Integration of vision and robot motion: a novel approach to teaching kinematics transformation using an industrial robot arm

S. Mekid (corresponding author),[a,b] A. W. Labib,[c] M. F. Rajemi[a] and H. Frost[a]
[a] *School of Mechanical Aerospace and Civil Engineering, University of Manchester, Manchester M60 1QD, UK*
E-mail: smekid@kfupm.edu.sa
[b] *King Fahd University of Petroleum and Minerals, Mechanical Engineering Department Dhahran, 31261, KSA*
[c] *University of Portsmouth, Portsmouth Business School, Portsmouth PO1 3DE, UK*

Abstract The paper reports on work carried out on an educational exercise for graduate and undergraduate students that aids the study of Denavit-Hartenberg (DH) programming, through a real application within a 'learning by doing' scheme. The work presented to students requires the integration of the motion of an industrial robot (ASEA IRB 6) with a camera vision system (Baxall CD9752); they are asked to develop software that will support: the motion of the robot arm, including forward and reverse kinematics; the detection of an object on a test table; the picking up of the object; and its move to a target position on another test table. To assess the experiment, students select seven random points in the active camera area to determine the difference between the actual location of the end-effector and the location calculated using two methods: the trigonometric solutions and the DH parameters. Students discover that the trigonometric technique gives better position accuracy than DH. The project forms an excellent practical exercise to enhance student learning.

Keywords forward and reverse kinematics; vision; robot motion

Introduction

Conventional lectures are not always an efficient way to convey complex information and secure good learning for students. Robotics, as a field, when taught to undergraduate and master's (MSc) students, has two sets of course contents: one set includes robot applications, robot configurations, and actuators; the other includes controls and sensors, as well as an introduction to some programming. Students tend to find the theoretical and mathematically intensive kinematics and control aspects difficult. The exercise described here allows students to try Denavit-Hartenberg (DH) programming and to compare it with a trigonometric method. It concerns robotic vision. The governing software is programmed in Matlab prior to the lab exercise and made ready for use by students. The students are introduced to this work by showing them the importance of robot vision in industry and the obvious support it can provide to humans in various applications. The classes are relatively small – around 25 – and comprise UK students, European Union exchange students and a few students from outside the European Union. The exercise exploits a widely used industrial robot. Although not very new, it is performing well after simple retrofitting of a new micro-controller [1].

Robots are used to maximize both productivity and product quality. The employment of industrial robots equipped with vision has provided industry with the

flexibility to operate robots from a distance without worrying about hazards that may arise where close operation is required. Visual capability can be of benefit in controlling or operating the robot. It has great advantage in hazardous industries such as refineries and nuclear power plants. In addition, the use of an industrial robot with vision helps to eliminate the danger faced by operators in the production line.

One of the objectives for the students in this exercise was to integrate camera vision with an ASEA IRB-6 industrial robot. A good understanding of kinematic transformations and precision of motion was required. The robot had to move to a location defined in the camera image displayed by the supervising computer. In order to achieve this objective, a camera was mounted vertically above a test table at a certain height. The software used to control the robot and to display the image from the camera was written using Visual Basic 6.0. In order to integrate the camera image and the motion of the robot arm, the camera was calibrated to find the relationship between the coordinates in pixels of the camera and the coordinates of the end-effector of the robot arm with respect to the axis origin of the robot arm. The test table was fixed at a certain location so that it did not affect the integration of the camera image and the robot coordinates.

The second objective was to automatically pick up an object displayed in the camera image and relocate it at a specified location on another test table. To pick up objects, robot arms are usually equipped with a gripper, but for this exercise the gripper was substituted for an electromagnet that could pick up a metal object weighing up to approximately 100 g. A suitable command line was required in the program to control the operation of the electromagnet. The end-effector of the robot arm had to approach the object accurately in order to pick it up.

Finally, the overall objective was to construct an educational tool to serve robotics courses in the School of Mechanical, Aerospace and Civil Engineering.

Experimental setup

Equipment

The equipment used in implementing the integration of robot motion and vision for the ASEA IRB 6 industrial robot consisted of the ASEA IRB 6 industrial robot itself, a magnetic coil, an indicator (an ammeter), an ASEA control unit, a supervising computer and a camera (Fig. 1).

The ASEA IRB 6 industrial robot is a five-axis robot. It can hold up to 6 kg of load [2]. The motion of the robot arm was controlled using the ASEA control unit. At the end of the robot arm, an electromagnetic coil (Fig. 2) was fitted to pick up metal objects; in this experiment the object was a black washer (Fig. 3) on the test table.

An ammeter was used to show whether the electromagnetic coil was ‘on’ or ‘off’ (if it was ‘on’, the ammeter needle gave a reading).

The supervising computer was equipped with Visual Basic 6.0 software as the programming language to govern the motion of the robot arm. A Baxall CD9752 camera was mounted on a beam above the test table. The image from the camera could be viewed using the Visual Basic interface (Fig. 4) on the supervising computer.

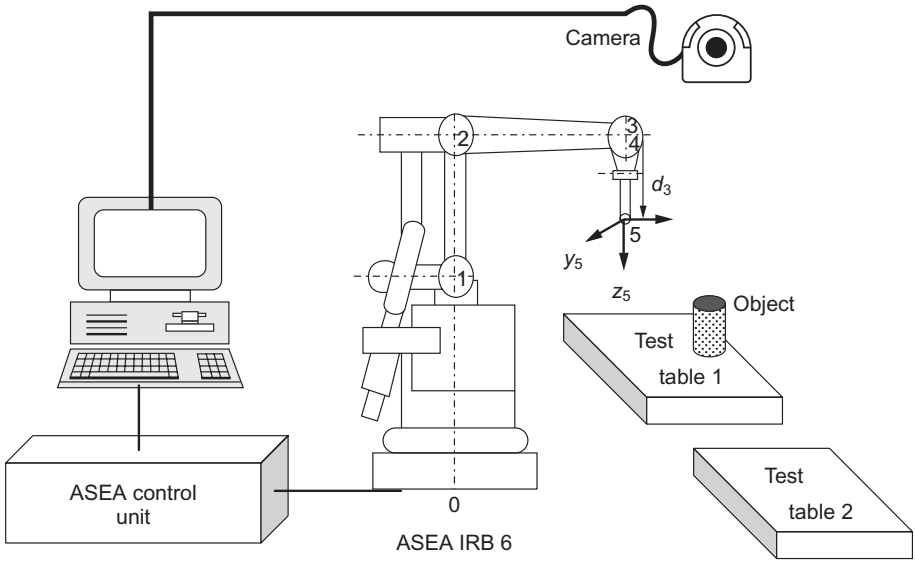

Fig. 1 *The experimental setup.*

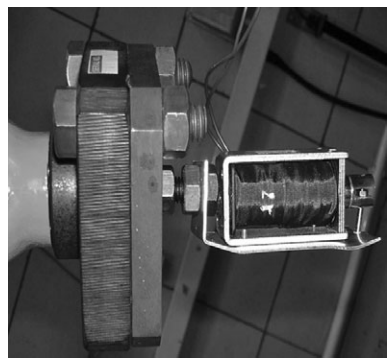

Fig. 2 *The magnetic coil.*

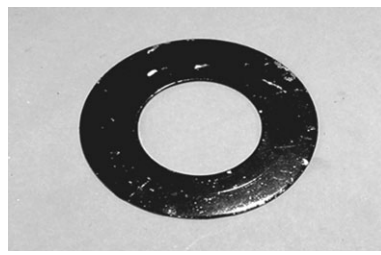

Fig. 3 *The black washer that served as the object the robot had to pick up and move.*

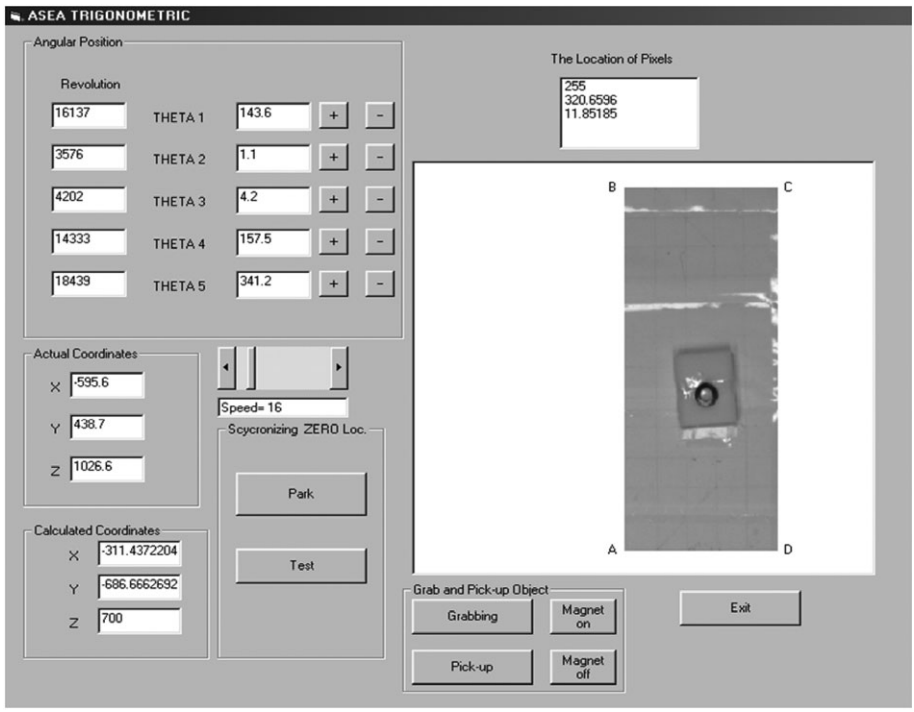

Fig. 4 *The Visual Basic user interface.*

Calibration of the camera

In order to reach and manipulate an object in its workspace, the robot controller had to 'know' in advance the object's position and orientation relative to the end-effector [3]. The relationship between the position of the object, camera and the robot arm had to be developed to ensure that the robot arm moves to the correct position.

The camera was positioned directly above the test table. The aperture, focus and zoom were adjusted until the object image was sharply produced in the visual interface. Lines were drawn on a sheet of white paper placed on the test table [4]. When the program ran, the mouse pointer was positioned on a certain point on the lines drawn on this grid to find the relationship between the camera pixels and the actual coordinates of the point using Cartesian coordinates (x, y and z). Using this point, four constants, C_i (see below), were defined and stored in order to relate the camera pixels to the Cartesian coordinates.

In this experiment, the reference axes for the robot arm were the x axis and y axis. To find the position of the pointer with respect to the robot arm, it was necessary to create a pair of imaginary axes (x_2 and y_2) parallel to the direction of the camera pixels (p_i and p_j) (Fig. 5). A reference point on the object to be moved could be defined using any edge-detection software [5, 6]. This point served as a target to be reached by the end-effector with its magnetic coil.

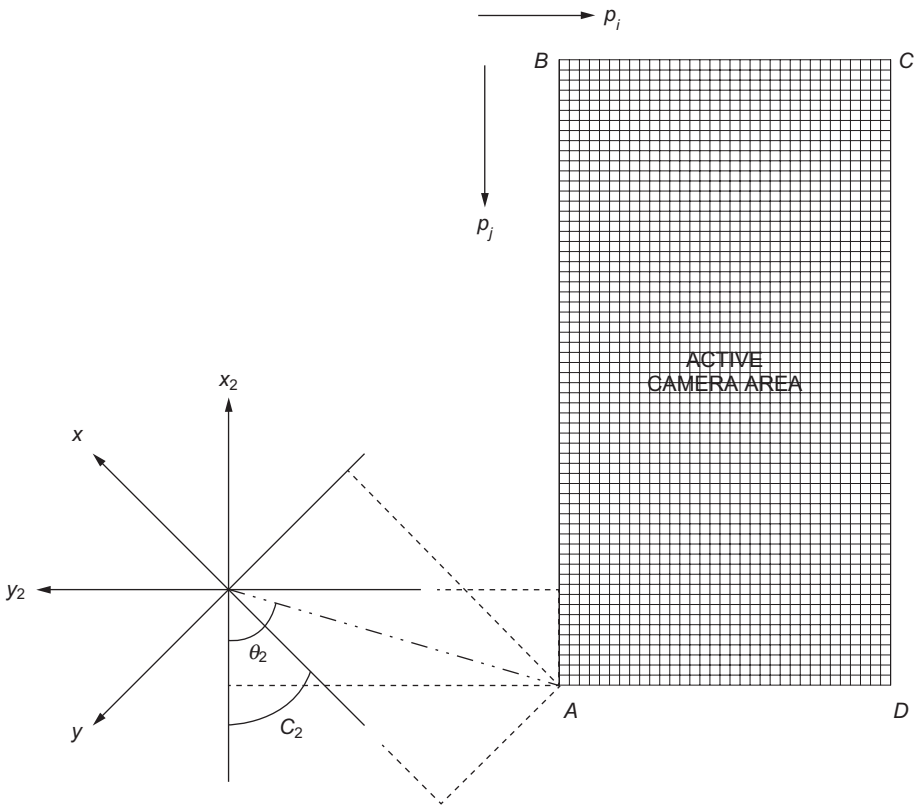

Fig. 5 *The robot arm's reference axes.*

The relationship between the robot reference axes (x and y) and a specific point on the image from the camera is given by:

$$x = x_2 \times \frac{\cos(\theta_2 - C_2)}{\cos\theta_2} \tag{1}$$

and

$$y = y_2 \times \frac{\sin(\theta_2 - C_2)}{\sin\theta_2} \tag{2}$$

where
θ_2 is the angle of the specified point from the x_2 axis,

$$x_2 = C_1 p_j + C_3$$

$$y_2 = C_1 p_i + C_4$$

C_1, C_2, C_3, C_4 are constants.

The kinematics of the ASEA IRB 6

The inverse kinematics for the ASEA IRB 6 can be solved using two different methods. The first involves trigonometric solutions. This method is the easier, since one can easily visualize the movement of each of the axes. The second method is by solving the kinematics using DH parameters [7]. In this method a homogenous transformation matrix is formed using the DH parameters to solve the rotation of each of the axes to achieve a specified point on the image from the camera.

The inverse kinematics using a trigonometric technique

By referring to a specified point on the camera, the movement of each of the axes for the robot arm to reach the point can be solved trigonometrically. Using this method, the axes of the ASEA IRB 6 can be visualized as shown in Fig. 6.

Referring to Fig. 6, we can develop the relationship between the x and y axes with θ_1, θ_2, θ_3 and θ_4, as below:

$$x = r\cos\theta_1 \tag{3}$$

$$y = r\sin\theta_1 \tag{4}$$

where $r = a_2\sin(-\theta_2) + a_3\cos\theta_3 + a_4\sin\theta_4$ (projection on to the x, y plane).

$$z = a_1 + a_2\cos(-\theta_2) + a_3\sin\theta_3 + a_4\cos\theta_4 \tag{5}$$

Note that the value of θ_2 is negative since the direction of rotation of the second axis of the robot arm is counter-clockwise.

Once the location in the image is specified (i.e. the coordinates of x, y and z are given), the movement of each of the axes can be determined by solving the inverse

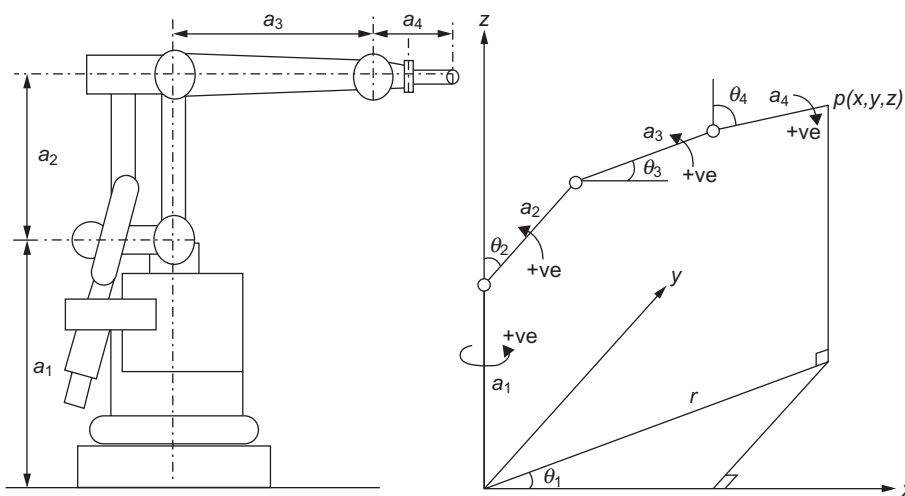

Fig. 6 *Trigonometric solution for the axes of the ASEA IRB 6.*

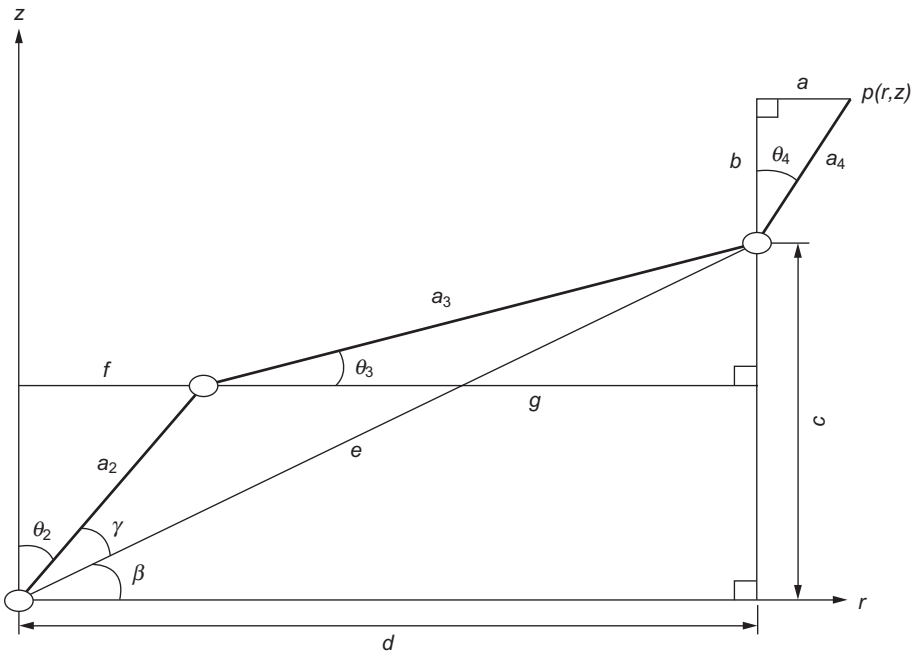

Fig. 7 *Simplified diagram for rotation of the robot arm.*

kinematics of the ASEA IRB 6. For the inverse kinematics we can simplify the axes of the robot arm from Fig. 6 to that shown in Fig. 7 by excluding the first axis, as the first axis is always vertical and moves only in the θ_1 direction.

From equations 3 and 4 we can derive formulae for θ_1 and r:

$$\theta_1 = \tan^{-1}\left(\frac{y}{x}\right) \tag{6}$$

$$r = \sqrt{x^2 + y^2} \tag{7}$$

For a given θ_4, the values of θ_2 and θ_3 can be found (as shown in Fig. 7) by solving equations 8 and 9:

$$a = a_4 \sin\theta_4; \quad b = a_4 \cos\theta_4$$

$$c = z - b$$

$$d = r - a$$

$$e = \sqrt{c^2 + d^2}$$

$$\beta = \cos^{-1}\left(\frac{d}{e}\right)$$

$$a_3^2 = a_2^2 + e^2 - 2a_2 e\cos\gamma \quad \text{hence,} \quad \gamma = \cos^{-1}\left(\frac{a_2^2 + e^2 - a_3^2}{2a_2 e}\right)$$

$$\theta_2 = 90 - \beta - \gamma \tag{8}$$

$$f = a_2 \sin\theta_2$$

$$g = d - f$$

$$\theta_3 = \cos^{-1}\left(\frac{g}{a_3}\right) \tag{9}$$

Programming and implementation

The forward and reverse kinematics methods described above were programmed using Visual Basic 6 as a platform. A user-friendly user interface (see Fig. 4) was developed to implement each of these methods. The operation of the robot arm was programmed to follow the sequence shown in Fig. 8.

The program started by finding the relationship between the pixel coordinates in the camera with the location of the object on the camera image; this process is called 'synchronization'. This relationship is as explained above. The synchronization of the camera needed to be done only after the program starts. Therefore it is important that the camera was fixed during the operation of the robot arm.

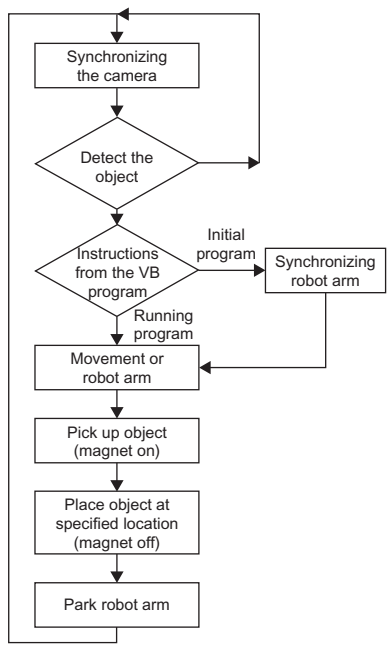

Fig. 8 *The main operations.*

After the camera was synchronized, it produced an image on the user interface program. The camera detected the object by differentiating a greyscale. For example, when the object was black, the greyscale was lower than a brighter background. In this paper the object used was a black metal washer (see Fig. 3). When the camera scanned the object on the test table, it recorded the first location of the change in the greyscale from white to black. The location of this interchange was recorded in the program in terms of pixels. By using the relationship discussed above, the camera pixel coordinates were mapped onto a coordinate with reference to the axis origin of the robot arm. Initially, when the program started, it instructed the user to synchronize the robot arm to calculate the position of the end-effector (i.e. the electromagnet) with respect to the axis origin of the robot axis. The robot arm could move even if it had not been synchronized, but the location of the endpoint would be unknown.

In this exercise, the movement of the robot arm was programmed for two types of motion: the forward kinematics motion, where the angles of rotation of the revolute joints were given to define the end-position; and the inverse kinematics motion, whereby we defined the position of the required point and the robot calculated the corresponding angles of rotation to reach that point. When the end-point/desired position was reached, the 'Pick-up' button in the VB interface was activated. The magnet automatically turned on. The robot arm moved slowly towards the object and picked it up using the magnetic coil. The robot arm then moved to a specified location at which to place (drop) the object. When it reached the desired location, the magnet turned off and the object was released. As soon as the object had been released, the robot arm moved to its parking position, to await further instructions.

In order to ensure that the robot arm moved to the correct position specified in the image of the camera, seven random points were selected, A–G (as shown in Table 1). To assess the results of the implementation, the actual coordinates of the end-effector were compared with the coordinates calculated using each of the two methods described above.

Results and discussion

The challenge for the students was to validate the program with the forward and reverse kinematics. They selected seven random points in the active camera image. Graphs were drawn to compare the calculated coordinates of the end-effector and the actual coordinates of the end-effector of the robot arm.

TABLE 1 *The results of the trigonometric solution (percentage positional accuracy)*

	Point A	Point B	Point C	Point D	Point E	Point F	Point G
Accuracy in x direction	3.7	2.9	3.0	3.7	2.5	2.9	3.4
Accuracy in y direction	3.7	2.9	3.1	2.5	3.1	6.4	2.8
Accuracy in z direction	3.5	3.8	3.7	3.9	3.7	3.7	3.8

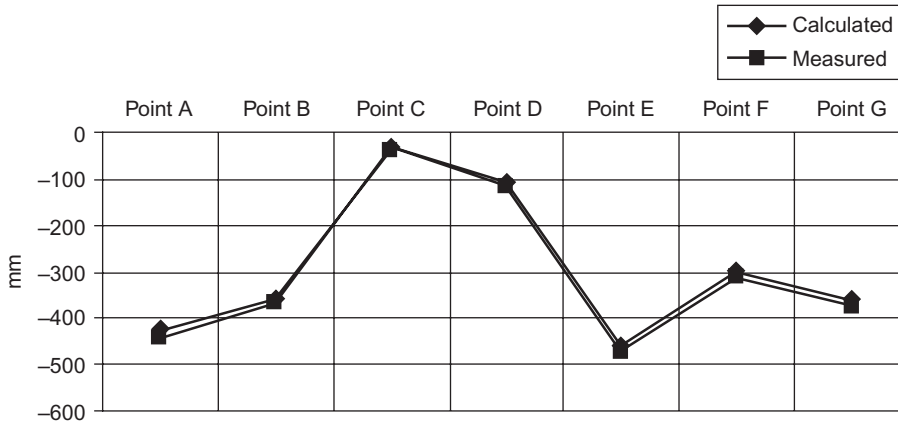

Fig. 9 *The difference between trigonometrically calculated and actual value of the x coordinates of points A–G.*

TABLE 2 *The results of the DH parameters solution (percentage positional accuracy)*

	Point A	Point B	Point C	Point D	Point E	Point F	Point G
Accuracy in *x* direction	15.6	21.6	15.9	11.6	21.8	8.9	19.5
Accuracy in *y* direction	15.6	21.5	15.9	11.3	21.8	8.9	19.5
Accuracy in *z* direction	6.8	6.2	6.3	4.5	6.0	4.0	6.6

The positional accuracy (as a percentage) at the seven random coordinates is shown in Table 1, and the actual differences between the calculated and actual coordinates are shown in Fig. 9.

The positional accuracy (as a percentage) of the seven random coordinates is shown in Table 2, and the actual positioning errors are shown in Fig. 10.

In this application, the trigonometric method seems to be very accurate compared to DH, although positional accuracy was not a principal objective. It is known that DH has a fundamental problem for any attempt to move about the *y* axis, and this may have had increased the errors shown in Table 2 for these relatively long-range displacements.

At the moment there is no algorithm for closed-form solutions in inverse kinematics for an arbitrary arm geometry. Numerical solutions could be obtained at computational expense, but this makes them unsuitable for real-time use [8, 9].

The component equations with the trigonometric method are combined with the exclusive use of the arctangent function of two arguments in order to avoid the problem of angle quadrant ambiguity inherent in trigonometry. This method works well for most manipulators [10].

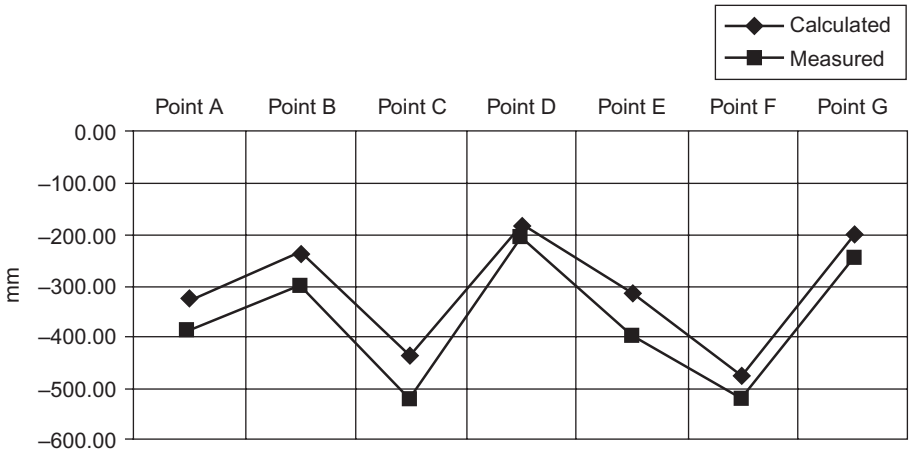

Fig. 10 *The difference between calculated using the DH parameters and actual value of the x coordinates of points A–G.*

Student sessions

Two groups of four students were scheduled in a session lab of three hours, one group to work on this topic, and another to work on programming analysis of a FANUC robot arm. The implementation of the lab exercise started with instructions on how to proceed given to students in the form of a lab sheet. A lab demonstrator supervised the work of the students. Preparation started prior to the session. Students had to write corresponding DH, inverse DH and trigonometric transformations for the robot arm. After the programming, students had to do several tests and compare the transformations. The tests carried out by students increased their motivation to learn from the hands-on experience and to monitor their progress to achieve good results during the lab session. Results and notes were taken regularly within the session. A lab exercise report had to be submitted after one week.

This new educational tool has been used over several semesters. Observations of students behaviour over semesters have led us to adjust some of the tasks, not only to fit within the time frame allocated to students but also to let students have more time to 'play' with the robot arm.

Student feedback

Both undergraduate and master's students have considered the exercise an excellent support for their learning. It gave them a better understanding of DH transformations, which are usually difficult to absorb in a lecture, as well as of the feasibility of implementation of a procedure with a real robot arm, with all the technical issues that they may encounter in the real world, such as embedding a camera for vision, calibration, some programming within an existing framework (e.g. Matlab), and comparison of two sets of transformations. They have appreciated the decision

making and the hands-on experience with the robot arm. Occasionally, some students were involved in repairing faults in the robot movement or supporting devices.

Conclusion

This paper has reported on a learning-by-doing approach as experienced by undergraduate and master's students in a robotics laboratory. In this exercise, locating and handling a specific object using a robot arm were achieved through the use of vision techniques and the implementation of forward and reverse kinematics using both the trigonometric and the Denavit-Hartenberg parameter method to move the object. The preparation of the transformation equations and the programming has greatly improved the ease of understanding by 'playing' with the arm and driving it to specific positions.

Both graduate and undergraduate students who worked on the project have found it extremely informative, as the practical, hands-on experience with an industrial robot supports the theoretical lectures.

Other sessions could include accurate calibration of the camera for greater precision in positioning. Real-time image processing could be added, based on real-time data acquisition with dedicated hardware (e.g. NI, LabView). This usually attracts students' interest.

References

[1] P. Brunn and A. W. Labib, '"New lamps for old!" A method to rejuvenate old robots through the use of a simple set of micro-controller based control boards to replace outdated control systems', *Int. J. Mech. Enging. Educ.*, **33**(4) (2005), 339–348.
[2] ASEA, *Programming and Operating ASEA's Industrial Robots* (ASEA, 1975).
[3] J. A. Akec, S. J. Steiner and F. Stenger, 'An experimental visual feedback control system for tracking applications using a robotic manipulator', *IEEE*, **2** (1998), 1125–1130.
[4] H. K. Fung, T. P. Lung and T. W. Lee, 'Planar vision system for object recognition and robot control', *Engineering Applications of Artificial Intelligence*, **1**(1) (1988), 41–46.
[5] O. Knudsen, *Industrial Vision*, PhD thesis, Technical, University of Denmark (1997).
[6] P. Liu, H. Yan and Z. Zheng, 'Survey of algorithms for edge detection with subpixel accuracy', in *Proc. 3rd International Symposium on Test and Measurement, ISTM-99*, 2–4 June 1999, Xi'an, China, pp. 621–625.
[7] P. J. McKerrow, *Introduction to Robotics* (Addison-Wesley, New York, 1991).
[8] R. Muszynski, 'A solution to the singular inverse kinematics problem for a manipulation robot mounted on a track control', *Engineering Practice*, **10** (2002), 35–43.
[9] W. E. Snyder, *Industrial Robots: Computer Interfacing and Control* (Prentice-Hall, Upper Saddle River, NJ, 1985).
[10] R. P. Paul, *Robot Manipulators* (MIT Press, Cambridge, MA, 1981).