned} \tag{12}$$

Case 2. Else

θ'_2 will always be equal to the summation of the angles β_1 and β_2 (i.e. $\beta_1 + \beta_2$) and is subtracted from 90°. The solution for θ_2 as in D-H is then added with 90°. Hence

$$\begin{aligned}\theta'_{2.2} &= \frac{\pi}{2} - (\beta_1 + \beta_2) \quad \text{(magnitude)}\\ \theta'_{2.2} &= -\left(\frac{\pi}{2} - (\beta_1 + \beta_2)\right) \quad \text{(direction)}\\ \theta_{2.2} &= \frac{\pi}{2} + \theta'_{2.2}\end{aligned} \tag{13}$$

Angle θ_4

$$\theta_4 = 0$$

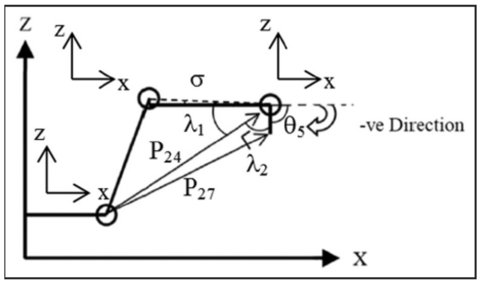


Figure 8. Calculation of θ_5 in X–Z plane, rotation about Y-axis.

Since θ_4 is a twist joint, this angle does not change the consecutive joint's coordinate. It is assumed that the orientation of Link 4 is constant throughout this study, and therefore, θ_4 is 0.

Angle θ_5. The law of cosine is applied twice for the two triangles in order to find θ_5, as shown in Figure 8. λ_1 can be easily determined using the following equation by substituting the length of the triangle (d'_4) obtained previously

$$\lambda_1 = \arccos\left(\frac{(d'_4)^2 + |p_{24}|^2 - a_2^2}{2 \cdot (d'_4) \cdot |p_{24}|}\right) \tag{14}$$

Following this, the length connecting from Joint 2 to Joint 7 (λ_2) needs to be determined. Knowing that the vector is equal to the position of the end of arm minus with the position of Joint 2 referenced to the base, the length $|p_{27}|$ can be obtained easily as follows

$$\begin{pmatrix} p_{27,x} \\ p_{27,y} \\ p_{27,z} \end{pmatrix} = \begin{pmatrix} p_x \\ p_y \\ p_z \end{pmatrix} - \begin{pmatrix} p_{02,x} = c_1 a_1 \\ p_{02,y} = s_1 a_1 \\ p_{02,z} = d_1 \end{pmatrix} \tag{15}$$

$$|p_{27}|^2 = p_{27,x}^2 + p_{27,y}^2 + p_{27,z}^2 \tag{16}$$

The law of cosines is applied to solve for λ_2 using the length obtained using equation (16). The direction of rotation for angle θ_5 moves downwards from the home position and therefore has a negative sign

$$\begin{aligned}\lambda_2 &= \arccos\left(\frac{|p_{24}|^2 + d_6^2 - |p_{27}|^2}{2 \cdot |p_{24}| \cdot d_6}\right)\\ \theta_5 &= -(\pi - (\lambda_1 - \sigma) + \lambda_2)\end{aligned} \tag{17}$$

Angle θ_6

$$\theta_6 = 0$$

θ_6 remains zero throughout the equation. This angle is dependent on the type of end-effector used. The position of the end of arm is unaffected.

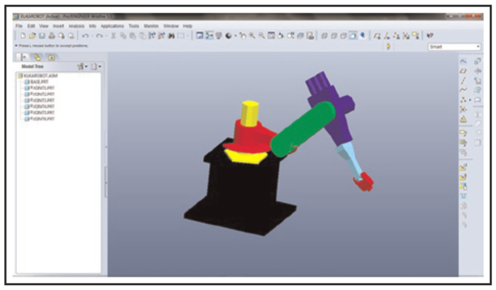


Figure 9. CAD model of the robot sketched using Pro-Engineer software.

CAD models

The actual 3D CAD model of the KUKA robot is rather complex, and therefore, a simplified version is used, while adhering to the kinematics study of the robot. Pro-Engineer software is used to sketch the robot, which is drawn to scale and follows the dimensions of the real KR 16 KS model, as shown in Figure 9. This means that each joint of the robot must be sketched individually to scale. The parts are then assembled to create the full robot arm in the 3D model in which each joint can be moved when they are dragged. The black base represents the pedestal. However, it shall be noted that the assembly file cannot be imported into the program, and thus, each joint needs to be individually imported as an STL file and re-assembled and rendered using OpenGL. Hence, the placement of the origin of the coordinate system for each joint needs to be accurate in order to reduce complications when assembling parts in the program.

Three additional parts are added in order to create a robotic work cell. Since it is required that the robot must interact and perform operations with other machines, a CNC machine, conveyor belt and pallet are added to the robot. The basic operation performed by the KUKA robot is pick and place, which is highly dependent on the tools provided at the end-effector and is not defined in this study. The purpose of this study is to demonstrate that the AR work cell is applicable for a variety of work cell operations. Unlike the robot arm, the drawings for the CNC machine, conveyor belt and pallet are part files and no assemblies are required. Since these components act as dummy machines without specific operations, the files are drawn to full scale and remain static throughout the study. They are also rendered and coloured using OpenGL.

HUD implementation

The addition of a HUD is extremely useful when virtual content is involved in any context. The use of a HUD not only extends the user's knowledge of the current operation, but also updates itself continuously on the current situation. HUD is purely AR generated and therefore, it is solely dependent on the codes added into the program. The information needs to be portrayed in such a way that it is viewable regardless of display systems. The display system can either be a HMD or a simple PC monitor such as that used in this study. The information overlay includes the current tool equipped to the CNC machine, speed of the conveyor belt (albeit not specific) as well as the current coordinate of the manipulated object in the virtual workspace.

It is relatively simple to print text on the virtual environment using OpenGL. This poses a problem as the global origin is set to the 'Hiro' marker beforehand, which causes the text to appear on the marker and moves accordingly when the marker is moved. Hence, the text that appears on the marker is merely a text, rather than a HUD. For HUD, the text is a 2D overlay located on the top left corner of the screen. The following codes are utilized to achieve the above purpose:

```
glMatrixMode(GL_PROJECTION);
glMatrixMode(GL_MODELVIEW);
```

These codes basically project the information onto a 2D space and the coordinate of the origin appears at the centre of the screen. A semi-transparent background is first drawn before the information is displayed on the screen in order to increase the visibility of the words. The following functions are added in the *init()* function of the program:

```
glEnable(GL_BLEND);
glBlendFunc(GL_SRC_ALPHA, GL_ONE_MINUS_SRC_
ALPHA);
```

These functions enable blending, in which incoming primitive colours are blended with the colours stored in the frame buffer. The relevant information on the screen is then printed out using the *printw* function. Figure 10 shows the real-time HUD display, with the information of the cube's coordinate updated in accordance with changes in position.

Results and discussion

The code is initially tested with a downscaled version of the robot in order to observe the degree of accuracy of the end-effector in following the teach pendant according to the angles calculated for each arm. This is made possible by assembling the joints individually in the AR environment based on the kinematic model. A vertex is drawn at a point by right-clicking the mouse button, and the coordinate and angle of each arm are automatically saved in separate files. Drawing additional vertices will create lines for the linear movement of the robot arm.

Link to Full-Text Articles :

http://pib.sagepub.com/content/229/6/1029