Advances in Robotics & Automation

González-Palacios et al., Adv Robot Autom 2016, 5:2
DOI: 10.4172/2168-9695.1000148

Research Article Open Access

The Generalized Architecture of the Spherical Serial Manipulator

González-Palacios MA*, Ortega-Alvarez CJ, Sandoval-Castillo JG, Cuevas-Ledesma SM and Mendoza-Patiño FJ

Division of Engineering Campus Irapuato-Salamanca, University of Guanajuato, Salamanca Carr - V of Santiago, Community Palo Blanco, Salamanca, Mexico

Abstract

It is well known that the inverse kinematics problem for the spherical serial manipulator has been solved in the past by diverse methods; but this problem for a generalized architecture based on the Denavit-Hartenberg parameters has not been treated. Therefore, this paper considers such treatment in a geometric analysis to derive a closed-form solution of the inverse kinematics problem, whose algorithm is validated by simulating pick and place operations. With the code implementation of a novel linear tracking algorithm introduced here, this application is accomplished in real time with the aid of a development software devoted to simulate robotic applications in real time, allowing the visualization of the performance of all possible architectures including the eight types defined in this paper; It is also shown that the Stanford arm is comprised within this classification. In order to demonstrate the great potential offered by combining the algorithms released here in simulations for industrial applications requiring a quick response, a case study is presented by taking as examples, the Stanford manipulator and a spherical manipulator with generalized architecture.

Keywords: Inverse kinematics problem; Industrial serial manipulators; Hartenberg-Denavit notation; Real time simulation; Linear path tracking; Robot architecture

Introduction

Some architectures of serial manipulators like the RRP have been studied in different ways, for instance, in [1] a RRP manipulator is studied to solve the inverse kinematics problem, IKP for short, near singularities applying the least-square method. Furthermore, by Saeidpourazar [2,3] the forward and inverse kinematics problems are analyzed in a RRP nanomanipulator called MM3A, which is able to perform specific tasks. It was also developed a controller which evaluates the vision and strength of the MM3A.

A numerical method that solves the IKP of six-degree-of-freedom manipulators with combinations of revolute, prismatic or cylindrical joints, is proposed in [4]. A similar study in [5] shows an algorithm for the solution of the inverse kinematics problem of six- or fewer degree-of-freedom manipulators, where the PUMA robot is analyzed as a case study, and being implemented in the KINEM package, which is a software that solves the inverse kinematics problem of several serial manipulators.

DYNAMAN [6], is another simulation software package developed in FORTRAN, where a method to formulate the dynamic equations is applied on serial robots with multiple links, with either revolute or prismatic joints. This software does not have an interface to modify the DH parameters in a simple way. Besides, the model generated lacks of details.

SYMORO [7] and OpenSYMORO [8] is an open source software package developed fundamentally in Pythom dedicated to the symbolic modeling of robots. This package provides support to robot models with flexible joints and to mobile robots. It offers support to serial robots with open and closed chains. A visualization tool to perceive the robot structure is also provided. Another software Package, RobSim [9], whose platform is MATLAB, is able tosimulate five manipulator types: spherical, cartesian, cylindrical, and two architectures of the articulated robot (KAU) PYR and RPR, where link dimensions and robot poses can be specified. It also displays the workspace in 2D.

Some references discussed the IKP for different architectures in the same configuration as in [10], which is obtained for a 5R manipulator using Clifford's exponential algebra, it is also presented in [11] a unified solution in closed form of a serial robot with six-degree-of-freedom by Pieper's geometry or the Duffy's geometry. On the other hand, Pieper [12], studies rigid body motions onsix-degree-of-freedom manipulators, with revolute or prismatic joints. FORTRAN is applied for the position analysis and trajectory generation. By means of using velocity methods and Newton-Rapshson technics, numeric solutions of the IKP of the general case are presented. Although Pieper's work is focused in the methodology required to solve any type of manipulator with six-degree-of-freedom, only presents the study case of the Stanford manipulator, where the equations to obtain the joint variables are shown.

Paul and Shimano [13], provide rules to assign coordinate frames to manipulator links, including simple cases and prismatic joints. In this work, the Stanford manipulator is presented as study case to prove their algorithm to solve position and orientation.

Khalil and Murareci [14] establish the possibility of considering special values of the geometric parameters of manipulators to take the characteristic polynomial to the simplest form using a minimum number of equations. This polynomial, for special architectures, can be obtained numerically. The RRP architecture is studied only when $\alpha_1 = n\pi$ and $\alpha_2 = n\pi$ (the first three axes are parallel), besides, an equation for each joint variable is not presented explicitly.

Much of the kinematic analysis presented in the literature provides numerical solution approaches, as in the case of [15], where a finite element method is applied to solve the forward kinematics

***Corresponding author:** González-Palacios MA, Division of Engineering Campus Irapuato-Salamanca, University of Guanajuato, Salamanca Carr - V of Santiago, Community Palo Blanco, Salamanca, Mexico, Tel: 2465-52 464 647 9940; E-mail: maxg@ugto.mx

Received April 08, 2016; **Accepted** April 27, 2016; **Published** April 30, 2016

Citation: González-Palacios MA, Ortega-Alvarez CJ, Sandoval-Castillo JG, Cuevas-Ledesma SM, Mendoza-Patiño FJ (2016) The Generalized Architecture of the Spherical Serial Manipulator. Adv Robot Autom 5: 148. doi:10.4172/2168-9695.1000148

problem of the Stanford manipulator using the software Maple V. A probabilistic approach, specifically the Monte Carlo method, is applied in [16] to model the kinematic and dynamic random errors of various manipulator parameters, such as those related to the Stanford manipulator. In [17] the equations of motion are developed for any manipulator with two - or three-degree-of-freedom, where the PUMA and the Stanford robots are presented as case studies.

An important subject is the validation of the results obtained in mathematical models with the application of computer packages for simulation. For example, in [18] are implemented in LabVIEW the forward kinematic and the dynamic analysis of a SCARA manipulator, whereas in [19], MATLAB is applied to solve the forward kinematics problem of a spherical manipulator by implementing a FPGA.

It was found, in general, that the specific solution of the IKP of the spherical manipulator is based on either the basic architecture or the architecture defining the Stanford arm, but to the best of our knowledge, the solution to a generalized architecture has not been reported explicitly in the literature. This generalized solution involves the formulation that embrace eight different possible architectures that the spherical manipulator can hold depending on the values of the first two twist angles according to the Denavit-Hartenberg notation [20].

The generalized closed-form solution of RRP serial manipulators presented in this paper, is an important contribution to the research and development field, since the resulting algorithm, represents a powerful tool to study in real time, the motion of all possible architecture combinations within the same framework.

In this research work, a section is dedicated to establish the guidelines of what we call the generalized architecture of the spherical manipulator. Then, we define the eight types of the generalized spherical manipulator, followed by the formulation of the inverse kinematics solution, which in most cases is represented as a closed-form solution. This formulation includes the pseudo-code that was applied as template to implement the code in ADEFID's platform [21]. We have dedicated a section to present an algorithm that generates intermediate poses by providing a single parameter, once the initial and final poses of a linear trajectory have been defined.

In the last section, we present a case study related to a simulation of pick and place operations. We integrate in the simulation, the algorithm for the inverse kinematics solution and the algorithm for the linear trajectory planning. In order to provide a closer taste of the performance, sets of frames are sequentially presented. The simulation is running in real time within the software developed for this application.

The Generalized Spherical Manipulator

The manipulator designed with spherical architecture for the positioning of the wrist center point, is the well-known Stanford arm, whose pioneer model is shown in Figure 1a, while Figure 1b shows a more up-to-date version.

The full architecture of the manipulators described above is identified as RRPRRR. Since the last three joints are connected with the architecture of a spherical wrist (RRR), we will refer to these type of manipulators by the first three joints only, namely, the RRP spherical manipulator.

Our study is focused in the derivation of the inverse kinematics solution for the positioning of the wrist center point of what we call the generalized architecture of the spherical manipulator, in other words, according to the definitions of the links parameters following the Denavit-Hartenberg's notation [20], DH notation for short, all link offsets and link lengths are not zero.

The skeleton representation of the basic architecture of the spherical manipulator along with the corresponding DH parameters, is shown in Figure 2, based on the values given in Table 1, where all link lengths a_i, as well as b_2 are zero. The architecture of the Stanford manipulator, shown in Figure 3, is obtained from the basic architecture of the spherical manipulator by setting b_2, the offset of the second link, not equal to zero, as shown in Table 2. Now, the generalized architecture, with the parameters given in Table 3, is plotted in Figure 4.

On the Classification

Applying the same concept introduced in [22] regarding the manipulators' architecture, we define here eight different types depending on α_1 and α_2 values. Thus, α_1 can be either 90° or 270°, if the manipulator is right shoulder (R) or left shoulder (L), respectively. Whereas α_2, which defines the direction of motion of the wrist center point while the manipulator is set at the zero configuration with $b_3 \neq 0$, can hold 0, 90°, 180° , and 270°. Table 4 shows the nomenclature defined for this classification, and Figures 5 and 6, the skeleton representation of right shoulder and left shoulder.

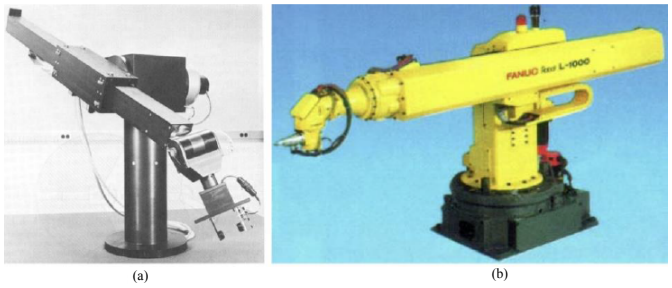

Figure 1: Stanford manipulator.

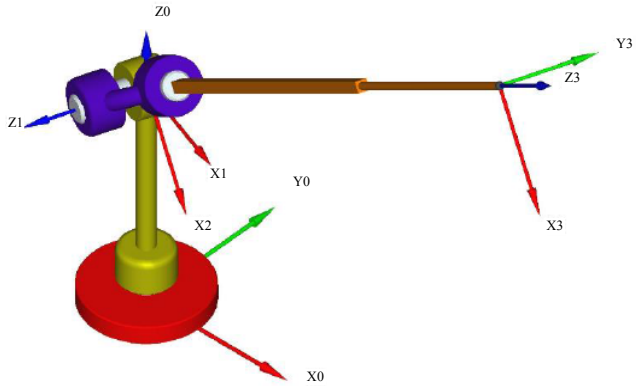

Figure 2: Skeleton representation of the spherical architecture.

Link	θ_i	b_i	a_i	α_i
1	θ_1^*	b_1	0	90°
2	θ_2^*	0	0	-90°
3	0	b_3^*	0	0
*Joint variables				

Table 1: DH parameters of the spherical architecture.

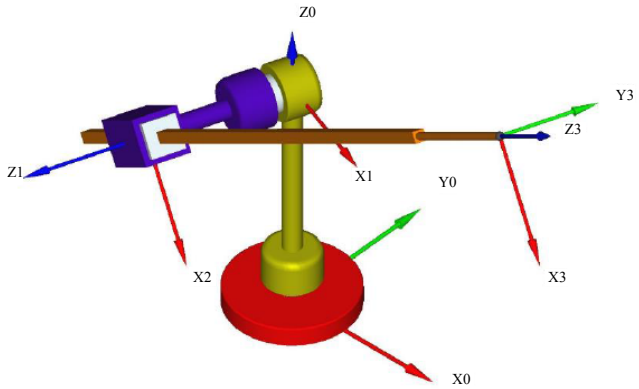

Figure 3: Skeleton representation of the Stanford architecture.

Link	θ_i	b_i	a_i	α_i
1	θ_1^*	b_1	0	90°
2	θ_2^*	b_2	0	-90°
3	0	b_3^*	0	0
*Joint variables				

Table 2: DH parameters of the stanford architecture.

Link	θ_i	b_i	a_i	α_i
1	θ_1^*	b_1	a_1	α_1
2	θ_2^*	b_2	a_2	α_2
3	θ_3	b_3^*	a_3	α_3
*Joint variables α_1=90°, 270° α_2=0,90°, 180°, 270°				

Table 3: DH parameters of the generalized spherical architecture.

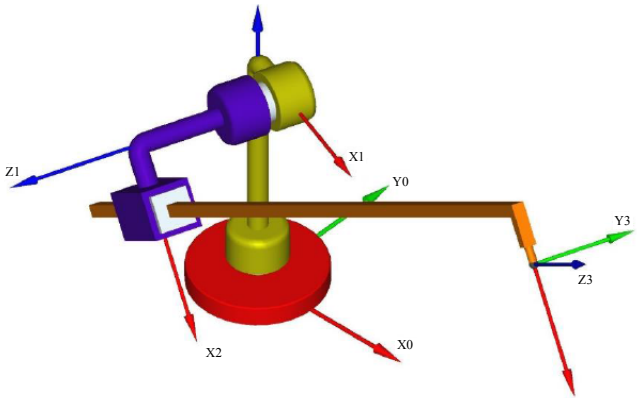

Figure 4: Skeleton representation of the generalized spherical architecture.

The Generalized Geometric Approach

In this section, we establish an algorithm that describes the inverse kinematics solution of the generalized spherical architecture RRP, with spherical wrist. The approach presented here, holds the eight architectures described in Section 3, nevertheless, we select the RD type to show the variables and to state their geometric relationship that will be applied along the formulation. Therefore, we depart from the fact that except for the joint variables, we know all the values given in Table 5. We also assume that the position of the wrist center point $p=(p_x, p_y, p_z)^T$, is known.

To obtain θ_1, the joint variable that orientates x_1 with respect to x_0, we use Fig. 7 as reference, where all the linear dimensions shown, lie in parallel planes to the x_0-y_0 plane. Thus, θ_1 is obtained as

$$\theta_1 = \phi + \beta \tag{1}$$

From eq. (1), ϕ is related with $r = \sqrt{p_x^2 + p_y^2}$ by,

$$\phi = \arctan(p_y/p_x) \tag{2}$$

while β is associated with h and r as,

$$\beta = \arctan(h/\sqrt{r^2 - h^2}) \tag{3}$$

with

$$h = b_2 + \lambda_2 b_3 \tag{4}$$

In eq. (4), $\lambda_2 = \cos\alpha_2$. Further down we will take into account that $\lambda_i \equiv \cos\alpha_i$ and $\mu_i \equiv \sin\alpha_i$ for i=1,2,3. It is important to note that the equations derived for the inverse kinematic solution are obtained for an arbitrary value of α_2, and the four values for the architectures defined here become particular cases Figure 7.

Now, to obtain θ_2, the joint variable that orientates x_2 with respect to x_1, we use Figure 8, in this case, all the linear dimensions shown, lie in parallel planes to the x_1 - z_0 plane. Thus, θ_2 is obtained as

Architecture	Description		a_1	a_2
RD	Right Shoulder	Down Wrist	90°	90°
RU		Up Wrist		270°
RE		External Wrist		0°
RI		Internal Wrist		180°
LD	Left Shoulder	Down Wrist	270°	270°
LU		Up Wrist		90°
LE		External Wrist		180°
LI		Internal Wrist		0°

Table 4: Classification according to the twist angles.

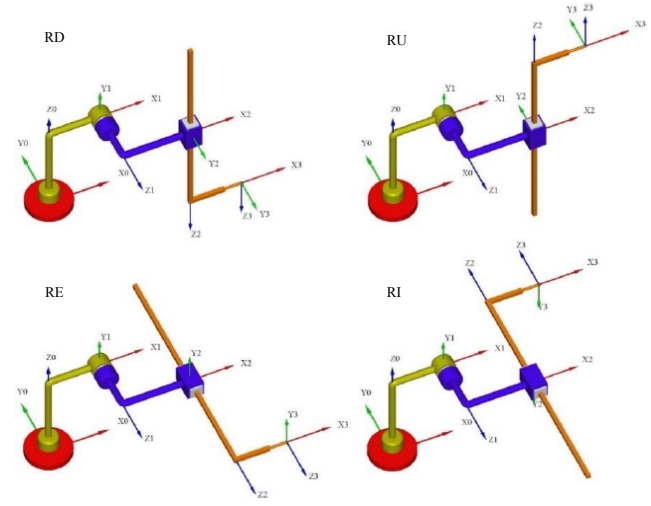

Figure 5: Right shoulder architectures.

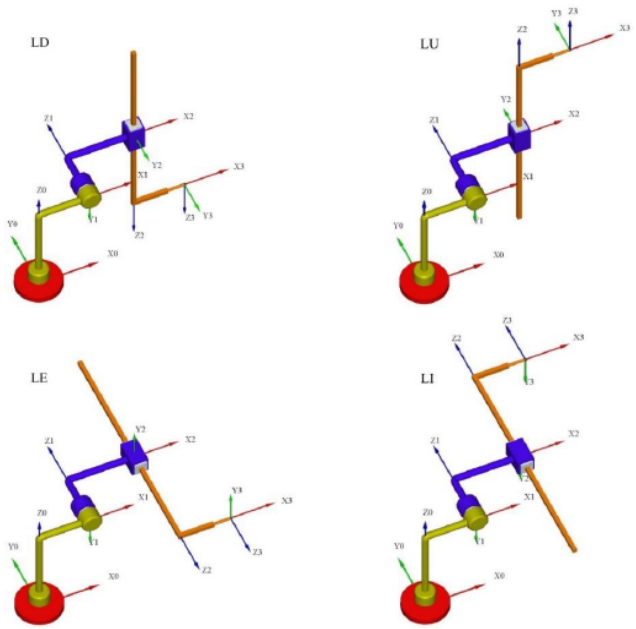

Figure 6: Left shoulder architectures.

θ_i	b_i	a_i	α_i
θ_1^*	b_1	a_1	90°
θ_2^*	b_2	a_2	α_2
θ_3	b_3^*	a_3	α_3
*Joint variables			

Table 5: DH table for the right shoulder architecture.

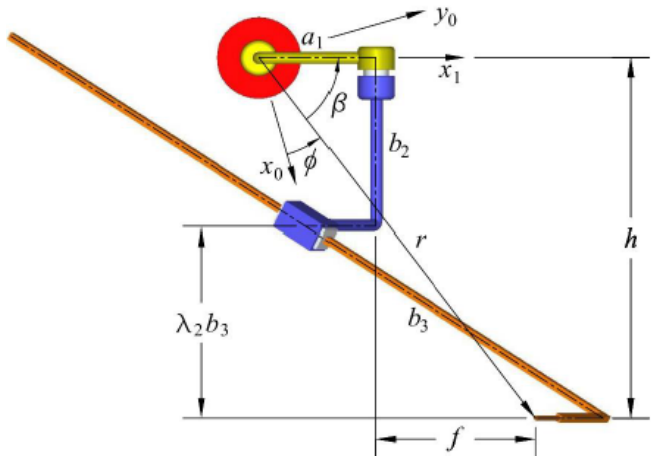

Figure 7: Plant view of a RD architecture.

$$\theta_2 = \gamma + \psi \tag{5}$$

In eq. (5), γ is associated with a_2, a_3 and a projection of b_3 as,

$$\gamma = \arctan(\mu_2 b_3/(a_2+a_3)) \tag{6}$$

While ψ is related with d and f by,

$$\psi = \arctan(d/f) \tag{7}$$

where,

$$d = p_z - b_1 \tag{8}$$

$$f = \sqrt{r^2 - h^2} - a_1 \tag{9}$$

Now, a projection of $b3$ is related to $a2$, $a3$ and e, namely,

$$\mu_2^2 b_3^2 = e^2 - (a_2 + a_3)^2 \tag{10}$$

with

$$e = \sqrt{d^2 + f^2} \tag{11}$$

To solve θ_1 and θ_2, from eqs. (1) and (5), respectively, we need ϕ, β, γ, and ψ. From eq. (2) we see that ϕ depends on the position of the wrist center point, which is given, but the other three angles, are functions of b_3, therefore, we must solve eq. (10) first, which for arbitrary values of α_2, must be evaluated numerically. However, if $a_1 = 0$, then b_3 is obtained in closed-form as,

$$b_3 = \sqrt{d^2 + r^2 - (\mu_2 b_2)^2 - (a_2 + a_3)^2} - \lambda_2 b_2 \tag{12}$$

A similar analysis can be performed to solve the inverse kinematics of the left shoulder manipulator. Now, for the eight types defined in Section 3, a closed-form solution for b_3 is also obtained. Taking into account that $b_3 > 0$, there are two possible solutions for a given architecture, and they depend on the position of the manipulator's body, which is represented by the first joint. The first set of solutions

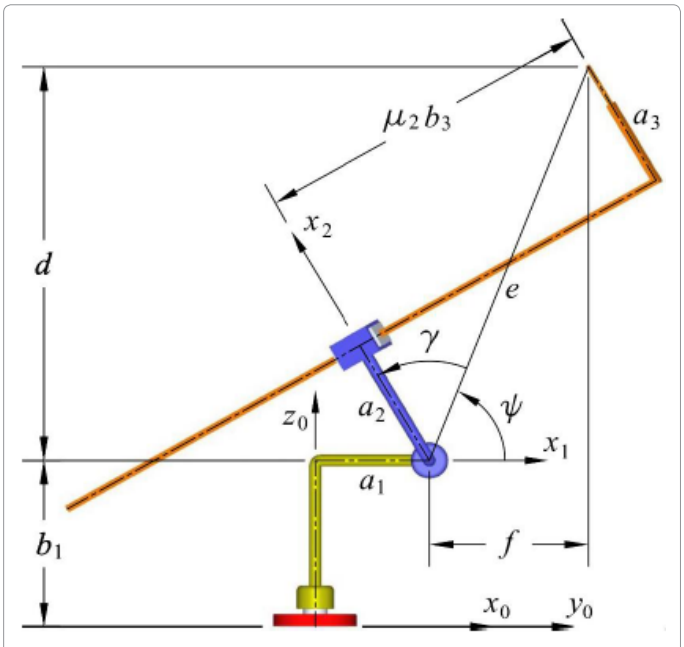

Figure 8: Side view of a RD architecture.

Arch.	b_3	θ_1	θ_2
RD	$\sqrt{d^2 + ([r^2 - b_2^2]^{1/2} - a_1)^2 - (a_2 + a_3)^2}$	$\phi+\beta$	$\gamma+\psi$
RU			
LD			
LU			
RE	$\sqrt{r^2 - ([(a_2 + a_3)^2 - d^2]^{1/2} + a_1)^2} - b_2$		
LE			
RI	$-\sqrt{r^2 - ([(a_2 + a_3)^2 - d^2]^{1/2} + a_1)^2} + b_2$		
LI			

Table 6: Joint variables according to the eight architectures. First body solution.

Arch.	b_3	θ_1	θ_2
RD	$\sqrt{d^2+([r^2-b_2^2]^{1/2}+a_1)^2-(a_2+a_3)^2}$	$(\phi-\beta)+\pi$	$(\gamma-\psi)-\pi$
RU			
LD			
LU			
RE	$\sqrt{r^2-([(a_2+a_3)^2-d^2]^{1/2}-a_1)^2}-b_2$		
LE			
RI	$-\sqrt{r^2-([(a_2+a_3)^2-d^2]^{1/2}-a_1)^2}+b_2$		
LI			

Table 7: Joint variables according to the eight architectures. Second body solution.

Link	θ_i (deg)	b_i (mm)	a_i (mm)	α_i (deg)
1	30*, 170.35**	300	100	90
2	110*, -137.31**	150	100	90
3	0	400*, 571.5**	75	0
*Body solution 1, **Body solution 2				

Table 8: DH parameters for the solutions of the RD architecture.

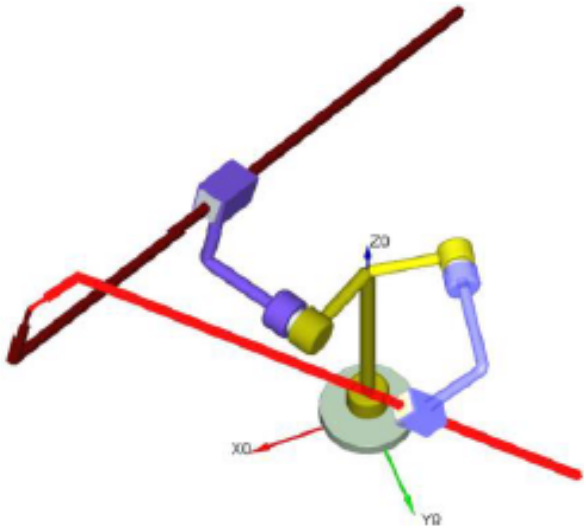

Figure 9: Representation of the two solutions of the RD architecture with $p = (435,78,601)^T$.

is listed in Table 6, while the second set, in Table 7. To the best of our knowledge, these sets have not been reported in the literature.

To illustrate the two solutions to reach a given wrist center point, an example of a RD manipulator is presented here. The coordinates of p are (435,78,601) mm, and the two solutions together with the design parameters are shown in Table 8. The simulation is achieved in ADEFID's platform [21], and Figure 9 illustrates the two poses of the manipulator.

With the closed-form solution for the inverse kinematics positioning approach, above introduced, we have derived a pseudo-code that consider all possible solutions in which b_3 can be evaluated with no need of a numerical solution. Furthermore, α_1 and α_2 are symbolically expressed so that not only the eight architecures are contemplated but also those architectures with an arbitrary value of α_2 if $a_1 = 0$. Such pseudo-code is listed next:

```
beginfunSetInversePosition(DH,p)
        r² ← p_x² + p_y²
        d ← (p_z − b_1)μ_1
        φ ← arctan(p_y/p_x)
        ifa_1 == 0
                κ ← d² + r² − μ_2²b_2² − (a_2 + a_3)²
                ifκ<0  return FALSE
                b_3 ← √κ − λ_2b_2
        else   if ABS(μ_2) < 0                  //If  α_2 = 0° or 180°
                if IsBodySol1 == TRUE
                        κ ← r² − (√((a_2 + a_3)² − d²) + a_1)²
                else
                        κ ← r² − (√((a_2 + a_3)² − d²) − a_1)²
                ifκ<0  return FALSE
                b_3 ← (√κ − b_2)λ_2
        else if ABS(μ_2) == 1           //If α_2 = 90° or −90°
                if IsBodySol1 == TRUE
                        κ ← d² + (√(r² − b_2²) − a_1)² − (a_2 + a_3)²
                else
                        κ ← d² + (√(r² − b_2²) + a_1)² − (a_2 + a_3)²
                ifκ<0  return FALSE
                b_3 ← √κ
        h ← (b_2 + λ_2b_3)μ_1
        β ← arctan(h/√(r² − h²))
        ifIsBodySol1 == TRUE
                f ← √(r² − h²) − a_1
        else
                f ← √(r² − h²) + a_1
        ψ ← arctan(d/f)
        γ ← arctan(μ_2b_3/(a_2 + a_3))
        ifIsBodySol1 == TRUE
                θ_1 ← φ + β
                θ_2 ← γ + ψ
        else
                θ_1 ← (φ − β) + π
                θ_2 ← (γ − ψ) − π
        return TRUE
endfun
```

Note that two arguments should be passed when Set Inverse Position ( ) is called, namely,

- The DH parameters of the manipulator through a structure defined as DH, which stores the values θ_i, b_i, a_i, and αi, with I=1,2,…,6.
- The desired position vector p of the wrist center point.
- Now, the pseudo-code for the full inverse kinematics solution is readily defined as:

```
beginfunSetFullInverse(DH, Q, d)
        p ← d − Q[a_6, 0, b_6]^T
        θ_1, θ_2, b_3 ←SetInversePosition(DH, p)
        θ_4, θ_5, θ_6 ←See [15] for the inverse
        orientationsolution
endfun
```

In this case, besides the argument DH, there are other two arguments to pass when calling Set Full Inverse ( ):

- The desired orientation of the end effector given by Q.
- The desired position vector of the end-effector reference point d.

Linear Path Tracking

In order to better understand the case study presented in the following section, we report here an algorithm devoted to generate discreet poses given by a single parameter and departing from the knowledge of the initial and final poses of a linear trajectory. Any

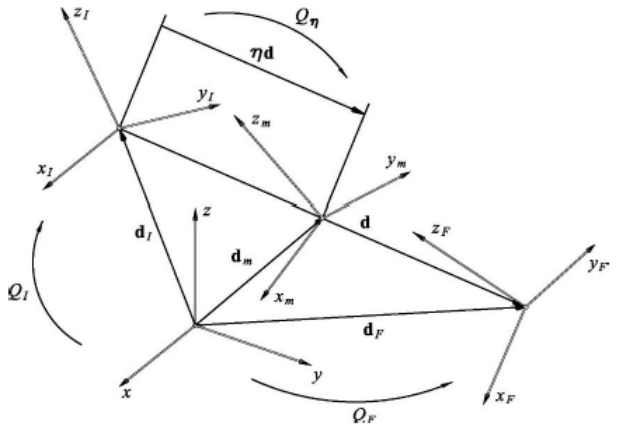

Figure 10: Linear path tracking by defining initial and final poses.

desired pose of the end effector is represented by its coordinate frame associated with a homogeneous matrix T containing the matrix Q and the vector d, which respectively represent the frame rotation and the position vector with respect to a given reference coordinate frame, as indicated in eq.(13).

$$T = \begin{bmatrix} Q & d \\ 0 & 1 \end{bmatrix} \tag{13}$$

Once the initial and final frames, $\mathcal{S}_I$ and $\mathcal{S}_F$, of the corresponding poses are defined by T_I and T_F, respectively, intermediate poses in a line path, represented by T_m, can be readily obtained with the aid of a single parameter η, with $0 \le \eta \le 1$. To achieve this, it is necessary to define some invariants, such as vector d, defining the position vector of the origin of frame $\mathcal{S}_F$ with respect to frame $\mathcal{S}_I$, and the roll, pitch and yaw angles, which we denote as φ_x,φ_y and φ_z. These angles define the orientation of frame $\mathcal{S}_F$ with respect to frame $\mathcal{S}_I$ by performing basic rotations about x_I,y_I and z_I, represented as $Q_x(\varphi_x)$, $Q_y(\varphi_y)$, and $Q_z(\varphi_z)$, respectively. In Figure 10 we can observe the relationship between the mentioned poses. The function that evaluates the invariants is named Set Invariants ( ), and the pseudo-code is presented below:

```
beginfun SetInvariants(T_I, T_F)
        Q_I ← GetRotMatFromPose(T_I)
        Q_F ← GetRotMatFromPose(T_F)
        d_I ← GetVecFromPose(T_I)
        d_F ← GetVecFromPose(T_F)
        φ_x, φ_y, φ_z ← GetRPYAngles(Q_I^T Q_F)
        d ← d_F − d_I
endfun
```

Now, with the invariants defined with Set Invariants ( ), it is possible to obtain any intermediate pose by calling the function named Get Pose ( ), whose pseudo-code is given as:

```
beginfun GetPose(η)
        Q_m ← Q_I Q_z(ηφ_z) Q_y(ηφ_y) Q_x(ηφ_x)
        d_m ← d_I + ηd
        T_m ← [Q_m  d_m; 0  1]
        returnT_m
endfun
```

Functions Get Rot Mat From Pose ( ) and Get Vec From Pose ( ) take care of retrieving respectively, Q and $\mathbf{d}$ from a given T matrix. Whereas Get RPY Angles ( ), returns roll, pitch and yaw angles by solving the following equation:

$$Q_I^T Q_F = Q_Z(\varphi_Z) Q_y(\varphi_y) Q_x(\varphi_x) \tag{14}$$

Case Study

The results obtained in this paper were validated through an on-line simulation developed in ADRS (Architecture Design and Robot Simulation) which is an application developed in ADEFID platform. The simulation consists in the execution of a pick and place operation. The full cycle of this task is composed on a set of control poses along the planned trajectory. In this case it takes four poses besides the home pose to complete the task.

In order to have a better picture of the great advantage of applying the linear tracking approach introduced here in combination with the closed-form solution of the IKP, we describe the algorithm that takes the robot smoothly from the initial pose to final pose of the linear trajectory.

To begin with, we distinguish two main control levels in the simulation program. The first level consisting in the invariants definition of a desired trajectory, and the second level, in the execution of the process to move from one control point to another.

Within a complete cycle, the robot is set in different states. For each state the robot moves from the beginning to the end of the linear trajectory. When the robot reaches the end pose of a given state, it is set as SUSPEND state, meaning that the robot waits for the next control step, and a control step might be triggered by an input signal, either virtual or real.

Let us consider that the action to move from pose A to pose B is defined by the state STATE1, and the input signal is the Boolean variable named Task from A to B. Then the pseudo-code to be performed in the first level, which is part of an infinite loop, is defined as follows:

```
...
if SUSPEND ==RobotState&& TRUE == TaskFromAtoB
        SetInvariants(T_A, T_B)
        Timer.Init(Period_AB)
        SetState(STATE1)
...
```

Note that at this control level, the statements are executed only once. Now, for the second level, a switch is applied to consider all the

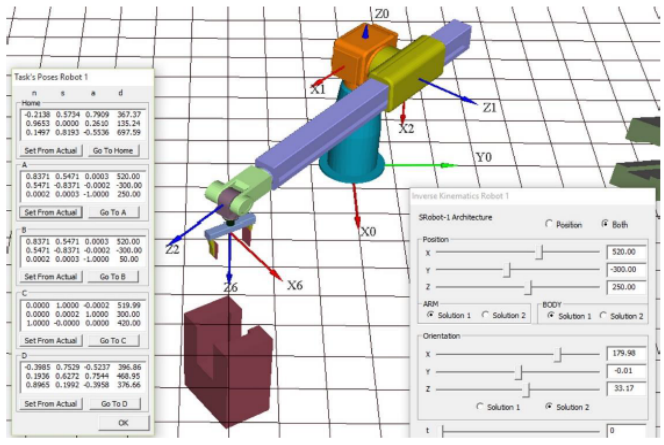

Figure 11: Setting the poses of the pick and place simulation.

θ_i (deg)	b_i (mm)	a_i (mm)	α_i (deg)
θ_1	400	0	-90
θ_2	120	0	90
0	b_3	0	0

Table 9: DH table, stanford manipulator (LU-manipulator).

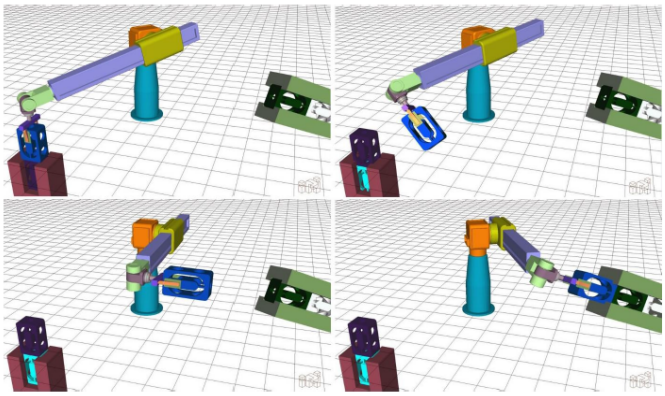

Figure 12: Stanford manipulator performing a pick and place task. First body solution.

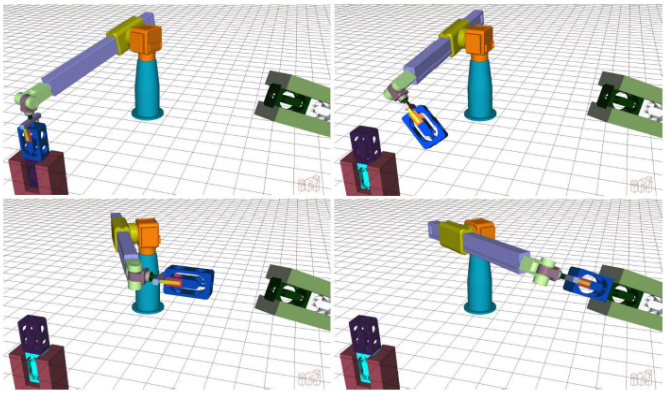

Figure 13: Stanford manipulator performing a pick and place task. Second body solution.

cases established for the simulation. In the pseudo-code presented below, the case STATE1 will be accessed several times until the state is changed to SUSPEND:

```
…
beginswitch(nState)
        …
        begincase STATE1
                if FALSE ==Timer.bIsExpired()
                        if TRUE == GetFrameParameter(Timer, &η)
                                Q_m, d_m ← T_m ← GetPose(η)
                                SetFullInverse(DH, Q_m, d_m)
                else
                        SetSate(SUSPEND)
        endcase
        …
endswitch
…
```

Setting the control poses that the manipulator must follow during the simulation execution becomes a simple task due to the visualization aids the program provides to the user, in other words, a dialog with tools like sliding bars helps to evaluate in-line the inverse kinematics. While interacting with the sliding bars the manipulator moves accordingly. Now with the aid of another dialog, the user can set a given position as one of the five control poses mentioned above. Figure 11, displays an image in which the two dialogs are opened together with a sample of a Stanford robot. Note that in the image, Pose A was set by just pressing the button labeled Set From Actual.

Now, we applied the algorithm to a simple case, i.e., the Stanford manipulator, which is set as a LU -manipulator. On the ADEFID platform we simulate it with the parameters shown in Table 9.

We have already indicated that for a given architecture there are two possible solutions with $b_3 > 0$, then we present still images of the simulation while performing the task. In Figure 12, four frames show the sequence of the simulation, starting from the top left corner, where the block is picked from the feeding conveyor, and ending to the right bottom corner where the block is delivered. The same simulation is performed in Figure 13, but in this case, with the second solution. These solutions are chosen by setting the variable IsBodySol1 true or false in the Inverse Position ( ) function.

The capacity to import STL files of models created in any CAD software having the feature to export this type of files, is an ADEFID's advantage that allows to create more realistic simulations. Such task,

θ_i (deg)	b_i (mm)	a_i (mm)	α_i (deg)
θ_1	270	153	90
θ_2	300	182	90
0	b_3	80	0

Table 10: DH table of a RD manipulator with generalized architecture.

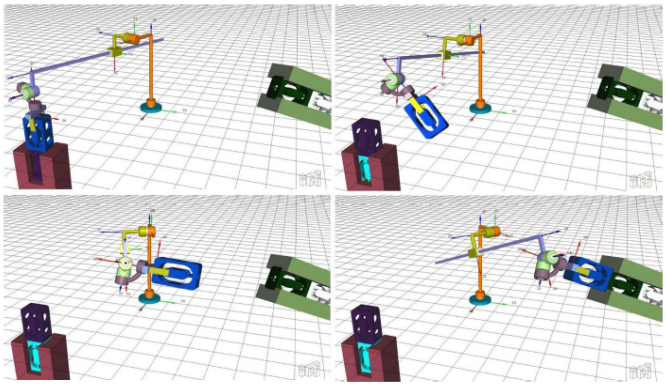

Figure 14: A generalized spherical architecture. First body solution.

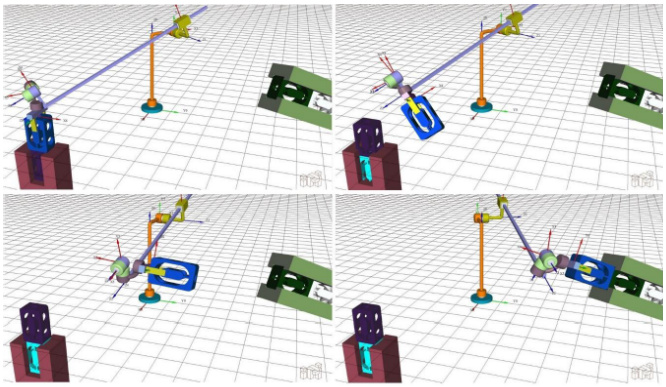

Figure 15: A generalized spherical architecture. Second body solution.

is achieved through the application of the DrawSTL ( ) function pertaining to the CSTL Draw class. With this function, an OpenGL list for each link of a robot is generated, and all lists are handled to achieve simulations as those shown in Figures 11-13.

As a second example we test the algorithm with the generalized spherical architecture. The DH design parameters applied in this case are given in Table 10. The first and second solutions are shown in Figures 14 and 15, respectively. Considering that it is a generalized architecture, and there is no a specific design of the links, the manipulator is shown in the skeleton form.

Conclusions

The unified orthogonal architecture concept presented in [22] for serial 6R manipulators, motivated the idea to extend the concept on spherical serial RRPRRR manipulators, and we were able to derive a formulation to solve the inverse kinematic problem for a generalized architecture, in other words, for cases in which all the Denavit-Hartenberg parameters involved for the positioning problem are not zero.

With the twist angles combination of the two first joints of the generalized architecture of the spherical serial manipulator, eight different types were defined, and it was shown that the Stanford arm belongs to one of these. It was also found that for each type, two possible solutions exist for the inverse kinematics problem on the positioning of the wrist center point.

Although the general solution reported here for the inverse kinematics problem is numerical, we obtained a closed-form solution for the case in which α_1=0 with an arbitrary value of α_2. Besides, closed-form solutions were also obtained for the eight architectures defined in this paper.

We were able to handle all closed-form solutions with the approach of a single algorithm. An original feature of this algorithm is that the user can control the solutions allowing to choose the one that best suits the application requirements. A novel algorithm to follow a linear path of the end effector was also introduced in this paper, knowing its initial and final poses, by controlling the position and orientation along the path with a single parameter. The corresponding pseudo-codes of both algorithms were developed, implemented, and validated within a case study focused in pick and place operations, with the aid of ADEFID through ADRS (Architecture Design and Robot Simulation), which is an open source application dedicated to design and simulate in real time, serial manipulators for industrial and research purposes.

Acknowledgement

The first author acknowledges the support from SNI, (Sistema Nacional de Investigadores), México. The second to the fifth authors acknowledge the support from CONACYT (Consejo Nacional de Ciencia y Tecnología), México.

References

1. Kircanski MV (1993) Inverse kinematic problem near singularities for simple manipulators: symbolical damped least-squares solution. Robotics and Automation Proceedings.
2. Saeidpourazar R, Jalili N (2008) Nano-robotic manipulation using a RRP nanomanipulator: Part A – Mathematical modeling and development of a robust adaptive driving mechanism. Applied Mathematics and Computation 206: 618-627.
3. Saeidpourazar R, Jalili N (2008) Nano-robotic manipulation using a RRP nanomanipulator: Part B – Robust control of manipulator's tip using fused visual servoing and force sensor feedbacks. Applied Mathematics and Computation 206: 628-642.
4. Raghavan M, Roth B (1993) Inverse kinematics of the general 6R manipulator and related linkages. ASME 115: 502-508.
5. Manocha D, Zhu Y (1994) A fast algorithm and system for inverse kinematics of general serial manipulators. Proceedings of IEEE Conference on Robotics and Automation 4: 3348-3353.
6. Sreenath N, Krishnaprasad PS (1986) DYNAMAN: A tool for manipulator design and analysis. Robotics and automation proceedings.
7. Khalil W, Creusot D (1997) SYMORO+: A system for the symbolic modelling of robots. Robotica 15: 153-161.
8. Khalil W, Vijayalingam A, Khomutenko B, Mukhanov I, Lemoine P, et al. (2014) Open SYMORO: An open-source software package for symbolic modelling of robots. International Conference on Advanced Intelligent Mechatronics.
9. Balamesh AS, Almatrafi TD, Aljawi AN, Akyurt M (2002) RobSim - A simulator for robotic motion. The 6th Saudi Engineering Conference, KFUPM, Dhahran.
10. Gracia PA, McCarthy JM (2006) Kinematic synthesis of spatial serial chains using clifford algebra exponentials. Journal of Mechanical Engineering Science 220: 1-16.
11. Xijian H, Yiwei L, Li J, Hong L (2015) VRM: A unified framework for closed-form solutions of a special class of serial manipulators. Int. J. Advanced Robotic Systems 12: 1-16.
12. Peiper DL (1968) The kinematics of manipulators under computer control. Stanford Artificial Inteligence Project.
13. Paul RP, Shimano B (1978) Kinematic control equations for simple manipulators. Decision and Control including the 17th Symposium on Adaptive Processes, IEEE Conference.
14. Khalil W, Murareci D (1994) On the general solution of the inverse kinematics of six-degrees-of-freedom manipulators. Advances in Robot Kinematics and Computationed Geometry pp: 309-318.
15. Torby BJ, Kimura II (1999) Dynamic modeling of a flexible manipulator with prismatic links. J Dyn Sys Meas Control 121: 691-696.
16. Rao SS, Bhatti PK (2001) Probabilistic approach to manipulator kinematics and dynamics. Reliability Engineering & System Safety 72: 47-58.
17. Ravishankar AS, Ashitava G (1999) Nonlinear dynamics and chaotic motions in feedback-controlled two- and three-degree-of-freedom robots. Indian Inst of Science pp: 1-16.
18. Kaleli A, Dumlu A, Corapsız MF, Erentürk K (2013) Detailed analysis of SCARA-type serial manipulator on a moving base with LabVIEW. Int J Advanced Robotic Systems 10: 1-10.
19. Sanchez DF, Muñoz DM, Llanos CH, Motta JM (2009) FPGA implementation for direct kinematics of a spherical robot manipulator. Reconfigurable Computing and FPGAs.
20. Denavit J, Hartenberg RS (1955) A kinematic notation for lower pair mechanisms based on matrices. Journal of Mechanical Design 77: 215-21.
21. González-Palacios MA(2012) Advanced engineering platform for industrial development. The Journal of Applied Research and Technology 10: 309-326.
22. González-Palacios MA (2013) The unified orthogonal architecture of industrial serial manipulators. J Robotics and Computer-Integrated Manufacturing 29: 257-271.