Modeling and Simulation of a 5 DOF Robot Manipulator for Assembly Operations Using Artificial Vision

Luis Felipe Mejía Rodríguez[1], Jesús Carlos Pedraza Ortega[2], Joaquín Santoyo Rodríguez[1]

[1]Universidad Tecnológica de Querétaro, Av. Pie de la Cuesta S/N, Col. Lomas de San Pedrito Peñuelas, Querétaro, Qro., C.P. 76148, México
lmejia@uteq.edu,mx, jsantoyo@uteq.edu.mx
[2]Centro de Ingeniería y Desarrollo Industrial, Av. Pie de la Cuesta # 702, Col. Desarrollo San Pablo, Querétaro, Qro., C.P. 76130, México
jpedraza@cidesi.mx

Abstract. The purpose of this paper is to show the applied methodology for modeling, simulation and automatic program generation for basic assembly operations using a 5 degree of freedom robot manipulator and a vision system. The robot used was a Pegasus II from Amatrol, and the vision system consisted of a web cam and a personal computer. The forward and inverse kinematics of the manipulator is developed, and the results are used for simulation purposes. The main goal is to manipulate small rods and put them into holes drilled in a mounting board, with a program for the robot generated using visual information. The software tools used are Visual C++ 6.0, the OpenGL graphics library and Matlab 7.

1 Introduction

Flexible manufacturing (FM) must have a fast response to changes in production requirements. In order to implement a FM cell, it is common to use robots, automatic conveyors and storage, and CNC machinery. Despite this, it is still a common practice to program the computers (including PLCs and robot controllers) "by hand", delaying the adjustment of the cell. One way for the solution of this problem is the automatic generation of programs for the computers, considering the physical characteristics of the operations required, like size, form, position, etc.

Computing power and costs enables us to process vision data for different tasks of increasing complexity. One of such tasks is the automatic generation of programs for robot controllers, using vision information to obtain the points that must be followed by a robot, and thus allowing us to do the trajectory planning.

Robot modeling requires a deep understanding of the structure and dimensions of the robot. Knowing the dimensions and movement restrictions of the robot, it is possible to apply the Denavit-Hartenberg method to do the direct kinematics, and using some positions constraints, based on the task required for the robot, the inverse kinematics can be calculated.

A. Gelbukh, S. Suárez, H. Calvo (Eds.)
Advances in Computer Science and Engineering
Research in Computing Science 29, 2007, pp. 15-24

Received 10/09/07
Accepted 19/10/07
Final version 24/10/07

This work shows the development of the direct and inverse kinematics for an educational robot of industrial quality, the Pegasus II from Amatrol, and its application for simulation purposes. This work also shows an alternative for the application of artificial vision and image processing in the extraction of positions where assembly operations are required, and the generation of the associated program.

The software tools picked were Visual C++, OpenGL libraries and Matlab. C++ is a general-purpose object-oriented language, very powerful and common in software development. Matlab is a powerful mathematical environment, with toolboxes specific for some kinds of applications, such as image acquisition and processing, robotics, fuzzy control and neural networks. With the right mix of C++ and Matlab, the developer can optimize the life cycle of software development, and with the use of OpenGL a virtual representation can be obtained.

Some related works are [1], [2] and [5].

2 Development

In order to reach the goals of this work, several activities are necessary. This activities are detailed in the next sections

2.1 System Design and Operational Environment

The general structure of the system is shown in the following figure.

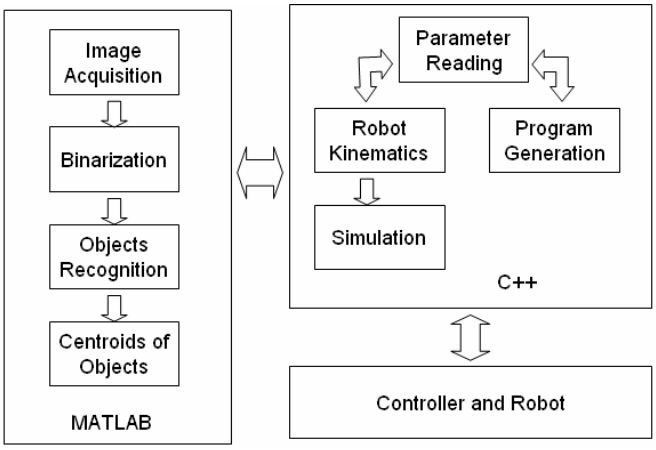

Fig. 1. System design and operational environment.

The tasks that involve images are carried out in Matlab. The outcome of this stage is an array of coordinates of the centers of the holes in a mounting board, stored in a text file. The tasks of simulation and robot program generation are carried out in C++. The outcome of this stage is a virtual robot doing the assembly and a program for the robot controller that can do the actual assembly.

2.2 Specifications and Characteristics of the Pegasus II from Amatrol

The robot manipulator used was a Pegasus II from Amatrol. This is an educational robot, but it has many common characteristics with an industrial one.

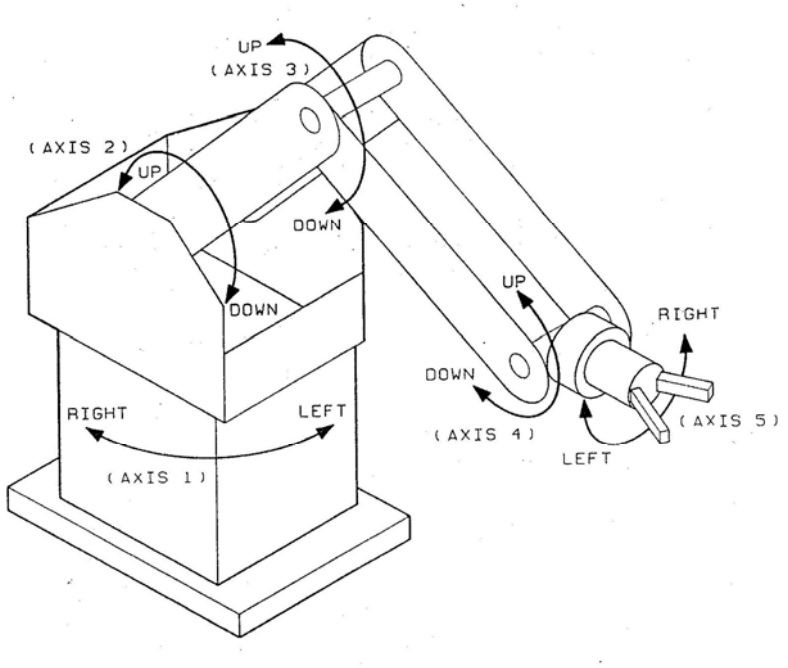

Fig. 2. General structure of the Pegasus II

The robot has 5 degree of freedom, with the following rotation restrictions.

Table 1. Manipulator restrictions.

Articulation	Angle of rotation
1 (waist)	345°
2 (shoulder)	220°
3 (elbow)	270°
4 (pitch)	270°
5 (roll)	Unlimited

2.3 Modeling

The Denavit-Hartenberg parameters of the robot are as follows.

Table 2. Denavit-Hartenberg parameters of the robot.

Join	θ	D	a	α
1	q_1	L_1	0	π/2
2	q_2	0	L_2	0
3	q_3	0	L_3	0
4	q_4	0	0	π/2
5	q_5	L_4	0	0

Where L_1 = 324 mm, L_2 = 230 mm, L_3 = 230 mm and L_4 = 42 mm.

For the inverse kinematics there exists some constraints, derived from the tasks characteristics:

- The gripper will always be oriented downwards (normal to the mounting surface of the robot), so the roll will be 0° ($q_5 = 0$).
- The pitch will be a function of q_2 and q_3: $q_4 = q_3 - q_2$.
- Angle q_1 is free.
- Angles q_2 and q_3 are determined in terms of x, y and z, and in general there will be 2 possible solutions.
- The z value for the pick and place coordinates is fixed, but the actual z value will change while the robot is moving between the storage and a mounting hole.

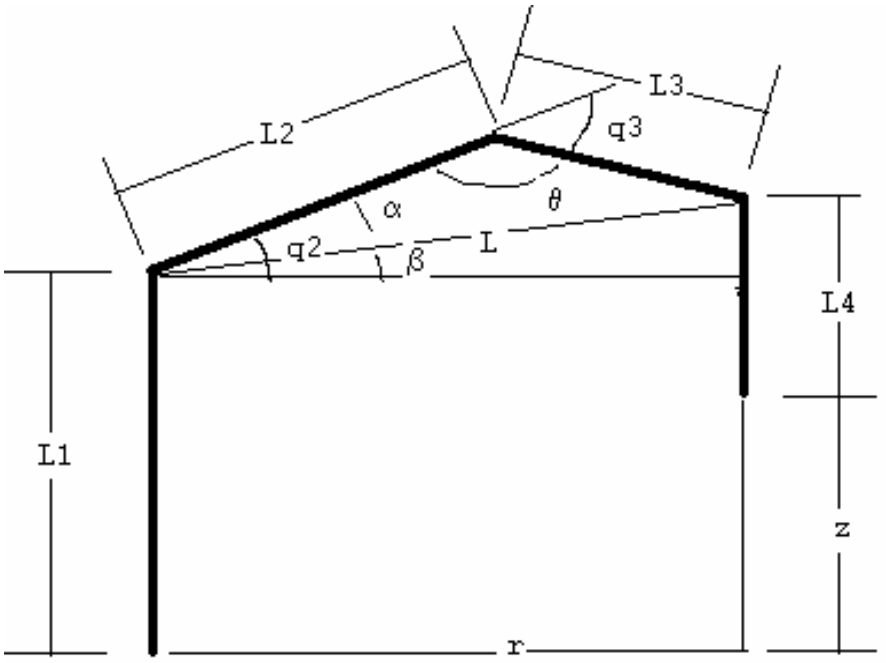

Fig. 3. Free body diagram for the inverse kinematics.

With these diagram, we can obtain the next equations, solving the inverse kinematics:

$$q_1 = \arctan\left(\frac{y}{x}\right) \tag{1}$$

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

$$\theta = \pi + q_3, \quad \cos\theta = -\cos q_3, \quad \sin\theta = -\sin q_3 \tag{3}$$

$$L^2 = x^2 + y^2 + \left(L_4 + z - L_1\right)^2 \tag{4}$$

$$L^2 = {L_2}^2 + {L_3}^2 - 2L_2L_3\cos\theta = {L_2}^2 + {L_3}^2 + 2L_2L_3\cos q_3 \tag{5}$$

$$q_3 = \arccos\left(\frac{L^2 - {L_2}^2 - {L_3}^2}{2L_2L_3}\right) \tag{6}$$

$$\beta = \arctan\left(\frac{L_4 + z - L_1}{r}\right) \tag{7}$$

$$\frac{\sin\theta}{L} = \frac{\sin\alpha}{L_3} = \frac{-\sin q_3}{L} \tag{8}$$

$$\alpha = \arcsin\left(\frac{-L_3\sin q_3}{L}\right) \tag{9}$$

$$q_2 = \alpha + \beta \tag{10}$$

$$q_4 = q_2 - q_3 - \pi/2 \tag{11}$$

$$q_5 = 0 \tag{12}$$

2.4 Simulation

A basic solid model corresponding to the links of the manipulator is used at this time. Given a sequence of desired mounting locations, the robot can follow the trajectories between a fixed position (the storage) and a mounting hole.

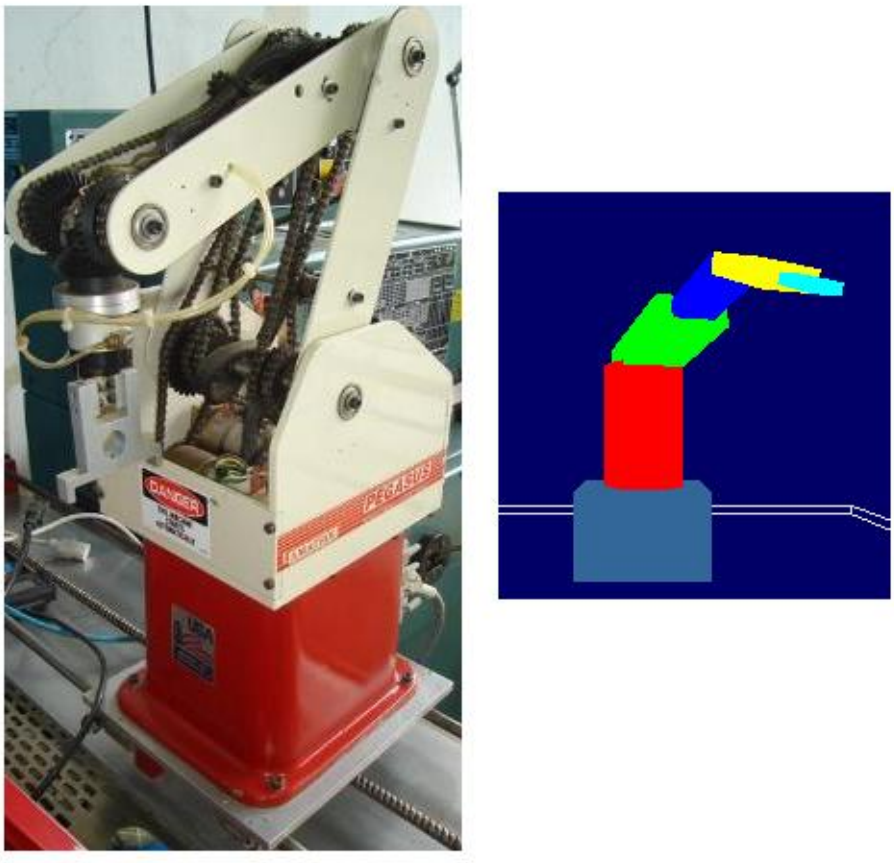

Fig. 4. Robot and virtual model.

2.5 Vision System

The vision system in use is a basic one, but the results obtained with it are good enough, and it wasn't necessary to spend money buying specialized hardware. A generic web cam and a PC are used. It is desired to have a good standard illumination, but small variations can be compensated by the processing algorithm.

2.6 Image Processing

With the image of the assembly that is required already in a file, the next step is the binarization, based on the optimum threshold.

With the binary image it is possible to recognize the objects in the image applying a cluster analysis. In this case the objects of interest are the holes for the mounting operation. This can be carried out using 4-connected or 8-connected components.

Now each component (object) is processed to obtain the coordinates of its centroid, using the formulas:

$$x = \frac{\sum_{i=1}^{n} x_i}{n} \tag{13}$$

$$y = \frac{\sum_{i=1}^{n} y_i}{n} \tag{14}$$

Where (x_i, y_i) are the coordinates of each pixel that belongs to a component and n is the total number of pixels. Next, a factor is applied to map the pixel coordinates with absolute coordinates in mm. This factor is determined by the physical size of the board and the position where it will be put in front of the robot for the assembly.

2.7 Robot Program Generation

The Pegasus robot controller allows the use of Cartesian coordinates in the application programs. In a structured world, it is possible to determine the position of the centroid of the mounting holes with the required precision, and to use these points to generate the trajectory of the manipulator.

A Cartesian coordinate for the Pegasus robot has 5 components: X, Y, Z, P (pitch) and R (roll). X, Y and Z can be expressed as inches or millimeters, and the angles normally are expressed in degrees. The Cartesian coordinates for the home position are X=0", Y=16.0", Z=21.5", P=–90°, R=0°.

The commands that can be used to move the robot and their syntax are:

- PMOVE(X coordinate, Y coordinate, Z coordinate, Pitch, Roll)
- TMOVE(X coordinate, Y coordinate, Z coordinate, Pitch, Roll)

The units are expressed as follows:

- Thousandths of an inch (X=7000 means X=7").
- Tenths of a millimeter (X=100 means X=10 mm).
- Hundreths of a degree (P=900 means P=9 degrees).

A sample program in MCL (Manufacturing Control Language) generated by the system is (4 mounting holes):

```
PMOVE <-3000,100,2000,-1221,0>
```

```
PMOVE <-3000,100,800,-942,0>
GRASP
PMOVE <-3000,100,2000,-1221,0>
PMOVE <-689,3530,2000,-1138,0>
PMOVE <-689,3530,800,-869,0>
RELEASE
PMOVE <-689,3530,2000,-1138,0>

PMOVE <-3000,100,2000,-1221,0>
PMOVE <-3000,100,800,-942,0>
GRASP
PMOVE <-3000,100,2000,-1221,0>
PMOVE <-290,2851,2000,-1236,0>
PMOVE <-290,2851,800,-951,0>
RELEASE
PMOVE <-290,2851,2000,-1236,0>

PMOVE <-3000,100,2000,-1221,0>
PMOVE <-3000,100,800,-942,0>
GRASP
PMOVE <-3000,100,2000,-1221,0>
PMOVE <96,3542,2000,-1147,0>
PMOVE <96,3542,800,-878,0>
RELEASE
PMOVE <96,3542,2000,-1147,0>

PMOVE <-3000,100,2000,-1221,0>
PMOVE <-3000,100,800,-942,0>
GRASP
PMOVE <-3000,100,2000,-1221,0>
PMOVE <495,2862,2000,-1232,0>
PMOVE <495,2862,800,-949,0>
RELEASE
PMOVE <495,2862,2000,-1232,0>

STOP
```

3 Experimental Work

So far the model for the robot is complete and the programs for simulation, image acquisition and processing, and robot program generation are also complete.

An example mounting board is shown in the next figure.

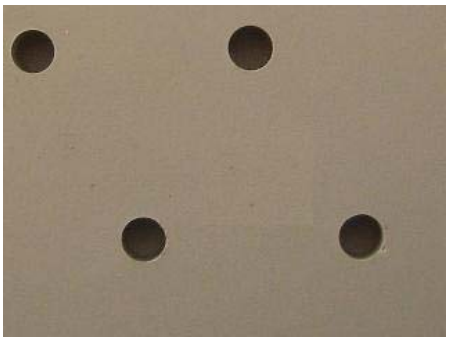

Fig. 5. Acquired image (the actual color of the board is gray).

The binary image after processing is:

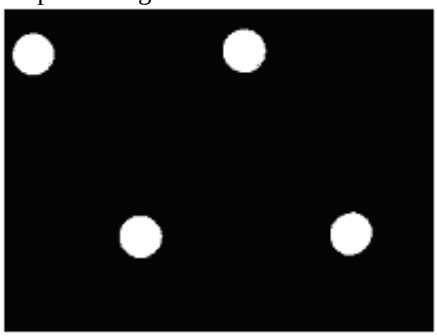

Fig. 6. Binarized image.

The parameters obtained using the algorithm, expressed in pixels, are:

Centroid 1 : 22.1160 34.0973
Centroid 2: 102.0321 169.8586
Centroid 3: 179.2522 31.5304
Centroid 4: 258.9001 167.6744

4 Conclusions and Future Work

At this moment, only basic assembly operations have been done, but in the future it will be possible to add another camera to obtain new data from the assembly operation required, allowing the execution of more advanced assembly between parts.

The robot used, despite the industrial quality of its components, is intended for educational purposes. It will be possible to replace the robot with an industrial one, and with the appropriate changes in the parameters of the robot and the syntax of the instructions, the simulation and program generation will be possible.

The web cam can also be replaced with one designed for industrial use, using a good illumination scheme, which can lead to better parameter determination for the assembly.

For simulation purposes it is possible to do a CAD definition of the robot, export it to a VRML or OBJ format, and use the file to obtain a better approximation to the real world.

References

1. Soto Cajiga J. A., Vargas Soto J. E., Pedraza Ortega J. C.: Trajectories generation for a robot manipulator using image processing and splines. Segundo Congreso Internacional de Ingeniería, Querétaro, Qro., México (2006).
2. Gamiño M., Pedraza J. C., Ramos J. M., Gorrostieta E.: Matlab–C++ interface for a flexible arm manipulator simulation using multi-language techniques. Proceedings of the fifth Mexican international conference on artificial intelligence (MICAI'06). México (2006).
3. Myler H., Weeks A.: Computer imaging recipes in C. Prentice-Hall, USA (1993).
4. Kurfess, Thomas R. (Editor): Robotics and automation handbook. 1st edition. CRC Press, USA (2005).
5. Viramontes Reyna, J. L., Pedraza Ortega, J. Carlos, González Galván, Emilio J.: Vision-based manipulator control using a monocular vision system. 5th Internatiomal Simposium on Robotics and Automation, San Miguel Regla, Hidalgo, México (2006).
6. Gonzalez, Rafael C., Woods, Richard E.: Digital image processing, 2nd edition. Prentice-Hall, USA (2002).