ABCM Symposium Series in Mechatronics - Vol. 1 - pp.134-143
Copyright © 2004 by ABCM

AN OFF-LINE ROBOT PROGRAMMING SYSTEM INCLUDING WORKCELL AND ROBOT CALIBRATION

José Maurício S. T. da Motta
Universidade de Brasília, Engenharia Mecânica e Mecatrônica, Grupo de Automação e Controle – GRACO, Brasília–DF.
E-mail: jmmotta@unb.br

Carlos Augusto Gurgel de Sousa
Universidade de Brasília, Engenharia Mecânica e Mecatrônica, Grupo de Automação e Controle – GRACO, Brasília – DF

Fábio Abdalla Afonso
Universidade de Brasília, Engenharia Mecânica e Mecatrônica, Grupo de Automação e Controle – GRACO, Brasília – DF

Abstract. Off-line programming provides an essential link for CAD/CAM. The implementation of off-line programming encounters problems in: a) generalized programming systems that are independent of both robots and robot applications, b) incompatibility between robots and programming systems, c) accounting for errors and inaccuracies that exist in the real world. Errors can be reduced substantially by using proper robot calibration. Robot Calibration is a term applied to the procedures used in determining actual values that describe the geometric dimensions and mechanical characteristics of a robot or multi-body structure. A robot calibration system must consist of appropriate robot modeling techniques, accurate measurement equipment, and reliable model parameter determination methods. For practical improvement of a robot's absolute accuracy, error compensation methods are required that use calibration results. The objective of this article is to discuss a method to construct kinematic models for robot calibration assuring model continuity and to present a computer program to carry out this task automatically before the identification step as part of an off-line programming system. It is also discussed the inclusion of robot calibration results in the off-programming system aiming at improving the robot global accuracy through re-calculation of spatial world coordinates.

Keywords: Off-line Programming, Robot Calibration, Model Optimization, Kinematic Models.

1. Introduction

The complete control of a manufacturing cell involves perfect task synchronization among all of its robots and equipment. Each task set may be tested in software which performs graphic simulation before execution, enabling all of the synchronization signals between the equipment, the program for each robot and the optimal sequence of task execution and cell layout to be determined in advance. After testing, the programs are downloaded from the host computer to the manufacturing cell and tested. Programs already executed may also be uploaded, to be simulated and edited (Carvalho et al., 1998).

A distinction has to be made between simple systems, and costly off-line programming and simulation systems. The former allows existing robot programs to be edited or archived and are used to develop the program frame, usually without the determination of the individual positions and orientations (poses) of the robot motion. The latter costly off-line programming and simulation systems are interactively operated and allow a description to be made of the nominal path in the base coordinate system of the robot.

The ideal case would be to load an off-line generated robot program down into the robot control and to execute it without any adaptation. However, there are inevitably differences between the computer model used to perform the graphic simulation and the real world. This is because the effective use of off-line programming in industrial robots requires, additionally, a knowledge of tolerances of the manufacturing components in order to enable realistic planning, i.e. to reduce the gap between simulation and reality.

In an actual robot system programming this is still a relatively difficult task and the generated programs still require manual on-line modification at the shop floor. A typical welding line with 30 robots and 40 welding spots per robot takes about 400 hours for robot teaching (Bernhardt, 1997). The difficulties are not only in the determination of how the robot can perform correctly its function, but also for it to be able to achieve accurately a desired location in the workspace. Robot pose errors are attributed to several sources, including the constant (or configuration-independent) errors in parameters (link lengths and joint offsets), deviations which vary predictably with position (e.g., compliance, gear transmission errors) and random errors (e.g., due to the finite resolution of joint encoders). Constant errors are referred to as geometric errors and variable errors are referred to as non-geometric errors (Roth et al., 1987). According to Bernhardt (1997) and Schröer (1993), constant errors represent approximately 90% of the overall robot pose errors. Industrial robots usually show pose errors from about 5 to 15mm, even when they are new, and after proper calibration these error can be reduced to less than 0.5mm (Bernhardt, 1997, Motta et al., 2001).

Robot calibration is an integrated process of modeling, measurement, numeric identification of actual physical characteristics of a robot, and implementation of a new model. The calibration procedure first involves the development of a kinematic model whose parameters represent accurately the actual robot. Next, specifically selected robot characteristics are measured using measurement instruments with known accuracy. Then a parameter identification procedure is used to compute the set of parameter values which, when introduced in the robot nominal model, accurately represents the measured robot behavior. Finally, the model in the position control software is corrected.

Important to robot calibration methods is an accurate kinematic model that is complete, minimal and continuous, and has identifiable parameters. Researchers have used specific kinematic models that depend on a particular robot geometry and/or calibration method. Although the identifiability of model parameters has been addressed (e.g., Everett, 1988, Zhuang, 1992), methods to determine precisely which model can be used for a given relative joint configuration have to date not been available. Because of the model function complexity, parameter dependencies leading to matrix rank deficiencies in the linearized model cannot always be predefined from inspection of the robot model. Rank deficiencies produce no unique solution of the numerical problem (Motta and McMaster, 1999).

This article shows an off-line robot programming system especially developed for the ABB IRB-2000 robot in JAVA and API JAVA3D languages, allowing full world modeling. Techniques to optimize the kinematic model are presented and implemented in the software. The optimized model can be used further to compensate the model errors in an off-line programming system, enhancing significantly the robot kinematic model accuracy. The optimized model can be constructed in an easy and straight operation, through automatic assignment of joint coordinate systems and geometric parameter to the robot links. World coordinate systems are directly converted to the robot controller coordinate system in an easy and friendly operation. It is also discussed the inclusion of robot calibration results in the off-programming system aiming at improving the robot global accuracy through re-calculation of spatial world coordinates.

2. OFF-LINE PROGRAMMING

Off-line programming is, by definition, the technique of generating a robot program without using a real machine. It presents several advantages over the on-line method, some of which are cited (Bien, 1998; Chan, 1991; Carvalho, 1998): a) programs are prepared without interruptions of robot operation, resulting in reduction of robot down time, which means flexibility and an increase in productivity; b) removal of the programmer from the potentially dangerous environment, as most of the program development is executed away from the robot. Thus, the time during which the programmer is at risk from aberrant robot behavior is reduced; c) there is a greater possibility for optimization of the workspace layout and the planning of robot tasks; d) new programs can include previously developed routines; e) program changes can be incorporated quickly by substituting only the necessary part of the program; f) signals from sensors can be incorporated into programs; g) information from the environment (i.e. computer-integrated-manufacturing systems, CAD/CAM systems) can be incorporated into programs; h) it permits verification of the robot behavior through graphical simulation and allows for the correction of any error in the program.

2.1. Off-line Programming Software

Off-line Programming Software were built using the JAVA language which is an object-oriented computer language which has become popular for the following reasons: a) easiness to use; b) integration to UML (Unified Modeling Language); c) support to C/C++ languages; d) architecture independence; e) advanced support to networks; f) documentation availability.

This language encompasses three important needs of off-line programming software: support to computer graphics, serial communication and TCP/IP communication. Computer Graphics have an elegant approach by using API (Application Programming Interface) JAVA3D.

The simulator was implemented including the following characteristics: a) workspace visualization; b) trajectory planning/design; c) communication to the robot control; d) system navigation.

The connection between these characteristics can be seen in Fig. 1.

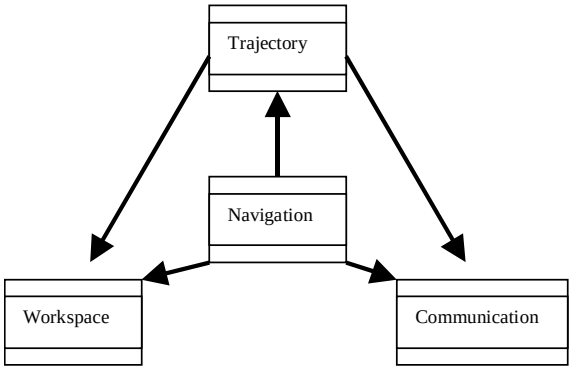

Figure 1. Simulator modules

The Workspace Module has the following attributes: a) stores screens and data collected from the screen; b) performs data format; c) allows the robot tool construction; d) allows table construction and object control in the workspace; e) support robot model construction.

Figure 2 shows a class diagram of the Workspace Module.

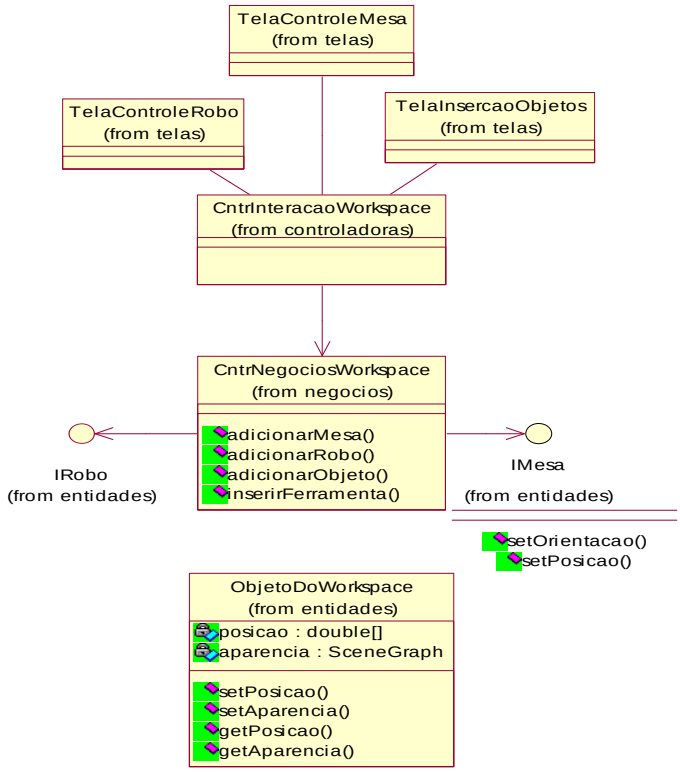

Figure 2. Class Diagram of the Workspace Module

The Navigation Module assures the interaction between the user and other subsystems. It is responsible for providing an interface to workspace visualization, communication to the robot and assigning trajectories through the screen.

The Communication Module is built up as a series of classes that implement the Pattern Command. This command provides a generic interface to auxiliary components to which the controller sends information. The Pattern Command in the Communication Module is shown in a diagram in Fig. 3.

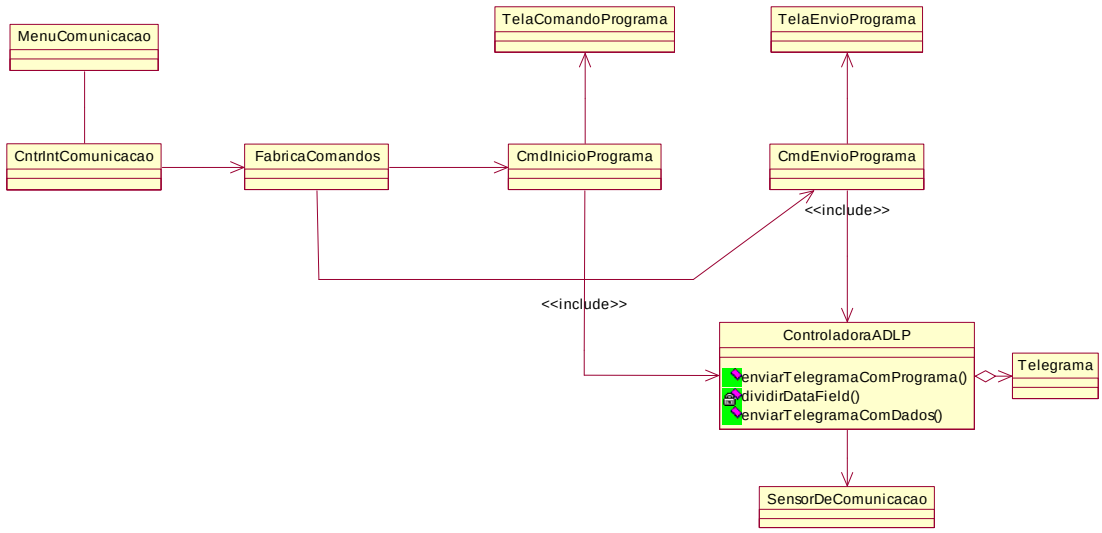

Figure 3. *Pattern Command* in the Communication Module

3. A Singularity-free Approach for Kinematic Models

Single minimal modeling convention that can be applied uniformly to all possible robot geometries cannot exist owing to fundamental topological reasons concerning mappings from Euclidean vectors to spheres (Gottlib, 1986, Baker, 1990). However, after investigating many topological problems in robots, concerning inverse kinematics and singularities, Baker (1990) suggested that the availability of a repertoire of methods for determining whether or not inverse kinematic functions can be defined on various subsets of the operational spaces would be useful, but even more important, a collection of methods by which inverse functions can actually be constructed in specific situations. Another insightful paper about robot topologies was published by Gottlieb (1986), who noted that inverse functions can never be entirely successful in circumventing the problems of singularities when pointing or orienting.

Motivated by the above arguments Schröer (1993) and Schröer et al. (1997) presented a concept to find complete, minimal and model-continuous kinematic models for all combinations and configurations of revolute and prismatic joints. Mathematically, model-continuity is equivalent to continuity of the inverse function T^{-1}, where T is the product of elementary transformations (rotation and translation) between joints. From this, the definition of parameterization's singularity can be stated as a transformation $T_s \in E$ (parameterization's space of the Euclidean Group - 3 rotations and 3 translations), where the parameter vector $p \in R^6$ (p represents distance or angle) exists such that the rank of the Jacobian $J_s = dT_s/dp$ is smaller than 6. In other way, each parameterization T can be investigated concerning their singularities detecting the zeroes of determinant $\det(J^T.J)$ considered as a function of parameter p.

Thus, investigating the main kinematic modeling conventions one can represent the transformation between links in the Euclidean Group as

$$T = T_x(p_x).T_y(p_y).T_z(p_z).R_z(\gamma).R_y(\beta).R_x(\alpha) \tag{1}$$

where p_x, p_y, p_z are translation coordinates and α, β, γ are rotation coordinates for the axes x, y and z respectively. Then,

$$T = \begin{bmatrix} C\gamma.C\beta & C\gamma.S\beta.S\alpha - S\gamma.C\alpha & C\gamma.S\beta.C\alpha + S\gamma.S\alpha & Px \\ S\gamma.C\beta & S\gamma.S\beta.S\alpha + C\gamma.C\alpha & S\gamma.S\beta.C\alpha - C\gamma.S\alpha & Py \\ -S\beta & C\beta.S\alpha & C\beta.C\alpha & Pz \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} n_x & o_x & a_x & p_x \\ n_y & o_y & a_y & p_y \\ n_z & o_z & a_z & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2}$$

where $C\gamma=\cos(\gamma)$ and $S\beta=\sin(\beta)$ and so fourth.

The Jacobian J(q) of the function T(q(t)), where q is the joint variable and t is the time, can be determined expressing the equality (Spong & Vidyasagar, 1989)

$$\begin{bmatrix} v \\ \omega \end{bmatrix} = J(q).\dot{q} \tag{3}$$

where v and ω are the linear and angular absolute velocity of each joint respectively. One can decompose singular positions simply making $\det(J(q)) = 0$, resulting in the manifold of singularities in Eq. (4a)

$$T_s = \begin{bmatrix} 0 & o_x & a_x & p_x \\ 0 & o_y & a_y & p_y \\ \pm 1 & 0 & 0 & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad \text{(a)} \qquad \mathbf{T_s} = \begin{bmatrix} n_x & o_x & 0 & p_x \\ n_y & o_y & 0 & p_y \\ n_z & o_z & \pm 1 & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad \text{(b)} \tag{4}$$

This manifold represents $\gamma = \pm\pi/2$ and $\beta = \pm\pi/2$ (Eq. 2), which means there is a singularity when y and z axes are parallel.

The Denavit-Hartemberg convention (parameterization) (Paul, 1981) leads to an elementary transformation represented by

$$T(\theta,p_z,p_x,\alpha) = R_z(\theta).T_z(p_z).T_x(p_x).R_x(\alpha) \tag{5}$$

Following the same procedure as before the manifold of singularities is shown in Eq. (4b). This result consists of all elements represented as parallel rotary joints. This can be verified by observing the third column showing the representation of one joint axis (z) into the previous one.

The Hayati-convention (Hayati & Mirmirani, 1985) can be represented by

$$T(\theta,p_x,\alpha,\beta) = R_z(\theta).T_x(p_x).R_x(\alpha).R_y(\beta) \quad (6)$$

There are two manifolds of singularities,

$$T_s = \begin{bmatrix} n_x & o_x & a_x & p_x \\ n_y & o_y & a_y & p_y \\ n_z & o_z & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad \text{or} \quad T_s = \begin{bmatrix} n_x & o_x & \lambda p_x & p_x \\ n_y & o_y & \lambda p_y & p_y \\ n_z & o_z & \lambda p_z & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (7)$$

These representations show that if the distal *z* axis is in the same *x-y* plane of the proximal coordinate system (so perpendicular) or points to its origin, then there is a singularity.

The Veitschegger convention (Veitschegger & Wu, 1986) is a 5-dimensional parameterization as

$$T = R_z(\theta).T_z(p_z).T_x(p_x).R_x(\alpha).R_y(\beta) \quad (8)$$

The manifolds of singularities are

$$T_s = \begin{bmatrix} n_x & o_x & a_x & 0 \\ n_y & o_y & a_y & 0 \\ n_z & o_z & a_z & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad \text{or} \quad T_s = \begin{bmatrix} n_x & o_x & a_x & \lambda a_x \\ n_y & o_y & a_y & \lambda a_y \\ n_z & o_z & 0 & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (9)$$

These representations show that if the origin of the distal joint coordinate system is on the *z*-axis of the proximal one, or the *z* axes are perpendicular and intercept each other, then there is a singularity. However this convention is not minimal.

Using the same technique presented so far for prismatic joints sets of parameterizations can be used so fourth. The results can be outlined in a list together with their application ranges. The set is a complete, minimal and model-continuous kinematic model [Schröer et al., 1997]. The transformations are modeled from a current frame C to a revolute joint, J_R, or to a prismatic joint, J_P, or a fixed and arbitrarily located target frame, TCP-frame. Some of them are not unique since other solutions can fulfill the requirements. Additional elementary transformations are possible, but any added elementary transformation is redundant and thus, cannot be identified. A non-expanded model can always be found, which describes the same kinematic structure as the expanded model.

Elementary transformations which are part of the complete, minimal and model-continuous sub-model being identified are marked bold (Schröer, 1997) in Tab. (1):

Table 1. Elementary transformations between joints for complete, minimal and continuous kinematic models.

	$\perp J_R$	$\|\| J_R$	$\perp J_P$	$\|\| J_P$
Transformation from robot base frame to first joint				
B	$P_X = (\mathbf{T_Y,T_Z,R_Z,R_X})^{(1)}$ $P_Y = (\mathbf{T_X,T_Z,R_Z,R_X})^{(2)}$	$P_Z=(\mathbf{T_X,T_Y,R_X,R_Y})$	$P=(T_X,T_Y,T_Z, \mathbf{R_Z,R_X})$	$P=(T_X,T_Y,T_Z,\mathbf{R_X,R_Y})$
Transformations between consecutive joint frames				
J_R	$P= (\mathbf{R_Z,T_Z,T_X,R_X},T_Z)$ D-H convention	$P=(\mathbf{R_Z,T_X,R_X,R_Y},\mathbf{T_Z})^{(3)}$ Hayati convention	$P=(\mathbf{R_Z},T_X,T_Y,T_Z,\mathbf{R_X})$	$P=(R_Z,T_X,T_Y,T_Z,\mathbf{R_X,R_Y})$
J_P	$P=(\mathbf{T_Z,R_Z,T_X,R_X},T_Z)$	$P=(T_Z,\mathbf{T_X,T_Y,R_X,R_Y},$ $T_Z)$	$P=(T_Z,T_X,T_Y,\mathbf{R_Z,R_X})$	$P=(T_Z,T_X,T_Y,\mathbf{R_X, R_Y})$
Transformation from last joint to TCP				
	$\perp$ TCP		$\|\|$ TCP	
J_R	$P=(R_Z,\mathbf{T_X,T_Y,T_Z,[R_Z,R_Y, R_Z]})$		$P=(R_Z,\mathbf{T_X,T_Y,T_Z,[R_Z, R_Y, R_X]})$	
J_P	$(R_Z, \mathbf{T_X, T_Y, T_Z, [R_Z, R_Y, R_Z]})$		$P = (\mathbf{T_Z, T_Y, T_X, [R_Z, R_Y, R_X]})$	

(1) - If joint axis is near **x**-axis of frame B.
(2) - If joint axis is near **y**-axis of frame B.
(3) - assumption: joint axes are not identical

$\perp$ - means perpendicular to
$\|\|$ - means parallel to

4. Kinematic Modeling – Assignment of Coordinate Frames

The first step to kinematic modeling is the proper assignment of coordinate frames to each link. Each coordinate system here is orthogonal, and the axes obey the right-hand rule.

For the assignment of coordinate frames to each link one may move the manipulator to its zero position. The zero position of the manipulator is the position where all joint variables are zero. This procedure may be useful to check if the zero positions of the model constructed are the same as those used by the controller, avoiding the need of introducing constant deviations to the joint variables (joint positions).

The first step to assign coordinate frames to joints is to make the z-axis coincident with the joint axis. This convention is used by many authors and in many robot controllers (McKerrow, 1995, Paul, 1981). For a prismatic joint, the direction of the z-axis is in the direction of motion, and its sense is away from the joint. For a revolute joint, the sense of the z-axis is towards the positive direction of rotation around the z-axis. The positive direction of rotation of each joint can be easily found by moving the robot and reading the joint positions on the robot controller display.

According to McKerrow (1995) and Paul (1981), the base coordinate frame (robot reference) may be assigned with axes parallel to the world coordinate frame. The origin of the base frame is coincident with the origin of joint 1 (first joint). This assumes that the axis of the first joint is normal to the x-y plane. This location for the base frame coincides with the PUMA manufacturer's defined base frame.

Afterwards coordinate frames are attached to the link at its distal joint (joint farthest from the base). A frame is internal to the link it is attached to (there is no movements relative to it), and the succeeding link moves relative to it. Thus, coordinate frame *i* is at joint *i+1*, that is, the joint that connects link *i* to link *i+1*.

The origin of the frame is placed as following: if the joint axes of a link intersect, then the origin of the frame attached to the link is placed at the joint axes intersection; if the joint axes are parallel or do not intersect, then the frame origin is placed at the distal joint; subsequently, if a frame origin is described relative to another coordinate frame by using more than one direction, then it must be moved to make use of only one direction if possible. Thus, the frame origins will be described using the minimum number of link parameters.

The x-axis or the y-axis have their direction according to the convention used to parameterize the transformations between links (Tab. 1). At this point the homogeneous transformations between joints must have been already determined. The other axis (x or y) can be determined using the right-hand rule.

A coordinate frame can be attached to the end of the final link, within the end-effector or tool, or it may be necessary to locate this coordinate frame at the tool plate and have a separate hand transformation. The z-axis of the frame is in the same direction as the z-axis of the frame assigned to the last joint (n-1).

The end-effector or tool frame location and orientation is defined according to the controller conventions. Geometric parameters of length are defined to have an index of joint and direction. The length *pni* is the distance between coordinate frames *i - 1* and *i* , and *n* is the parallel axis in the coordinate system *i - 1*. Figure 4 shows the above rules applied to the PUMA-560 robot with all the coordinate frames and geometric features.

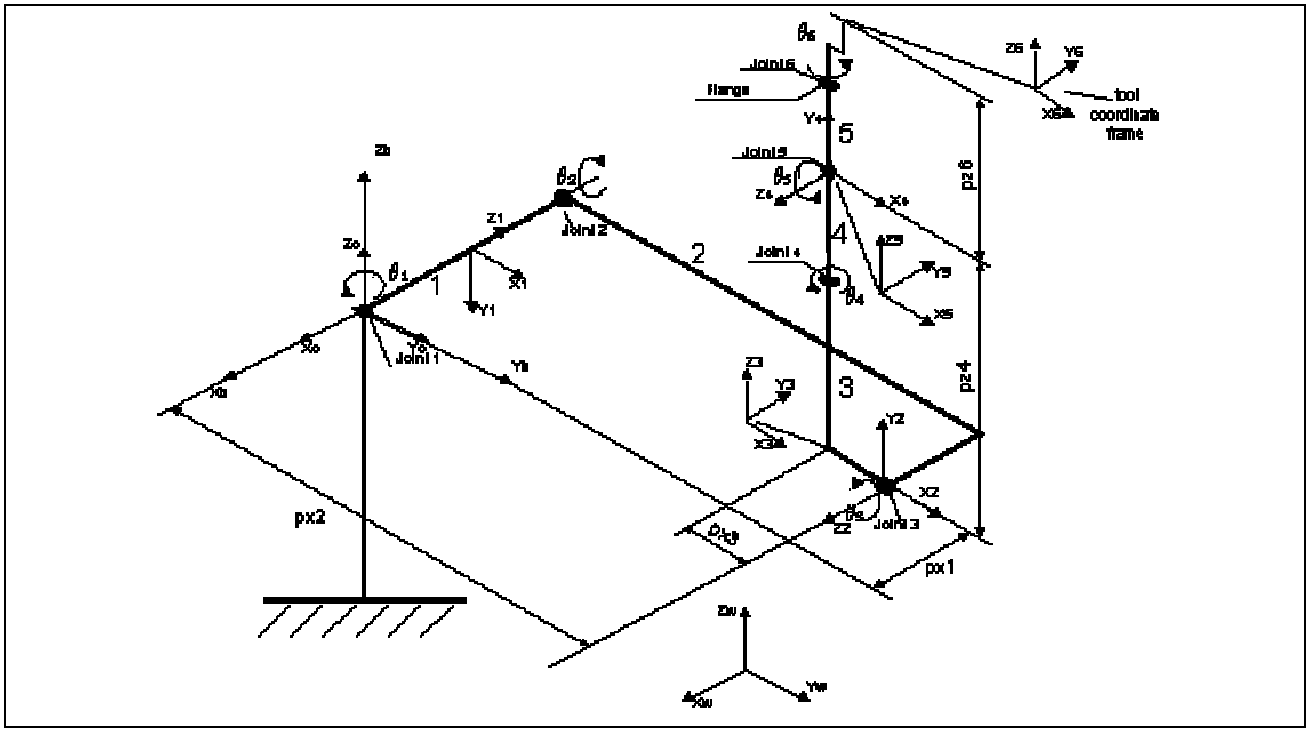

Figure 4. Skeleton of the PUMA 560 Robot with coordinate frames in the zero position and geometric variables for kinematic modeling. (Out of scale)

5. Parameter Identification

The most important part of parameter identification procedures is concerned with numerical methods. Procedures in which model parameters are identified from several measured robot end-effector poses require numerical optimization methods (Motta & McMaster, 1999). These methods utilize local linearization of the non-linear robot model and then an iterative solution of the non-linear least-squares problem.

5.1. Parameter Identification Kinematic Model

The kinematic equation of the robot manipulator is obtained by consecutive homogeneous transformations from the base frame to the last frame. Thus,

$$\hat{T}^0{}_N = \hat{T}^0{}_N(k) = T^0{}_1 . T^1{}_2 \ldots T^{N-1}{}_N = \prod_{i=1}^{N} T^{i-1}{}_i \tag{10}$$

where N is the number of joints (or coordinate frames), $k = [p_1^T \quad p_2^T \quad \ldots \quad p_n^T]^T$ is the parameter vector for the manipulator, and p_i is the link parameter vector for the joint i, including the joint errors. The exact link transformation $A^{i-1}{}_i$ is (Driels and Pathre, 1990):

$$A^{i-1}{}_i = T^{i-1}{}_i + dT_i \quad , \quad dT_i = dT_i(\delta p_i) \tag{11}$$

where δp_i is the link parameter error vector for the joint i.

The exact manipulator transformation $\hat{A}^0{}_{N-1}$ is

$$\hat{A}^0{}_N = \prod_{i=1}^{N} (T^{i-1}{}_i + dT_i) = \prod_{i=1}^{N} A^{i-1}{}_i \tag{12}$$

Thus,

$$\hat{A}^0{}_N = \hat{T}^0{}_N + d\hat{T} \quad , \quad d\hat{T} = d\hat{T}(q, \delta k) \tag{13}$$

where $\delta k = [\delta p_1^T \, \delta p_2^T \ldots \delta p_n^T]^T$ is the manipulator parameter error vector and *q* is the vector of joint variables $[\theta_1^T, \theta_2^T, \theta_N^T]^T$. It must be stated here that $d\hat{T}$ is a non-linear function of the manipulator parameter error δk.

Considering *m* the number of measurement positions it can be stated that

$$\hat{A} = \hat{A}^0{}_N = \hat{A}(q, k) \tag{14}$$

where $\hat{A}: \Re^n \times \Re^N$ is function of two vectors with *n* and N dimensions, *n* is the number of parameters and N is the number of joints (including the tool). It follows that

$$\mathbf{\hat{A}} = \mathbf{\hat{A}}^0{}_N = \mathbf{\hat{A}}(\mathbf{q}, k) = (\hat{A}(q_1, k), \ldots, \hat{A}(q_m, k))^T \quad : \Re^n \times \Re \tag{15}$$

and

$$\mathbf{d\hat{T}} = \mathbf{d\hat{T}}(\mathbf{q}, \delta k) = (d\hat{T}(q_1, \delta k), \ldots, d\hat{T}(q_m, \delta k)^T \quad : \Re^n \times \Re^{mN} \tag{16}$$

All matrices or vectors in bold are functions of *m*. The identification itself is the computation of those model parameter values $k^* = k + \delta k$ which result in an optimal fit between the actual measured positions and those computed by the model, i.e., the solution of the non-linear equation system

$$\mathbf{B}(\mathbf{q}, k^*) = \mathbf{M}(\mathbf{q}) \tag{17}$$

where **B** is a vector formed with position and orientation components of $\mathbf{\hat{A}}$ and

$$\mathbf{M}(\mathbf{q}) = (M(q_1), \ldots, M(q_m))^T \quad \in \Re^{\phi m} \tag{18}$$

are all measured components and ϕ is the number of measurement equations provided by each measured pose. If orientation measurement can be provided by the measurement system then 6 measurement equations can be formulated per each pose. If the measurement system can only measure position, each pose measurement can supply data for 3 measurement equations per pose and then **B** includes only the position components of **Â**.

The solution of Eq. (17) is presented in detail by Motta & McMaster (1999) and is not repeated here for short. The determination of k* allows a much more accurate kinematic model by correcting the parameter vector.

World coordinates can be compensated correcting the robot base coordinates as shown in Fig. 5. X_{target} are the desired coordinates that can be used as input to the inverse nominal kinematic model, $1/f_N$ (in the computer), obtaining nominal joint positions, q_N. These joint position values feed the corrected kinematic model (calibrated), f_{corr}, obtaining end-effector positions which can be compared to X_{target} to calculate the compensated coordinates, X_{comp}. These compensated coordinates are the input to the inverse nominal model to calculate the real joint position values, q_{real}, which can be sent to the robot controller model, f_{cont}. Thus, the end-effector position will reach the real target coordinates, X_{real}.

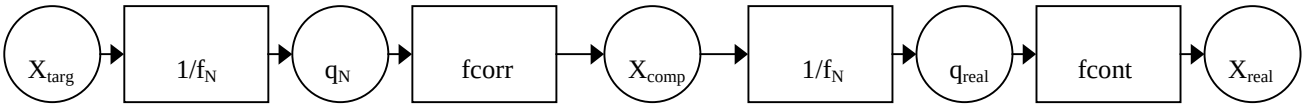

Figure 5. Robot Base Coordinate compensation block diagram.

6. Results

The computer program for off-line programming ROBOPRO was developed as a graphic tree in JAVA3D. Each tree node is a TransforGroup representing one robot link. A screen visualization for the IRB 2000 Robot class can be seen in Fig. 6. The graphic model was created on AutoCad™ software, but could be modeled with any software with image outputs in the formats DXF, 3DS, VRLM, or other formats supported by APIs of NCSA.

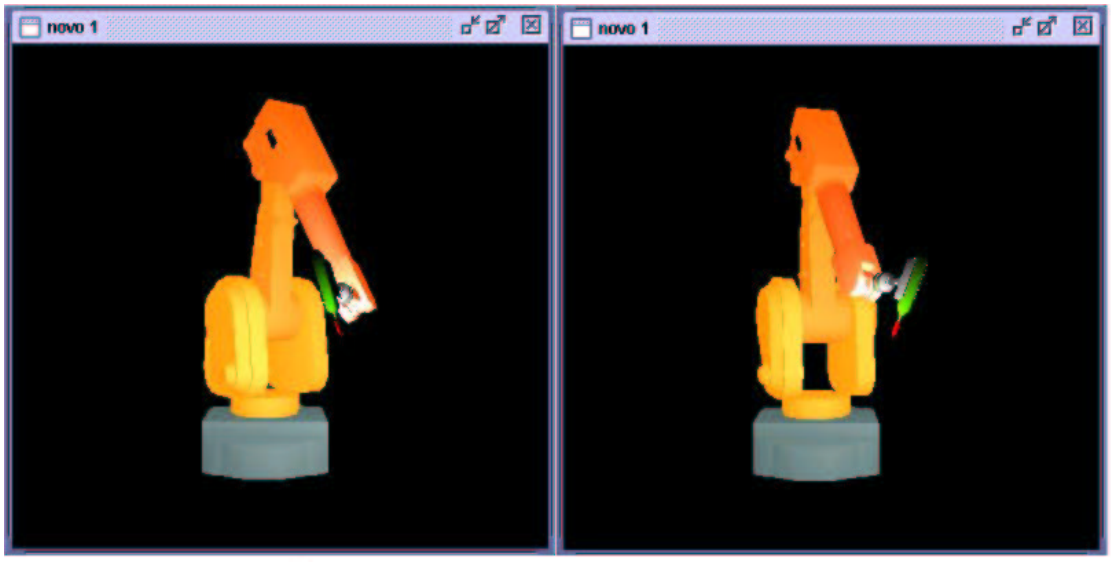

Figure 6. Graphic environment for robot model visualization.

The computer program for construction of kinematic models for robot calibration was developed in API-JAVA3D language. Figure 7 illustrates the input window for entering model parameter data. As shown in the figure, the system needs 3 types of information about the robot joints. The first is concerned to the type of joint: prismatic or revolute. The second refers to the position of the joint coordinate system origin relative to the previous joint system, measured according to the Base Coordinate System (BCS). The last input information is the joint axis orientation (*z* axis), measured relative to the BCS.

Model joint data are entered one by one up to the TCP (Tool Center Position), which is the last point in the model geometric sequence.

Figure 8 shows the final model obtained by the computer program, showing the complete robot model for the PUMA-560 on the screen. All kinematic parameters and homogeneous transformations are already determined at this stage. The model is then ready for the parameter identification step.

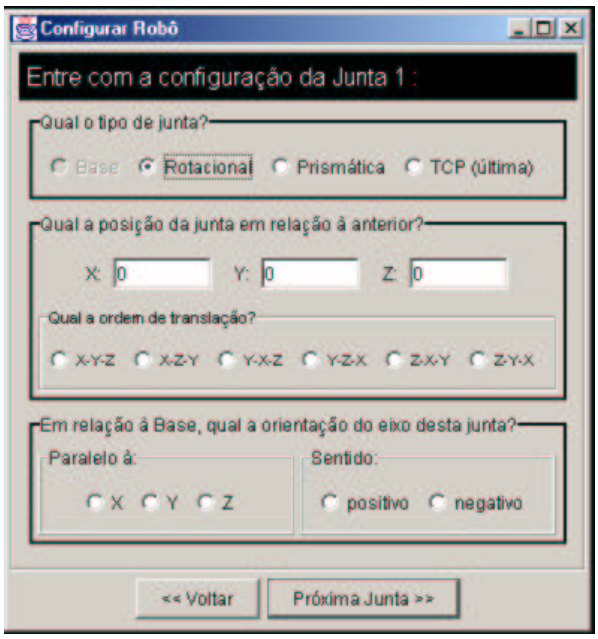

Figure 7. Input data window for entering robot model joint positions.

Comparing Fig. 4 to Fig. 8 it is evident that both are the same.

The software was tested with many different robot systems as IRB-2000, IRB-2400 and specially designed ones. Up to date all tests were well succeeded.

7. Conclusions

It is discussed the main topics of off-line robot programming and practical aspects that bring about difficult solutions in a day-by-day industrial robot application in welding tasks. The mismatch between nominal robot models and the actual robot model can be reduced drastically by using a robot calibration system to correct robot pose errors (low accuracy). This can be easily carried out through re-calculation of world coordinates and robot base coordinates. Software were developed to implement an off-line programming system including robot pose error corrections. This system was conceived specifically for the IRB-2000 robot but it can be easily extended to any other robot. The system is ready to be used for planning and simulation of welding tasks, with robot pose errors reduced by implementing robot calibration results.

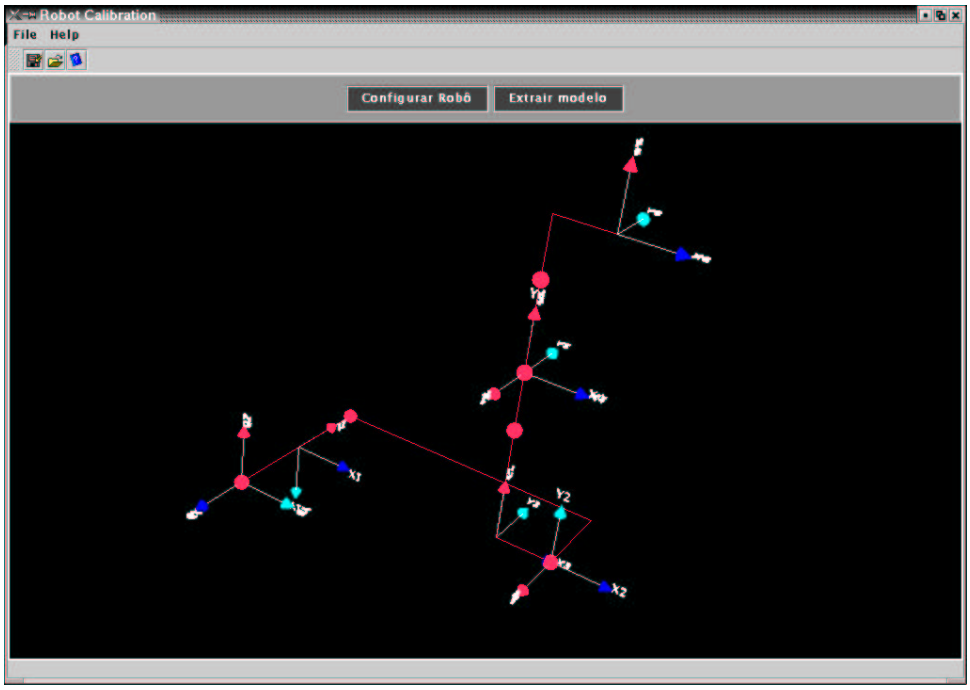

Figure 8. PUMA 560 Robot model obtained by the software.

It was also presented a computer system to construct optimized kinematic models for robot calibration. The rules adopted to select proper joint coordinate systems in the kinematic model are based on a singularity-free approach that avoids ill-conditioned mathematical models in the identification process. The system has been tested in many robot topologies and results show coincidence between the computer program outputs and predicted models. So the way is paved for a complete robot calibration system encompassing modeling, data acquisition from measurements, parameter identification and model compensation, bringing about a system that can be linked to robot off-line programming systems, allowing considerable improvements to the accuracy of this type of robot programming.

8. References

Baker, D.R., 1990, “Some topological problems in robotics”, The Mathematical Intelligencer,Vol.12, No.1, pp. 66-76.

Bernhardt, R., 1997, “Approaches for commissioning time reduction”, Industrial Robot, Vol. 24, No. 1, pp. 62-71.

Bien, C., 1998,"Simulation a necessity in safety engineering", Robot World, Vol. 10, No. 4, pp. 22-27.

Carvalho, G. C., Siqueira, M. L. and Alfaro, S. C. A., 1998, "Off-line programming of flexible welding manufacturing cells" , Journal of Materials Processing Technology, No. 78, pp. 24-28.

Chan, K. C., Lin, G. C. I. amd Wang, T. C., 1991,"CAD based off-line robot programming", Proceedings of the International Conference on Computer-Integrated Manufacturing, ICCIM, Singapore, pp. 381-384.

Driels, M.R. and Pathre, U.S. (1990) “Significance of Observation Strategy on the Design of Robot Calibration Experiments”, Journal of Robotic Systems, Vol. 7, No. 2, pp. 197-223.

Everett, L.J. and Hsu, T-W, 1988, “The Theory of Kinematic Parameter Identification for Industrial Robots”, Transaction of ASME, No. 110, pp. 96-100.

Gottlieb, D.H., 1986, “Robots and Topology”, Proceedings of the IEEE International Conference on Robotics and Automation, pp. 1689-1691.

Hayati, S. and Mirmirani, M., (1985), "Improving the Absolute Positioning Accuracy of Robots Manipulators", Journal of Robotic Systems, Vol. 2, No. 4, pp. 397-413.

McKerrow, P. J. (1995), "Introduction to Robotics", 1st ed., Ed. Addison Wesley, Singapore.

Motta, J. M. S. T. and McMaster, R. S., 1999, "Modeling, Optimizing and Simulating Robot Calibration with Accuracy Imporovements", Journal of the Brazilian Society of Mechanical Sciences, Vol. 21, No. 3, pp. 386-402.

Motta, J. M. S. T., Carvalho, G. C. and McMaster, R. S., 2001, "Robot Calibration Using a 3-D Vision-Based Measurement System With a Single Camera ", Robotics and Computer Integrated-Manufacturing, Ed. Elsevier Science, U.K., v. 17, n. 6, p. 457-467.

Paul, R. P., (1981), "Robot Manipulators - Mathematics, Programming, and Control", Boston, MIT Press, Massachusetts, USA.

Roth, Z.S., Mooring, B.W. and Ravani, B., 1987, “An Overview of Robot Calibration”, IEEE Journal of Robotics and Automation, RA-3, No. 3, pp. 377-85.

Schroer, K., 1993, “Theory of kinematic modeling and numerical procedures for robot calibration”, Robot Calibration, edited by R. Bernhardt and S. L. Albright, Chapman & Hall, London.

Schroer, K., Albright, S. L. and Grethlein, M., 1997, “Complete, Minimal and Model-Continuous Kinematic Models for Robot Calibration”, Robotics & Computer-Integrated Manufacturing, Vol. 13, No. 1, pp. 73-85.

Spong, M. W., Vidyasagar, M. (1989), "Robot Dynamics and Control", 1st ed., Ed. John Wiley & Sons, Singapore.

Veitscheggar, K. W. and Wu, C. W., (1986), "Robot Accuracy Analysis based on Kinematics". IEEE Journal of Robotics and Automation, Vol. 2, No. 3, pp. 171-179.

Zhuang, H., 1992, “A Complete and Parametrically Continuous Kinematic Model for Robot Manipulators”, IEEE Transactions on Robotics and Automation, Vol. 8, no. 4, pp. 451-63.