International Journal of Industrial Engineering, 15(3), 314-322, 2008.

VR-Based Robot Programming and Simulation System for an Industrial Robot

Yap Hwa Jen, Zahari Taha, and Lee Jer Vui

Centre for Product Design and Manufacturing (CPDM),
Department of Engineering Design and Manufacture,
Faculty of Engineering, University of Malaya,
50603 Kuala Lumpur, Malaysia

Corresponding author's e-mail: {YH Jen; hjyap737@.um.edu.my}

Traditional robot programming such as teach by lead etc have been used for many years. These methods are considered not efficient and outdated in the current industrial and market demands. In this paper, virtual reality (VR) technology is used to improve human-robot interface - no complicated command or programming knowledge is required. The system is divided into three major parts: task teaching by demonstration in a computer generated virtual environment, a graphical robot simulator with intelligent robot command generator and last but not least, real task execution. The user is requested to complete the desired task in a virtual teaching system by wearing a data glove attached with a sensor tracker. The process path will be simulated and analyzed to obtain the optimum trajectory. Robot motions can be checked through the simulation program and robot program can be generated for the real task execution.

Significance: The designed VR-based system has overcome the safety issues and the lack of trained personnel. Simple and user-friendly interfaces are designed for inexperienced staff to generate robot command without damage the robot or interrupting the production line. They can try many times to obtain the optimum solution.

Keywords: Robot programming, virtual reality (VR), human-robot interface, process path, simulation

(Received 15 November 2007; Accepted in Revised form 19 February, 2008)

1. INTRODUCTION

Virtual reality (VR) is a very powerful human-machine interface that provides the user with visually realistic images and completed with multiple sensorial channels. Virtual reality is created by computer graphics with depth buffer. This technology allows human to visualize, manipulate and interact with a virtual environment. The virtual environment is updated in real-time, according to the programmer inputs. It is also recognized as a very powerful human-machine computer interface, which changes the ways of human interact with machines (Grigore & Philippe, 1994).

Graphical simulation is an evolving technique being used in robotic systems to generate robot command with no risk and cost effective. In general, a simulator is used for teaching robot kinematics and is widely used for teach robot programming. One example is the KUKA robot based simulation software – KUKA SimPro. The simulation package includes a program called KUKA OfficeLite, which generates KUKA robot command (Sett *et al*, 2002). However, the system is manipulated and controlled by mouse and keyboard only. VR tools should be included to allow the user to program in an easier and "natural" way.

An industrial robot is a programmable automation machine used in manufacturing. It can be programmed and perform some task over and over, until reprogrammed to perform some other tasks (Mikell, 1986). Industrial robots are used in factories as positioning machines to replace human being. The robot can be programmed, which robot programming is concerned with teaching and assigning the robot cycle to perform the required tasks. Traditionally, robot programming is either online or offline. The user needs to observe the robot's movement and behaviour when performing a task. To make sure the robot can perform the task using the optimum path, analysis, discussions and trial-and-error testing need to be conducted. This will increase the development time and cost.

In this paper, virtual reality technology is used to improve the human-robot interface. Offline programming is used to generate a robot program to reduce robot down time. OpenGL and C/C++ programming are used to develop the software. The developed system allows the user to teach the robot in a stereoscopic virtual environment without touching the real robot and this is done using special hardware, which are 3D glasses, data glove and sensor tracker.

The proposed VR-based robot programming system consists of a series of objects, which can be viewed from any position and angle by the user. Objects have certain actions assigned to them, which simulate the actual environment. Through this virtual environment, users can select the appropriate sensing modality and programming that interact directly

with the actual case. They can change the position and orientation of the monitor window in the virtual world in real-time using input devices.

The system enables the students or non-experts to apply their programming techniques without interacting with the real robots. Offline programming within the virtual environment also reduces the required skills of a programmer, programming error and programming time. In additions, experiences users can use the system to teach and demonstrate their techniques through this platform.

On the other hand, users can simulate and plan an operation through the robot simulation program. They can run trials as many times as they want with no risk and extra cost. The possibility of damage to the robot is reduced to a minimum level because the robot commands are applied after testing in a virtual environment. It also removes the user from a potential hazardous environment. Programs are generated and tested through virtual teaching. Therefore, production lines are not disturbed during the teaching stage.

2. SYSTEM ARCHITECTURE

In this research, the user is requested to perform a pick-and-place task (palletizing). The user is requested to move the work piece on the conveyer to the storage table in the Virtual Teaching System. The sensor tracker and data glove will track the hand movements and fingers flexure as the process path and loaded into the simulation program. The simulation program is used to check the robots' aspects like clearances, reaches and so on. The Command Generator allows the user to generate the robot command after an optimum path is obtained. The robot command will then be downloaded into the robot system to perform the task. A special program called ModGenSTL is used to convert CAD models (in STL format) into ASCII format, which can be read by both the teaching system and simulation program. The data flow and system architecture of the developed system are shown in figure 1 and figure 2, respectively.

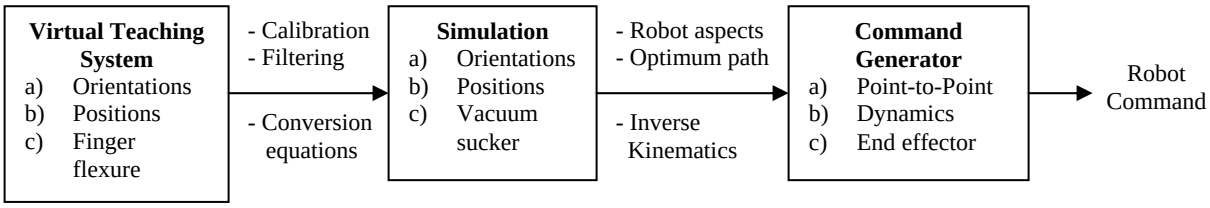

Figure 1: Data flow

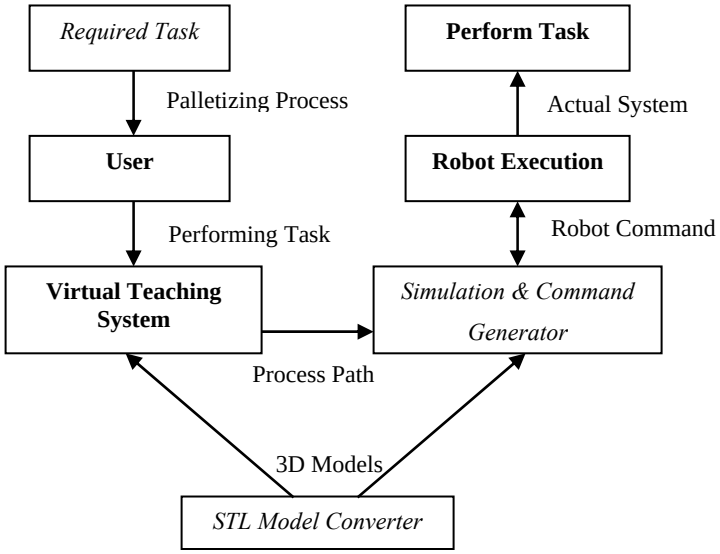

Figure 2: System architecture for the developed system

2.1 Tools

A Virtual Teaching System requires a high-end computer system (memory and graphic card performance) to provide stereoscopic images and update the virtual environment in real-time with minimum simulation latency. On the other hands,

a high-performance system is also required for the simulation program to reduce the simulation period. These can be done on a personal computer with a high processing speed and larger memory capacity. Quad-buffer supported graphic card is used to generate stereoscopic images, in which the refreshing rate is synchronized with a 3D glass.

There are three special VR tools required in the research – data glove, sensor tracker and 3D glasses. The right-handed data glove, attached with a three-dimensional sensor tracker, is used to interact with the virtual objects. Sensor tracker will record the positions and orientations of the user's hand, while the glove tracks the hand gesture (pick and place). On the other hands, the 3D glasses are used to allow the programmer to view stereoscopic images. It is synchronized with the monitor-refreshing rates to provide depth buffer of the virtual objects. In this research, a refreshing rate at the frequency of 100 Hz (screen resolution is 1024 x 768 pixels) is used to reduce the probability of dizziness and cyber sickness.

An industrial robot (Kuka Robot – KLC6) is used to execute the robot command that been generated from the simulator. To date, these industrial robots have achieved a mature level in terms of accuracy and repeatability.

3. VIRTUAL OBJECTS AND MODELS

An interactive and economical design visualization software for a three-dimensional model generator has been developed to support the system. ModGenSTL (Yap *et al*, 2006), is a 3D CAD model generator and visualization software to allow a user to visualize and manipulate stereolithography (STL) geometry with fundamental graphical techniques. In this research, 3D models are created in ProEngineer and exported as Stereolithography (STL) ASCII file (P.F. Jacobs, 1996). odGenSTL is able to import these models and then converted into a format that can be read and loaded by other programs in the system. The new format stored information about 3D objects as number of facets or 3D triangles in a complex digital model. These 3D triangles are used for collision detection algorithms using bounding-boxes methods. Figure 3 shown the different types of the object mode of a gripper loaded by the program.

The program also supports the simple assembly of the model. Transformation, rotation and rendering features can be performed to verify the matching and collisions between two models. Scaling factor can be changed if necessary. It is important to check the above aspects because the unit used for these files many differ (during modelling). To date, the program only allows two models to be loaded into a single documents interface.

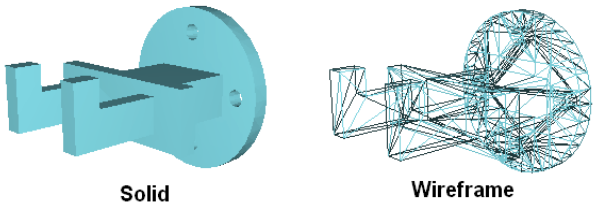

Figure 3: Different types of object mode

4. VIRTUAL TEACHING SYSTEM

The Virtual Teaching System is developed using C/C++ with OpenGL. Open GL is a generic version of the interface made available to a wide variety of outside vendors in the interest of standardization. Open GL support quad buffering to generate stereoscopic images with minimum flickering. It is also a platform independent API (Application Programming Interface) in which the virtual environment can operate either in UNIX or Windows operating system (John Vince 1995).

4.1 Co-location Workstation

The virtual teaching system consists of a series of objects, which can be viewed from any positions and angles. Through real-time updating of stereoscopic images, the programmer will feel himself / herself as a part of this virtual environment. The programmer works "together" with the virtual objects to complete the desired task. Therefore, a physical environment for simulation workstation is needed, to make the users feel, see and interact with objects in the most natural and ergonomic position. This simulation workstation should co-locate between 3-D view, virtual objects and user's hand. The user can see and interact with the virtual objects in the same place, as shown in figure 4. The structure of the platform is tested during the design stage so that it can support the weight of the equipments, e.g. monitor. An adjustable platform is designed to fulfill the variation in human anatomy.

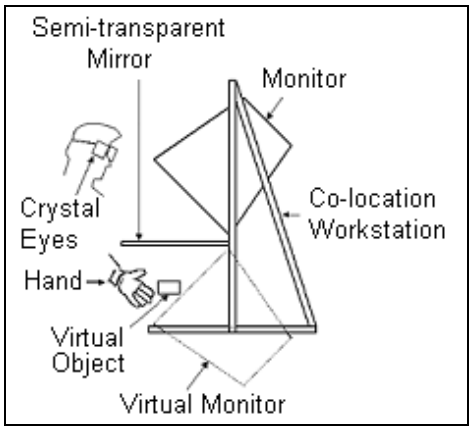

Figure 4: Co-location workstation

4.2 The Virtual Environment

The models are developed using solid modelling software, which mimic the actual set-up in the laboratory. Models are converted using ModGenSTL and re-rendered using OpenGL API. The computer graphics is not only used to define the shape and geometry of the objects, but also to describe the spatial and behaviour of the virtual objects. There are four virtual objects: the hand, work pieces, conveyer and storage table, as shown in figure 5. A database is created to store all the information of the three-dimensional models and the virtual environment. This information includes the physical properties and dynamic behaviour of the models. Data from the input devices are also stored in the database.

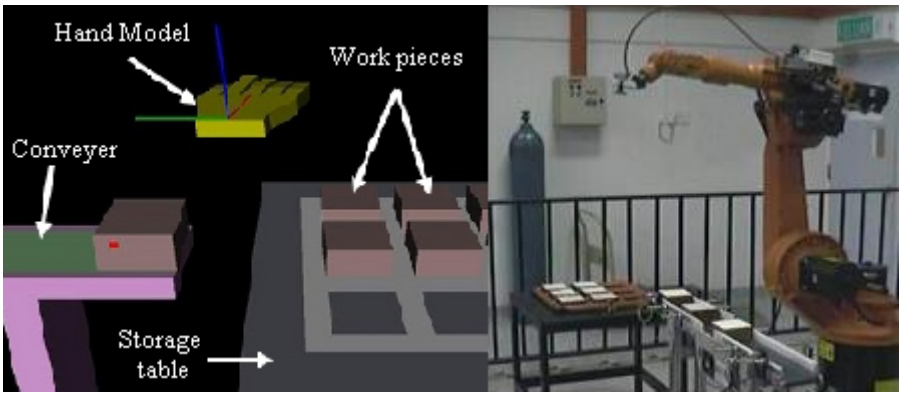

Figure 5: Virtual objects and actual laboratory set-up

4.3 Enhancing performance

The conveyer and storage table in the virtual environment are static. The work pieces are located either on the storage table or conveyer. Only the work pieces can be grasped and removed by the virtual hand. Collision detection technique is used to perform grasping. Grasping is recognized by the bending angle of the fingers and distance between the hand model and work piece (Tomoichi & Hiroyuki, 1992). In the virtual environment, the programmer is not provided with force or tactile feedback. The only mean of confirming the grasping is through visual feedback. Colour changes are used to indicate grasping. The colour change is seen on a small rectangular shape on the work piece. The colour and position of the rectangular shape will change according to the status of the grasping. There are three different colour changes, as shown in figure 6, which represented three situations: (a) No collision is detected, (b) collided and ready to grasp and (c) grasped.

Figure 6: Grasping signals

5. DATA PROCESSING

Both of sensor tracker and data glove communicates with the host computer through serial communication ports (DB9). The interface is used to retrieve the data from equipments and then transferred to the computer system. A dedicated C/C++ class, *CSerial*, is created to communicate and initialize sensor tracker with the host computer. However, hardware initialization and serial port communication of the data glove is done by a driver. The main functions of this class are to open/close and send/read data for communication port

Calibration is needed in order to improve the accuracy of the data and this data should be filtered to reduce the computational load. For the data glove, the raw data (raw_{val}) for each sensor is a 12 bit unsigned value. However, the dynamic range (difference between maximum and minimum output value) may differ when peoples with different hand sizes use the glove. The dynamic range, figure 7, can be obtained by flexing and un-flexing the fingers and calculated using equation (1). On the other hand, the data fluctuates although the sensor is in a static position. The fluctuation data is used to define the cut-off line for the filtering process. Any data within the cut-off line is assumed as zero (static position).

$$DynamicRange = raw_{\max} - raw_{\min} \quad \text{---------- (1)}$$

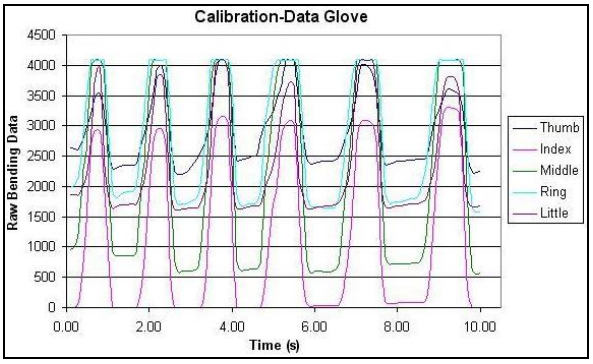

Figure 7: Calibration data for data glove and sensor tracker

Conversion equations are needed to convert the raw data to a readable and useful form. For the data glove, finger flexures are converted into bending angle by assuming that the maximum bending angle for each finger is equal to 90°. Therefore, the value can be calculated using the following equation:

$$FingerBendingAngle = \frac{raw_{val} - raw_{\min}}{DynamicRange} \times 90° \quad \text{---------- (2)}$$

However, data from the sensor is in volts (0V to 5V). A conversion equation is required to convert the voltages value to acceleration values (m^2/s), where the slope and intercept is defined by the calibration data for the sensor, as shown in equation (3). Kinematics' equations are used to convert accelerations into velocities and positions.

$$Acceleration = (Slope \times VoltageValue) + Intercept \quad \text{---------- (3)}$$

$$v = u + at \quad \text{---------- (4)}$$

$$s = ut + \frac{1}{2}at^2 \quad \text{---------- (5)}$$

5.1 Data filtering

Fifty samples are taken per second to reduce latency and flickering problems. The sensor records the positions, orientations, velocity and accelerations of the programmer's hand. Positions and orientations are used for the robot kinematics. However, velocity and accelerations are used to define the dynamic behaviours of the robot. A filtering process is required to reduce the large amount of data. However, the filtering processes are done for the sensor tracker only, which is used to generate the robot parameters. Three methods are used, namely:

a) Stopped positions – Movement of the hand is stopped for a while. The error occurred because the human's hand is not 100% static when the movement is stopped. The hand will slightly shake in space, which cannot be filtered using the cut-off line. These data are too close to each other and the average data can be calculated by using a statistics equation.

$$\bar{x} = \frac{\sum x_i}{N} \text{ ---------- (6)}$$

b) Straight paths – In general, only two points are required to define a line: starting point and ending point. Intermediate points can be ignored and erased from the database. Figure 8 shown the process path differs between actual data from sensor tracker and expected robot movement in a linear line (Point-to-Point).

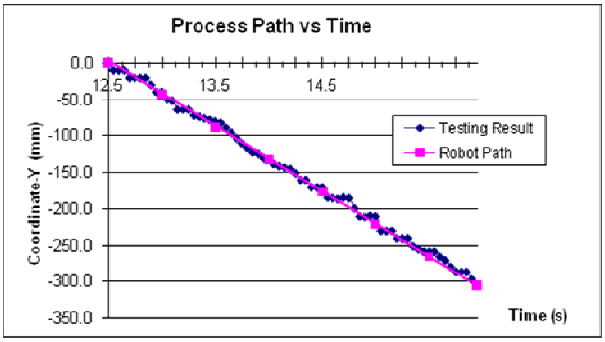

Figure 8: Data filtering for straight path

c) Curving paths – In this particular study, no curve path is used to perform the task. However, a method is developed based on the Bézier curve and this method is supported by OpenGL (Francis, 2000). An average value (control point) is calculated for every 5 to 10 points. Four control points are required to define the curving path in OpenGL.

6. SIMULATION AND COMMAND GENERATOR SYSTEM

The development of robot a simulation system offers potentially great and cost-effective benefits. The time for which a robot is out of production may be reduced by as much as 85% by using offline programming (A.C. Boud & S.J. Stiner, 2000). Before the actual operation, the movement of robot manipulators with any tools is simulated on the PC. The simulation program offers the ability to record and analyse the manipulations during a procedure. This type of analysis enable the users to improve their skills, as well as test them as to whether or not their manipulative skills and paths are at an acceptance level of performance for a given task.

6.1 Robot kinematics

There are two basic ways of describing robot movements for industrial robot in robot control system – in terms of manipulator joint angles or end-effector location. The forward kinematics determined the position and orientation of the robot's end (tool center point), with all joints' variables are known. However, inverse kinematics is used to calculate the value of each joint variable by knowing the position and orientation of the end-effector.

The Denavit-Hartenberg representation (Lee & Chang, 2003) has been used to derive the forward and inverse kinematics equations of all possible configuration of a Kuka robot. The Kuka is a 6-axis manipulator robot. The link transformation that relates the first joint until the sixth joint is summarized and equations (7) specify how to compute the position and orientation of the tool center point refer to the base.

$$ {}^{R}R_{H} = {}_{1}^{0}A\,{}_{2}^{1}A\,{}_{3}^{2}A\,{}_{4}^{3}A\,{}_{5}^{4}A\,{}_{6}^{5}A = {}_{1}^{6}A = \begin{bmatrix} r_{11} & r_{12} & r_{13} & P_x \\ r_{21} & r_{22} & r_{23} & P_y \\ r_{31} & r_{32} & r_{33} & P_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \text{ ---------- (7)}$$

where

$$r_{11} = C_1C_{23}C_4C_5C_6 + S_1S_4C_5C_6 - C_1S_{23}S_5C_6 - C_1C_{23}S_4S_6 + S_1C_4S_6$$

$$r_{21} = S_1C_{23}C_4C_5C_6 - C_1S_4C_5C_6 - S_1S_{23}S_5C_6 - S_1C_{23}S_4S_6 - C_1C_4S_6$$

....

$$P_z = S_{23}C_4S_5d_6 - C_{23}C_5d_6 - S_{23}a_4 - C_{23}d_4 - S_2a_3 + d_2 + d_1$$

Besides that, the Kuka is characterized by a spherical wrist, which the inverse kinematics problem can be separated into positional part *($\theta_1, \theta_2, \theta_3$)* and orientation part *($\theta_4, \theta_5, \theta_6$)*. The details of derivations are not shown in this paper.

$$\theta_1 = A\tan 2(P_{yw} / P_{xw}) \quad \text{, and } \theta_1 \pm 180° \text{ ---------- (8)}$$

$$\theta_2 = A\tan 2\left[\frac{(d_1 + d_2 - P_{zw})(C_3 a_4 - S_3 d_4 + a_3) - (C_1 P_{xw} + S_1 P_{yw} - a_2)(S_3 a_4 + C_3 d_4)}{(d_1 + d_2 - P_{zw})(S_3 a_4 + C_3 d_4) + (C_1 P_{xw} + S_1 P_{yw} - a_2)(C_3 a_4 - S_3 d_4 + a_3)}\right] \text{ ---------- (9)}$$

$$\theta_3 = A\tan 2\left[\frac{d_4}{a_4}\right] + A\tan 2\left[\frac{\pm\sqrt{a_3^2 + d_4^2 - [P_{xw}^2 + P_{yw}^2 + (d_1 + d_2 - P_{zw})^2 - a_2^2 - a_4^2 - a_3^2 - d_4^2]^2}}{P_{xw}^2 + P_{yw}^2 + (d_1 + d_2 - P_{zw})^2 - a_1^2 - a_3^2 - a_4^2 - d_4^2}\right] \text{ ---------- (10)}$$

$$\theta_4 = a\tan 2\left(\frac{-S_1 m_{13} + C_1 m_{23}}{-C_1 C_{23} m_{13} - S_1 C_{23} m_{23} + S_{23} m_{33}}\right) \text{, and } \theta_4 \pm 180^{o} \text{ ---------- (11)}$$

$$\theta_5 = a\tan 2\left(\frac{\pm\sqrt{(-S_1 m_{13} + C_1 m_{23})^2 + (-C_1 C_{23} m_{13} - S_1 C_{23} m_{23} + S_{23} m_{33})^2}}{-C_1 S_{23} m_{13} - S_1 S_{23} m_{23} - C_{23} m_{33}}\right) \text{ ---------- (12)}$$

$$\theta_6 = a\tan 2\left(\frac{C_1 S_{23} m_{12} + S_1 S_{23} m_{22} + C_{23} m_{32}}{-C_1 S_{23} m_{11} - S_1 S_{23} m_{21} - C_{23} m_{31}}\right) \text{, and } \theta_6 \pm 180^{o} \text{ ---------- (13)}$$

6.2 The virtual robot

A robot simulation system has been developed based on C/C++ programming language with OpenGL. After filtering, the new set of data is sent to the simulator. The simulator has the capacity of real-time simulation of a virtual Kuka robot through 3D animation on a personal computer. The user can control and visualize the virtual robot by mouse or keyboard through a user-friendly interface window. The movement of the virtual robot can be simulated many times after the filtered data has been loaded. The user can change the viewing angle or zoom in from any position. The simulator is designed to:

- Compare the filtered data with the original data.
- Check the dynamic behaviour. Since the teaching system is based on the human hand, robot limitations need to be checked and confirmed.
- Check the robot aspects like orientations, reaches, clearances, collision detections and so on.

6.3 Command generator

After the user is satisfied with the process, the database can be translated into a Kuka Robot Language (KRL) through a command generator or postprocessor. The output commands generated by the developed post-processor is only readable by the controller of the Kuka Robot model KRC6. Robot languages differ from one another. The creation of a generic post-processor is not practical. The movements of the robot are executed with constant velocity and acceleration (default). Figure 9 shows an example of KRL commands that been generated. These KRL commands can be loaded into the controller of the real Kuka robot.

```
DEF PTP_AXIS()
$VEL_AXIS[1] = 100
$VEL_AXIS[2] = 100
$VEL_AXIS[3] = 100
$VEL_AXIS[4] = 100
$VEL_AXIS[5] = 100
$VEL_AXIS[6] = 100

$ACC_AXIS[1] = 100
$ ACC_AXIS[2] = 100
$ ACC_AXIS[3] = 100
$ ACC_AXIS[4] = 100
$ ACC_AXIS[5] = 100
$ ACC _AXIS[6] = 100

PTP {AXIS:A1 26.5651, A2 3.46629,A3 34.4992, A4 220.355, A5
43.6572, A6 149.288}
PTP {AXIS:A1 -42.605, A2 -10.3901,A3 19.0239, A4 -62.5439, A5
3.33649, A6 45.7593}
PTP {AXIS:A1 37.8414, A2 -8.14868,A3 -2.29048, A4 57.5508, A5 -
7.09465, A6 -42.9173}
END
```

Figure 9: An example of KRL command

7. DISCUSSIONS AND CONCLUSION

The developed system allowed the user to teach the robot through the virtual environment. The process path is automatically tracked by the sensor tracker and saved into a database. The software allows the user to try as many times as required, until an optimum path is obtained. The system has been tested for the palletizing process (pick-and-place).

7.1 Case study – Palletizing Process

Experiment on the palletizing task was performed, which the robot is requested to move the work-pieces from a fixed location (conveyer) and rearrange into the nine different locations (storage table), as shown in figure 5. In the VR-based system, experiments were done using 0.01 second as the interval time, which means the program is able to generate 100 data per second. The setting is chosen to generate flicker free real-time simulation. However, actual speed of the robot can be adjusted according to the conveyer system and production line.

Based on the result from virtual reality system, the KUKA robot can be programmed by using Point-To-Point (PTP) method in linear motion. The complete path for a single work-piece is illustrated in figure 10, and the loop can be repeated for another eight variable locations. The robot command is then generated and loaded into robot for actual testing. The robot is able completed the task as expected. The system also been tried by students who do not have any knowledge about KUKA robot programming skill, and the same result is obtained.

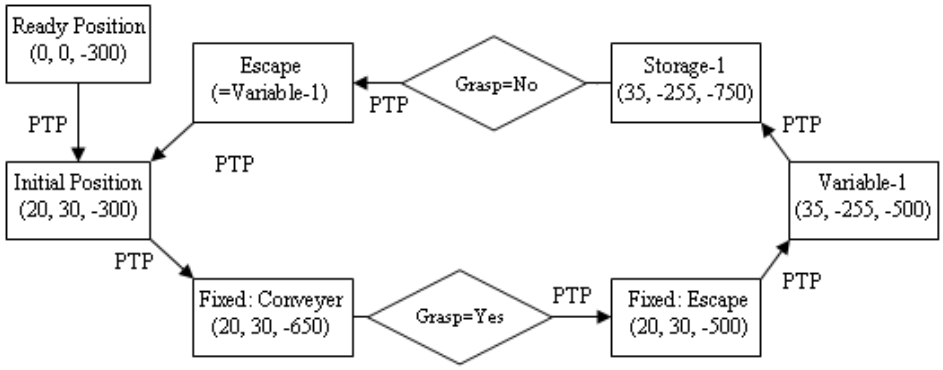

Figure 10: Complete path for the first work-piece.

7.2 Conclusion

Advanced computer technology and modern electronic developments (virtual reality technology) have made it possible to develop an offline programming system with real-time simulation. It demonstrates the application of virtual reality as a powerful tool for human-robot interaction. The developed VR-based system can be used to generate a Kuka KLC6 robot command though three sub-system – virtual teaching system, robot simulator and command generator. The system has been tested with palletising process. The robot command also has been loaded into the robot to perform the task.

It is expected that the system developed in this research can be directly applied to an industrial robot. Programmer can use the virtual teaching system and simulation to confirm the robot program. Mistake in programming can be avoided and damage to the robot can be reduced. It also can be used to improve the work envelope, efficiency and safety. The developed system also can be used to test a variety of robots. However, the robot aspects (model, workspace, kinematics, robot languages and so on) need to be updated accordingly. This may improve the interaction and integration between human and machine (robots). A lot of experience can be gained from the robot simulations without having the actual robots at no extra costs and risks.

8. REFERENCES

1. A.C. Boud and S.J. Stiner (2000). A New Method for Off-Line Robot Programming: Application and Limitations Using a Virtual Environment. Proceedings of the 5th International Conference on Factory 2000, pp. 450-455.
2. Francis S. Hill, Jr, (2000). Computer Graphics using OpenGL, 2nd ed., Prentice-Hall Inc.
3. Grigore Burdea and Philippe Coiffet (1994). Virtual Reality Technology. John Wiley & Sons, Inc., New York.
4. John Vince (1995). Virtual Reality System. Addison-Wesley Publishing Company, Singapore.
5. Lee H.S. and Chang S.L. (2003). Development of a CAD/CAE/CAM System for a Robot Manipulator. Journal of Materials Processing Technology, vol 140, Issues 1-3, pp. 100-104.
6. Mikell P. Groover (1986). Industrial Robotics: Technology, Programming and Application. McGraw-Hill, New York.

7. P.F. Jacobs (1996). Stereolithography and Other RP&M Technologies. Society of Manufacturing Engineers.
8. Sett, A. & Vollmann K. (2002). Computer Based Robot Training in a Virtual Environment. IEEE International Conference on Industrial Technology 2002 (ICIT '02), vol.2, pp. 1185-1189,
9. Tomoichi Takahashi and Hiroyuki Ogata (1992). Robotic Assembly Operation based on Task-Level Teaching in Virtual Reality. Proceedings of the IEEE International Conference on Robotics and Automation, pp. 1083-1088.
10. Yap Hwa Jen, Z. Taha, Irvin Hoh and Lee J.V. (2006). Development of a 3-Dimensional CAD Model Generator. Proceedings of the 1st International Conference on Manufacturing & Material Processing (ICMM 2006), University of Malaya, pp. 268-271, Kuala Lumpur.

BIOGRAPHICAL SKETCH

Yap Hwa Jen is a researcher cum PhD student in the Centre of Product Design and Manufacture (CPDM), University of Malaya, Malaysia. He is also a Lecturer in the Department of Engineering Design and Manufacture, Faculty of Engineering, University of Malaya, Malaysia. He obtained his bachelor degree in Mechanical Engineering with Honors from the University of Malaya in 2000. He received a Master of Engineering Science from University of Malaya in 2005. His research interests included virtual reality, human-computer interface, product design, robotics and automation.

Zahari Taha is currently director and Professor of Centre for Product Design and Manufacturing (CPDM) Faculty of Engineering University of Malaya (UM). He graduated with a BSc in Aeronautical Engineering with Honors from the University of Bath, UK. He obtained his PhD in Dynamics and Control of Robots from the University of Wales Institute of Science and Technology in 1987. From 1995 to 1998, he completed his postdoctoral at the University of Wales Institute College Cardiff in the area of engineering design and ergonomics. He was awarded the Hitachi Fellowship in 1992 and also the Young Malaysian Scientist award by Ministry of Science, Technology and Environment in 1997. Dr. Zahari has published more than 100 papers in academic books, journals and proceedings. His major research interest includes mobile robots, underwater robots, surgical robots, ergonomics design, ergonomics at work, software development for traffic applications and motion analysis.

Lee Jer Vui has obtained his BSc (Applied Physics) from University of Malaya in 1999. In addition, he has obtained his MSc (Laser and Optoelectronics) from the same university in 2001. Currently, he is pursuing his PhD in Robotics at Department of Engineering Design and Manufacture, Faculty of Engineering, University of Malaya. His research interests are robotics, computer simulation and visualization, and Laser Material Processing.