GEOMETRIC APPROACH TO SOLVING THE INVERSE DISPLACEMENT PROBLEM OF CALIBRATED DECOUPLED 6R SERIAL ROBOTS

Albert Nubiola and Ilian A. Bonev
Department of Automated Manufacturing Engineering, École de Technologie Supérieure, Montreal, QC, Canada
E-mail: ilian.bonev@etsmtl.ca

Received March 2013, Accepted January 2014
No. 13-CSME-118, E.I.C. Accession 3576

ABSTRACT
This paper presents a simple but efficient way to numerically calculate the inverse displacement problem of calibrated decoupled 6R serial robots. The method is iterative and works with any type of calibrated robot model, such as level-3 models, since it requires no algebraic computation and no resolution of high-order polynomials, only the computation of the forward displacement problem of the calibrated robot model and the inverse kinematics of the nominal robot model. The method proposed can find up to eight possible solutions for a given end-effector pose. A numerical example is presented, with one million arbitrary end-effector poses of a level-3 calibrated ABB IRB 120 robot. The computation time for solving the inverse problem is analyzed, and in most cases is found to be only four times the time needed to calculate the nominal inverse kinematics and the calibrated direct kinematics. Furthermore, the method is fast enough to be implemented directly into the robot controller using the RAPID programming language.

Keywords: inverse kinematics; decoupled serial robot; robot calibration.

APPROCHE GÉOMÉTRIQUE POUR RÉSOUDRE LA CINÉMATIQUE INVERSE DES ROBOTS SÉRIELS 6R DECOUPLÉS

RÉSUMÉ
Cet article présente une méthode simple et efficace pour calculer la cinématique inverse des robots sériels 6R découplés. La méthode est itérative et fonctionne pour tous les modèles théoriques de robots étalonnés (par exemple, des modèles du niveau 3), étant donné qu'aucun calcul algébrique ou résolution des polynômes d'ordre élevé n'est nécessaire. Ce qui est nécessaire c'est uniquement le calcul de la cinématique directe du robot étalonné et la résolution de la cinématique inverse du modle nominal du robot. La méthode proposée peut trouver jusqu'à huit solutions possibles pour une pose de l'effecteur. Un exemple numérique est montré incluant un million de poses arbitraires pour l'effecteur d'un robot ABB IRB 120 étalonné (étalonnage du niveau 3). Le temps de calcul nécessaire pour résoudre la cinématique inverse est analysé ; dans la plupart des cas on trouve qu'il est quatre fois le temps nécessaire pour calculer la cinématique inverse nominale et la cinématique directe du modèle étalonné. En plus, cette méthode est assez rapide pour être implémentée dans le contrôleur du robot en utilisant le langage de programmation RAPID.

Mots-clés : cinématique inverse ; robot sériel découplé ; étalonnage de robots.

1. INTRODUCTION

Solving the inverse kinematics of a decoupled 6R serial robot is relatively straightforward, and is generally achieved using analytical methods [1, 2] (closed-form solution). However, solving the inverse displacement problem (IDP) of such a robot when calibrated with a so-called level-2 model (only geometric errors considered) or, even worse, a level-3 model (where non-geometric errors are also considered), is much more complicated, and requires numerical methods [3–6].

It is well known that up to sixteen solutions can be found by solving the IDP of a general 6R serial robot. However, such a complete solution requires substantial computing time [7, 8] and is not suitable for real-time computation. As a result, researchers usually prefer to use practical numerical approaches [9]. Ways to efficiently calculate the IDP have improved considerably since the first attempts to solve the IDP [10]; however, the most recent works [11, 12] still require the solution of a univariate polynomial of degree 16.

An example of level-2 calibration is given in [13], where a camera is mounted on the robot to perform a closed-loop calibration. Unfortunately, the authors do not mention how they solve the IDP. It becomes even more complicated for level-3 models, due to the presence of non geometric parameters, such as compliance coefficients or thermal expansion coefficients. Depending on the complexity of such a model, a numerical method might often be the only way of solving the inverse problem.

Numerical methods can be divided into three types [5]: (1) Newton–Raphson methods, (2) predictor-corrector type algorithms, and (3) optimization techniques using the formulation of a scalar cost function. Examples of the first type can be found in [14], where a modified Newton-Gauss method is used. These methods involve the computation of the Jacobian matrix. A comparison between Newton–Raphson methods and predictor-corrector-type algorithms is provided in [15], where the authors conclude that the latter are faster.

An example of a predictor-corrector algorithm is given in [16]. This type of method often requires a large number of iterations, and does not always converge. Some of these algorithms also require computation of the Jacobian matrix, like the algorithm proposed in [17], which combines a predictor-corrector type of algorithm with a least-squares optimization technique, or the closed-loop method [6] where the inverse kinematic problem is solved as a control problem for a simple dynamic system. However, optimization techniques are usually complex, and several iterations are needed to achieve a solution.

Two approaches for level-2 calibrations are proposed in [4] using the nominal inverse kinematics. However, the solution becomes complex when the number of error parameters increases. The resolution of a nonlinear programming problem is divided into two phases in [18] for greater efficiency.

Finally, [5] greatly reduced the amount of computing time needed to solve the inverse problem, by calculating a so-called pose shift by iteratively adding and subtracting computed poses based on a truncated-series expansion of the desired pose. However, we realized that their method could be further improved by using pose multiplications based on a geometric principle, instead of adding and subtracting the poses of the truncated-series expansion of the desired pose.

This paper presents a new type of numerical solution: a geometrical and iterative method which incrementally approaches the final solution at each iteration. Like the method proposed in [5], our method does not need computation of the Jacobian matrix or derivatives of any kind. From the sixteen possible solutions to the IDP of the calibrated robot, up to eight can be obtained, and the new method only needs the forward displacement model of the calibrated robot model and the nominal inverse kinematics solution of the nominal robot model.

Although our iterative method works with any decoupled 6R serial robot and any calibration model, our numerical example is based on an ABB IRB 120 robot calibrated using a typical level-3 model. These models are presented in Section 2. The method proposed for solving the inverse problem is then detailed in

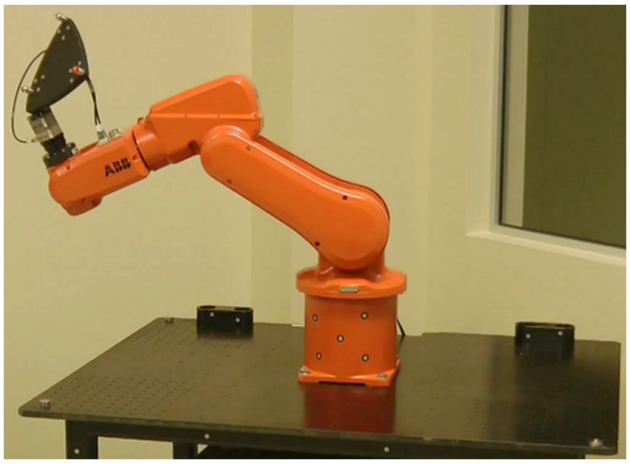

Fig. 1. One of the identification configurations used for calibrating the ABB IRB 120 industrial robot.

Section 3. Section 4 presents extensive analyses of the performance of our method, as well as one of those proposed in [5], for one million end-effector poses. Conclusions are presented in Section 5.

2. ROBOT MODEL

An ABB IRB 120 robot is used as an example in the validation of our method (Fig. 1). This robot does not have the Absolute Accuracy option (the robot calibration provided by ABB), and was calibrated by us using a Faro laser tracker (not shown in Fig. 1) and a special triangular end-effector made of steel, with three 0.5 in. spherically-mounted reflectors (SMRs) [19]. The maximum position error at any of the three SMRs when using the nominal kinematics is nearly 5 mm. After calibration, that error is reduced to less than 1 mm.

The calibration model consists of 31 error parameters: 6 parameters to locate the robot base frame, 20 geometric parameters modeling the robot kinematics (4 Denavit–Hartenberg modified parameters for links 2 to 6) and 5 compliance parameters (for joints 2, 3, 4, 5, and 6).

2.1. Stiffness Model

A five-parameter model is used to represent robot elasticity. This model takes into account the elasticity of the gearboxes of joints 2, 3, 4, 5, and 6 (one parameter per joint). The elasticity in each gearbox is modeled as a linear torsional spring (about the joint axis), so this parameter represents the effective constant compliance c_i of each joint i. Traction and compression effects are neglected, as are the torsional effects perpendicular to the joint axis. We did not consider a compliance parameter to model the flexibility of joint 1 because the axis of this joint is always vertical.

Torques are calculated according to link and tool mass, which apply a torque to each motor due to the gravity effect (only static errors are modeled, and no other external force is applied). The deformations are supposed to be small enough so that we can apply the superposition principle and neglect beam foreshortening. ABB provided us with the weight and center of gravity of each link for the ABB IRB 120 robot. The center of gravity of the end-effector is obtained from its CAD model. Figure 2 represents an arbitrary configuration of the robot arm for joints 2 and 3. This configuration can be considered as two generic links (consecutive or non consecutive). Although the 3D space is taken into account, the figure simplifies the representation in a vertical plane.

Torques are calculated recursively from joint 6 up to joint 1, so, for example, joint 5 is affected by link 5, link 6, and the tool, joint 4 is affected by links 4, 5, 6, and the tool, and so on. Actually, the tool mass and

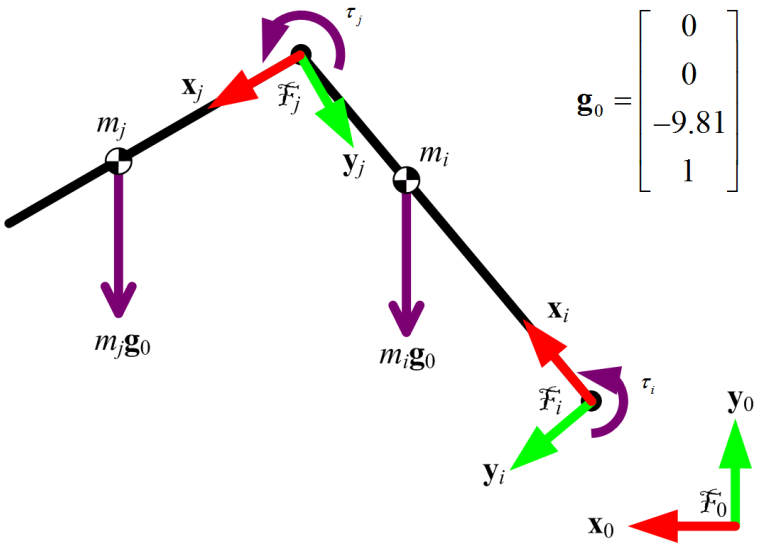

Fig. 2. Schematics of two consecutive links under the gravity effect.

its center of gravity can be combined with link 6, as they do not move with respect to one another, and so this recursive method already includes the tool.

The gravity vector seen by joint i corresponds to

$$\mathbf{g}_i = (\mathbf{H}_i^0)^{-1}\mathbf{g}_0, \tag{1}$$

where

$$\mathbf{H}_b^a = \mathbf{A}_{a+1}^a \ \ \mathbf{A}_{a+2}^{a+1} \ \ \ldots \ \ \mathbf{A}_b^{b-1}, \tag{2}$$

if $b > a$. Then, the gravity force of link j seen by link i can be found as

$$\mathbf{f}_j^i = m_j\mathbf{g}_i = m_j(\mathbf{H}_i^0)^{-1}\mathbf{g}_0 = [f_x^i, f_y^i, f_z^i, 1]. \tag{3}$$

We can also find the center of gravity of link j with respect to the coordinates of frame i, $\mathbf{c}_j^i$ if we know the center of gravity of link j with respect to the local coordinates, $\mathbf{c}_j$

$$\mathbf{c}_j^i = \mathbf{H}_j^i\mathbf{c}_j = [c_{x,j}^i, c_{y,j}^i, c_{z,j}^i, 1]^T, \tag{4}$$

where $\mathbf{H}_j^i$ is the identity matrix if $i = j$.

Once we have calculated all the combinations where $i \leq j$, we can calculate the torque supported by each joint

$$\tau_i = \tau_{i,\text{link}i} + \tau_{i,\text{link},i+1} + \cdots + \tau_{i,\text{link}6}, \tag{5}$$

where $\tau_{i,\text{link}j}$ is the contribution of the torque supported by joint i caused by link j

$$\tau_{i,\text{link}j} = f_{y,j}^i c_{x,j}^i - f_{x,j}^i c_{y,j}^i \ . \tag{6}$$

In our case, we took into account the gravity forces due to the masses of links 2, 3, 4, and the tool. We neglected the masses of links 5 and 6, as they are relatively small.

Table 1. Pose of the base frame with respect to the world frame with six error parameters (parameters α, β, and γ are the Euler angles according to the XYZ convention).

x	y	z	α	β	γ
$x_w + \delta x_w$	$y_w + \delta y_w$	$z_w + \delta z_w$	$\alpha_w + \alpha x_w$	$\beta_w + \delta\beta_w$	$\gamma_w + \delta\gamma_w$

Table 2. Complete DHM robot model with 25 error parameters.

i	$\alpha_i(^\circ)$	a_i (mm)	$\theta_i(^\circ)$	d_i (mm)
1	0	0	θ_1	290
2	$-90 + \delta a_2$	δa_2	$\theta_2 - 90 + \delta\theta_2 + c_2\tau_2$	δd_2
3	δa_3	$270 + \delta a_3$	$\theta_3 + \delta\theta_3 + c_3\tau_3$	δd_3
4	$-90 + \delta\alpha_4$	$70 + \delta a_4$	$\theta_4 + \delta\theta_4 + c_4\tau_4$	$302 + \delta d_4$
5	$90 + \delta\alpha_5$	δa_5	$\theta_5 + \delta\theta_5 + c_5\tau_5$	δd_5
6	$-90 + \delta\alpha_6$	δa_6	$\theta_6 + 180 + \delta\theta_6 + c_6\tau_6$	$72 + \delta d_6$

2.2. Complete Robot Model

Our level-3 robot model corresponds to a complete kinematic calibration of the robot, including the base and the end-effector, and five parameters related to the stiffness of the gearboxes of joints 2, 3, 4, 5, and 6. Tables 1 and 2 summarize all 31 error parameters. The DHM (Denavit–Hartenberg Modified) model used is the one described in [1], which can be represented as

$$\mathbf{T}_{A6} = \mathbf{A}_0^{\text{world}}\ \mathbf{A}_1^0\ \mathbf{A}_{A,2}^1\ \mathbf{A}_{A,3}^2\ \mathbf{A}_{A,4}^3\ \mathbf{A}_{A,5}^4\ \mathbf{A}_{A,6}^5, \tag{7}$$

where

$$\mathbf{A}_0^{\text{world}}\ \mathbf{A}_1^0 = \text{Trans}(x_w, y_w, d_1 + z_w)\text{Rot}(\mathbf{x}, \alpha_w)\text{Rot}(\mathbf{y}, \beta_w)\text{Rot}(\mathbf{z}, \gamma_w + \theta_1) \tag{8}$$

and

$$\mathbf{A}_{A,i}^{i-1} = \text{Rot}(\mathbf{x}, \alpha_i)\text{Trans}(a_i, 0, 0)\text{Rot}(\mathbf{z}, \theta_i)\text{Trans}(0, 0, d_i) \tag{9}$$

for $i \geq 2$, constitute the homogeneous transformation matrix representing the pose of reference frame Φ_i with respect to reference frame $\Phi_{i-1}, \mathbf{A}_0^{\text{world}}$ is the transformation matrix representing the pose of the *base frame* Φ_0 with respect to the *world frame* Φ_{world}, and $\mathbf{A}_1^0$ is the transformation matrix representing the pose of reference frame Φ_1 with respect to the base frame.

3. INVERSE DISPLACEMENT PROBLEM

The proposed iterative algorithm is a geometrical approach that uses a shifted pose error. Briefly, this pose error is the difference between a desired pose, $\mathbf{H}_d$, and the approximated pose, $\mathbf{H}_{\text{approx}}$, obtained by computing the forward displacement problem of the calibrated model for the joint variables ($\mathbf{q}_{\text{approx}}$). The joint variables are the nominal inverse kinematics solution of a shifted pose, $\mathbf{H}_{\text{fake}}$, which is calculated at each iteration. The desired pose $\mathbf{H}_d$ is always imposed numerically and it is the position where we would like to place the robot end effector with respect to the robot base. This so-called *fake pose* accumulates the pose errors of the previous iterations. As the algorithm evolves, $\mathbf{H}_{\text{approx}}$ gets closer to the desired pose $\mathbf{H}_d$, these two poses are related by a pose error $\mathbf{H}_e$. The algorithm is stopped when an error tolerance is achieved for $\mathbf{H}_e$.

For simplicity, the expression $\mathbf{T}_6$ is defined as the analytical forward kinematic model (using the nominal parameters), which depends on the joint variables ($\mathbf{q}$)

$$\mathbf{T}_6 = \mathbf{A}_1^0\mathbf{A}_2^1\mathbf{A}_3^2\mathbf{A}_4^3\mathbf{A}_5^4\mathbf{A}_6^5 = \mathbf{T}_6(\mathbf{q}). \tag{10}$$

Defining $\mathbf{T}_{A6}$ as the forward displacement model of the calibrated robot (where the index A stands for "actual", closer to the real behavior), which depends on the joint variables, $\mathbf{q}$, and the identified error parameters, we have once again

$$\mathbf{T}_{A6} = \mathbf{A}_0^{\text{world}}\mathbf{A}_1^0\mathbf{A}_{A,2}^1\mathbf{A}_{A,3}^2\mathbf{A}_{A,4}^3\mathbf{A}_{A,5}^4\mathbf{A}_{A,6}^5 = \mathbf{T}_{A6}(\mathbf{q}). \tag{11}$$

If we want to find the inverse solution for a desired pose $\mathbf{H}_d$, which represents the pose of frame 6 with respect to the robot base frame, we must proceed with the following steps of the iteration k:

A. Set $k = 1$.

B. Calculate the vector of joint variables $\mathbf{q}_{\text{approx},1}$, by solving the inverse kinematics of the nominal robot model at pose $\mathbf{H}_d$ [1–2] (see appendix).

C. Calculate the first pose shift as $\mathbf{H}_{\text{approx},k} = \mathbf{T}_{A6}(\mathbf{q}_{\text{approx},k})$.

D. If the error between $\mathbf{H}_{\text{approx},k}$ and $\mathbf{H}_d$ is smaller than the tolerance error, we take $\mathbf{q}_{\text{approx},k}$ as the joint solution of the calibrated model for $\mathbf{H}_d$; otherwise, we increase k and we continue with the following steps.

E. Calculate the fake pose using the following equation ($\mathbf{H}_{\text{fake},1} = \mathbf{H}_d$)

$$\mathbf{H}_{\text{fake},k} = \mathbf{H}_{\text{fake},k-1}(\mathbf{H}_{\text{approx},k-1})^{-1}\mathbf{H}_d \tag{12}$$

F. Find the vector $\mathbf{q}_{\text{approx},k}$ using the nominal robot model, i.e. using the algebraic inverse kinematics solution [1–2] for $\mathbf{H}_{\text{fake},k}$ (closed-form solution).

G. Go back to step C.

The block diagram of this algorithm is shown in Fig. 3, where IK stands for the inverse kinematics of the nominal robot model. Figure 4 illustrates the iterative approach for iteration k.

The core of the algorithm remains in step E, which calculates the fake pose that will find a better configuration of the robot joints at every iteration. This method differs from [5] in that the fake pose is calculated using matrix multiplications instead of additions. The main advantage is that a pure homogenous pose is obtained as a result. Referring to Fig. 4, we can see the geometric principle of the algorithm: the fake pose $\mathbf{H}_{\text{fake},k+1}$ is obtained as the pose shift between the desired pose $\mathbf{H}_d$ and the approximate pose $\mathbf{H}_{\text{approx},k}$, added to the fake pose of the previous iteration $\mathbf{H}_{\text{fake},k}$. Figure 4 could be seen as the second iteration (for the first iteration $\mathbf{H}_{\text{fake},1} = \mathbf{H}_d$).

The error between $\mathbf{H}_{\text{approx},k}$ and $\mathbf{H}_d$ takes into account the position error and the orientation error, and is defined in [20] (first and second term respectively)

$$E = \sqrt{x_e^2 + y_e^2 + z_e^2} + p\cos^{-1}\left(\frac{r_{e1,1} + r_{e2,2} + r_{e3,3}^{-1}}{2}\right), \tag{13}$$

where x_e, y_e, and z_e are the components of the position error and $r_{e1,1}, r_{e2,2}$, and $r_{e3,3}$ are the diagonal elements of the rotation matrix representing the orientation error. Note that for iteration k, the pose error is

$$\mathbf{H}_{e,k} = \mathbf{H}_d(\mathbf{H}_{\text{approx},k})^{-1} = \begin{bmatrix} r_{e1,1} & r_{e1,2} & r_{e1,3} & x_e \\ r_{e2,1} & r_{e2,2} & r_{e2,3} & y_e \\ r_{e3,1} & r_{e3,2} & r_{e3,3} & z_e \\ 0 & 0 & 0 & 1 \end{bmatrix}. \tag{14}$$

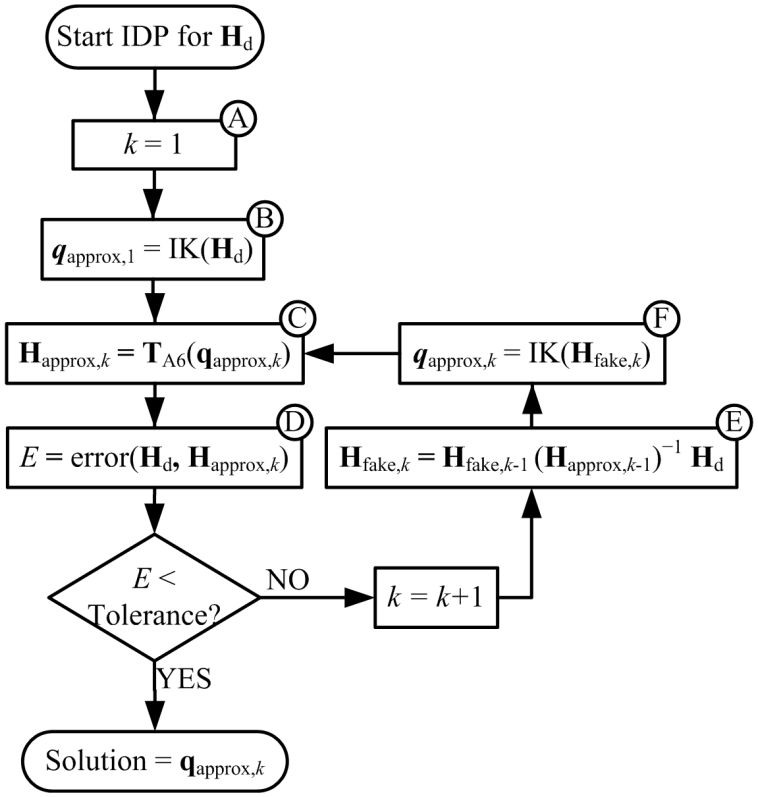

Fig. 3. Block diagram for the proposed algorithm.

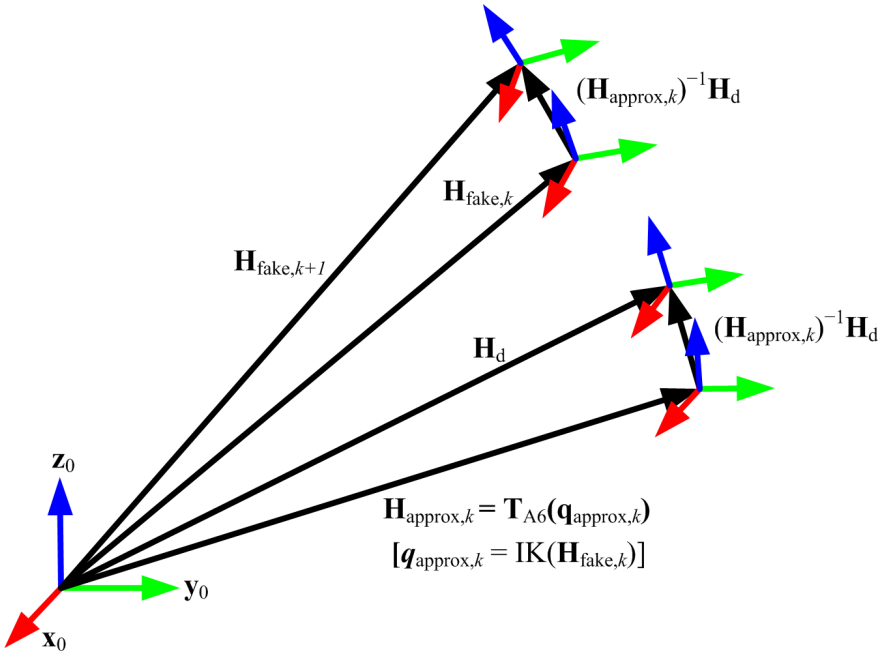

Fig. 4. Iterative representation for iteration k.

Finally, we used $p = (180/\pi)$ mm/° for our tests, so that the same importance is attributed to both 1 mm and 1° ($\cos^{-1}$ returns the angle in radians).

For some robot configurations, it can happen that the algorithm does not behave as expected. For example, there may be no solution after a certain number of iterations because we are too close to an elbow singularity. In the results section, we tagged these configurations as "unstable" and their behavior is studied in detail. A robot configuration is considered to be unstable if any of the three following conditions apply: (1) the resulting assembly mode for any iteration is different than the desired one (i.e., elbow-up vs. elbow-down), (2) the error increases or (3) the algorithm reaches ten iterations.

4. RESULTS

To experimentally test convergence, we used one million end-effector poses to numerically compute the inverse solution of our calibrated ABB IRB 120 industrial robot. Each end-effector pose is calculated by a random configuration of robot joints. The random configurations are equally distributed in the joint space, which means that for every configuration, a random joint angle set is chosen between the robot joint limits. The real robot is not used anytime to perform this tests. From each configuration, the modified forward kinematics is calculated obtaining a pose. Our inverse algorithm is applied to each of these poses. The goal is to obtain the joints that were used to generate every pose, so the final configuration must be the same. The algorithm can find up to eight solutions, however, it would take eight times longer to compute with respect to the presented results (the final configuration must be chosen after the first iteration).

It is important to note that, although it might appear that we deliberately chose a small robot to test our method, this particular robot is actually less accurate (before calibration) than a much larger industrial robot. Indeed, the ABB IRB 120 robot presented a maximum position error of about 5 mm before calibration. This error was reduced to less than 1 mm after calibration.

Our algorithm was implemented in MATLAB using mex functions to improve the algorithm speed. Although MATLAB is known to be relatively slow, our goal was not to obtain the solution as quick as possible, but to evaluate its convergence and find the average number of iterations needed to reach the solution for a given tolerance. MATLAB's profile function was used to track the time spent on each task.

We computed a maximum of 10 iterations for each of the end-effector pose. The position and orientation errors were stored for each iteration. Other information regarding the stability of our algorithm was also stored.

To decide when to stop iterating for one robot configuration, we can force the algorithm to stop when the error function reaches a certain numerical tolerance. Taking into account that the unidirectional repeatability of the ABB IRB 120 robot is 0.010 mm, we considered that it was more than sufficient to use a numerical error tolerance of $E = 0.001$.

Table 3 shows the average and the standard deviation of the position and orientation errors at each iteration for which $E < 0.001$. For every iteration, if $E < 0.001$, the configuration is considered to be stable and the algorithm is stopped; if any of the unstable conditions apply, the configuration is considered unstable and the algorithm is stopped; otherwise, the algorithm keeps iterating. For the one million configurations tested, 94.43% were found to be stable. Table 3 also shows the number of configurations that become unstable for each iteration.

We found that up to four iterations were enough in most cases (94.65%) to obtain a result. From this table, we can also calculate the average number of iterations needed, which is about 3.28 (the standard deviation being 1.24 iterations). After eight iterations, the position and orientation errors are negligible.

Using MATLAB's profile function, we can display the time spent for each step of the algorithm presented in Fig. 3 (with their respective tags) in Table 4.

Table 3. Stabilization of the iterative inverse displacement algorithm.

Iterations	Position error (mm)		Orientation error (°)		Stable configurations ($E < 0.001$)	Unstable configurations
	μ	σ	μ	σ		
1	1.411	0.740	0.286	0.108	0	0
2	0.009	0.024	0.001	0.004	18,498	248
3	0.001	0.010	0.000	0.001	889,789	3,442
4	0.006	0.023	0.001	0.002	25,204	9,337
5	0.006	0.020	0.001	0.001	6,326	5,936
6	0.003	0.012	0.000	0.001	2,503	4,078
7	0.001	0.004	0.000	0.001	1,150	3,171
8	0.001	0.001	0.000	0.000	565	2,422
9	0.001	0.000	0.000	0.000	283	1,962
Configs. with 10 iterations or more are considered unstable					0	25,086
Total					944,318	55,682

Table 4. Time spent for each step of the algorithm.

Tag	Time (%)	Description
C	52.6	Solving the calibrated forward displacement problem
D	1.7	Computing the error function
E	1.7	Computing the fake pose
F	35.7	Solving the nominal inverse kinematic problem
	8.3	Other computations

As we can see from Table 4, solving the forward displacement problem of the calibrated robot model and the inverse kinematics of the nominal model accounts for 88.3% of the total computation time (tag C plus tag F). We can conclude that the time needed to calculate the inverse solution is about four times the time needed to calculate the nominal inverse kinematics plus the forward kinematics of the calibrated method. Also, it takes 21 μs to compute the nominal inverse kinematics by itself; however, when used in combination with the iterative inverse algorithm the effective computation time is improved. Surprisingly, the time needed to calculate the forward displacement problem of the calibrated model takes longer than that needed to solve the inverse kinematics of the nominal model. This is because of the calculation of the joint torques, which, as explained in Section 2, is based on an iterative algorithm that takes into account every link mass and the direction of gravity.

The time needed to run the algorithm for a given pose was 64 μs on the average with 16 μs as standard deviation. All computation times were obtained using a PC with an Intel i7 processor running one single thread.

A study of the 55,682 unstable configurations revealed that they were all close to at least one of the three types of singularities [21] (also, see http://youtu.be/zlGCurgsqg8): (1) an elbow singularity with the arm extended with θ_3 close to -76.95° (94% of the unstable configurations), (2) a wrist singularity with θ_5 close to zero (6% of the unstable configurations) and/or (3) a shoulder singularity with the center of the robot wrist lying close to axis 1 (2% of the unstable configurations).

Since 94% of the unstable configurations were close to an elbow singularity, these were studied in greater detail. Figure 5 shows a histogram for these configurations with respect to angle θ_3. Note that at $\theta_3 =$

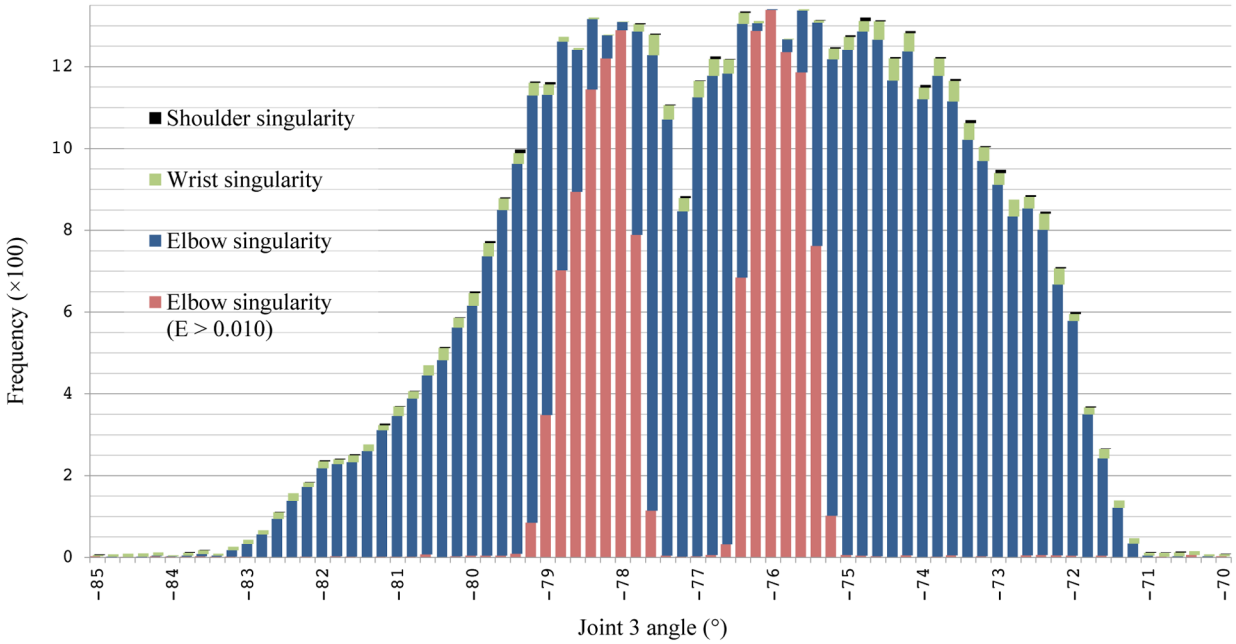

Fig. 5. Unstable configurations close to the elbow singularity ($\theta_3 = -76.95°$).

$-76.95°$, joint 3 should rotate more than $10°$ for an end-effector deviation of about 5 mm (the maximum nominal error for our robot). Configurations that were additionally close to a shoulder singularity or a wrist singularity are displayed in black and light green respectively (i.e., the top two parts of each bar in the histogram). In addition, the configurations that had an error $E > 0.001$ mm are shown in light red (i.e., the bottom part of each bar in the histogram).

We can see that the closer we get to the elbow singularity, the more chances that we have to find an unstable configuration. We can also see that the histogram highlights two peaks. This result is expected since the calibration of a decoupled robot turns it into a coupled robot. Finally, we found that the configurations having an error $E > 0.010$ mm are less than one third of the unstable configurations (15,623).

If we perform the same test, with the same robot configurations, the same robot model and the same error tolerance ($E = 0.001$), using the method described in [5], we obtain the results shown in Table 5.

Overall, the number of iterations needed to compute the inverse algorithm is slightly higher. The mean number of iterations is 3.36 and the standard deviation 1.34. In addition, slightly more configurations are found to be unstable. However, the distribution of unstable configurations with respect to singularities in the nominal robot model is very similar to that in the case of our algorithm.

The time needed to run the algorithm for a given pose was 65 µs on the average with 17 µs as standard deviation, which is slightly more than in the case of our algorithm.

We also tested both algorithms in the case of an ABB IRB 1600-1.45/6 industrial robot [22] with one million configurations. The number of iterations needed (average plus three times sigma) to reach a numerical stability of 0.001 mm was 4.39, using our algorithm, and 4.66, using the algorithm presented in [5]. The unstable configurations were 9,496 and 10,253, respectively. This industrial robot presents a mean position error of 0.968 mm before calibration, and 0.364 mm after calibration.

Finally, different numerical error tolerances E were tested (ranging between 0.0001 and 0.01) with both methods and both robots, and in all cases, our method was found to be slightly more stable and faster.

The results presented in this section are based in one calibrated model. We repeated the same analysis using 10,000 different random models in 10,000 different configurations (using the same set of joints per

Table 5. Stabilization using the inverse displacement algorithm proposed by Chen and Parker [5].

Iterations	Position error (mm)		Orientation error (°)		Stable configurations E < 0.001	Unstable configurations
	μ	σ	μ	σ		
1	1.409	0.740	0.285	0.108	0	0
2	0.011	0.029	0.005	0.014	25,119	337
3	0.001	0.011	0.000	0.004	841,967	3,059
4	0.004	0.017	0.001	0.004	53,413	9,011
5	0.004	0.015	0.001	0.003	12,058	5,898
6	0.002	0.010	0.001	0.002	4,701	4,009
7	0.001	0.005	0.001	0.001	2,248	3,192
8	0.001	0.001	0.001	0.000	1,183	2,467
9	0.001	0.000	0.000	0.000	687	1,978
Configs. with 10 iterations or more are considered unstable					0	25,086
Total					941,376	58,624

model). The standard deviation of the error parameters used to generate the random models was chosen so that an average of 3.2 iterations was needed to compute the inverse algorithm. The number of the unstable configurations was 9.2% less on average compared to [5]. The number of unstable configurations varied from 1% to 8% among the total 10,000 configurations for our method. On the other hand, the number of iterations to compute the inverse algorithm was improved by only 0.13 iterations on average.

5. CONCLUSION

The method presented for solving the inverse displacement problem was found to reach numerical stability after only 3.28 iterations, on average, using a very tight numerical tolerance of 0.001 mm. The computational time for this novel algorithm is about four times the time needed to calculate the inverse kinematics of the nominal robot model plus the forward kinematics of the calibration model. Furthermore, our method is relatively simple, and can even be implemented directly into the controller of a commercial industrial robot. For example, we were able to implement the complete procedure for computing the fake targets in ABB's RAPID programming language. As a result, although the user still needs a separate PC to run the identification software, filtering the so-called "robtargets" can be performed online, and the process will be transparent to the user. In contrast, Nikon Metrology, for example, offers separate robot calibration software packages, one for the identification (*Rocal ID*) and another for the filtering (*Rocal RT*), and they are both PC-based. Furthermore, the filtering package alone costs more than US$13,000.

Our method can also provide up to eight solutions (though this would require eight times the computational time), this is obvious since these solutions correspond to the eight different modes of assembly (robot configurations) that can be analytically found during the first iteration. In this case, we only need one solution, as, after the first iteration, we have a choice of up to eight approximate solutions. At this time, one of these solutions must be chosen, knowing that the final solution will be close to the one selected. This is why our method cannot work when the nominal robot model is in a singular configuration, but it may work well in proximity to such a configuration. If we want more solutions, the iterative method must be repeated for every additional solution desired.

As previously mentioned, our method is similar to the one proposed by Chen and Parker [5]. However, our algorithm is based on a geometric principle instead of a truncated series approximation of the desired

pose. It was also found to be slightly faster and more stable. Indeed, the method proposed in [5] is more unstable close to singular configurations, since 58,624 configurations were found to be unstable (vs. 55,682 using our algorithm). In addition, since this method is based on an iterative first order approximation of each component of the homogeneous transformation matrix representing the desired pose, the calculated fake pose is not homogeneous for each iteration. As a result, computing the nominal inverse kinematics of this fake pose sometimes leads to negative square roots, which become imaginary variables that must be ignored.

ACKNOWLEDGMENTS

This work was funded by La Caixa d'Estalvis i Pensions de Barcelona, "la Caixa", the Canada Research Chair program and the Canada Foundation for Innovation.

REFERENCES

1. Craig, J.J., *Introduction to Robotics: Mechanics and Control* (3rd ed.), Prentice Hall, 2004.
2. Angeles, J., *Fundamentals of Robotic Mechanical Systems: Theory, Methods, and Algorithms* (3rd ed.), Springer Verlag, 2007.
3. Veitschegger, W. and Wu, C., "A method for calibrating and compensating robot kinematic errors", in *Proceedings of 1987 IEEE International Conference on Robotics and Automation*, pp. 39–44, 1987.
4. Vuskovic, M.I., "Compensation of kinematic errors using kinematic sensitivities", in *Proceedings of 1989 IEEE International Conference on Robotics and Automation*, pp. 745–750, 1989.
5. Chen, N. and Parker, G.A., "Inverse kinematic solution to a calibrated PUMA 560 industrial robot", *Control Engineering Practice*, Vol. 2, pp. 239–245, 1994.
6. Siciliano, B., "A closed-loop inverse kinematic scheme for on-line joint-based robot control", *Robotica*, Vol. 8, pp. 231–243, 1990.
7. Angeles, J. and Zanganeh, K., "The semigraphical determination of all real inverse kinematic solutions of general six-revolute manipulators", in *Proceedings of the Ninth CISM-IFToMM Symposium on Theory and Practice of Robots and Manipulators*, pp. 23–32, 1993.
8. Xin, S.Z., Feng, L.Y., Bing, H.L. and Li, Y.T., "A simple method for inverse kinematic analysis of the general 6R serial robot", *Journal of Mechanical Design*, Vol. 129, p. 793–798, 2007.
9. Manocha, D. and Canny, J.F., "Efficient inverse kinematics for general 6R manipulators", *IEEE Transactions on Robotics and Automation*, Vol. 10, No. 5, pp. 648–657, 1994.
10. Duffy, J. and Crane, C., "A displacement analysis of the general spatial 7 link, 7R mechanism", *Mechanism and Machine Theory*, Vol. 15, No. 3, pp. 153–169, 1980.
11. Husty, M.L., Pfurner, M. and Schröcker, H.P., "A new and efficient algorithm for the inverse kinematics of a general serial 6R manipulator", *Mechanism and Machine Theory*, Vol. 42, No. 1, pp. 66–81, 2007.
12. Qiao, S., Liao, Q., Wei, S. and Su, H.J., "Inverse kinematic analysis of the general 6R serial manipulators based on double quaternions", *Mechanism and Machine Theory*, Vol. 45, No. 2, pp. 193–199, 2010.
13. Meng, Y. and Zhuang, H., "Self-calibration of camera-equipped robot manipulators", *The International Journal of Robotics Research*, Vol. 20, No. 1, p. 909–921, 2001.
14. Angeles, J., "On the numerical solution of the inverse kinematic problem", *The International Journal of Robotics Research*, Vol. 4, No. 2, pp. 21–37, 1985.
15. Gupta, K. and Kazerounian, K., "Improved numerical solutions of inverse kinematics of robots", in *Proceedings of 1985 IEEE International Conference on Robotics and Automation*, pp. 743–748, 1985.
16. Tsai, Y.T. and Orin, D.E., "A strictly convergent real-time solution for inverse kinematics of robot manipulators", *Journal of Robotic Systems*, Vol. 4, pp. 477–501, 1987.
17. Goldenberg, A.A., Apkarian, J.A. and Smith, H.W., "A new approach to kinematic control of robot manipulators", *Journal of Dynamic Systems, Measurement, and Control*, Vol. 109, p. 97–103, 1987.
18. Wang, L.C.T. and Chen, C.C., "A combined optimization method for solving the inverse kinematics problems of mechanical manipulators", *IEEE Transactions on Robotics and Automation*, Vol. 7, No. 4, pp. 489–499, 1991.

19. Nubiola, A., Slamani, M., Joubair, A. and Bonev, I.A., "Comparison of two calibration methods for a small industrial robot based on an optical CMM and a laser tracker", *Robotica*, pp. 1–20, 2013 (online).
20. Schneider, P.J. and Eberly, D.H., *Geometric Tools for Computer Graphics*, Morgan Kaufmann Publishers, 2003.
21. Hayes, M.J.D., Husty, M.L. and Zsombor-Murray, P.J., "Singular configurations of wrist-partitioned 6R serial robots: A geometric perspective for users", *Transactions of the Canadian Society for Mechanical Engineering*, Vol. 26, No. 1, pp. 41–55, 2002.
22. Nubiola, A. and Bonev, I.A., "Absolute calibration of an ABB IRB 1600 robot using a laser tracker", *Robotics and Computer-Integrated Manufacturing*, Vol. 7, pp. 236–245, 2012.

APPENDIX: INVERSE KINEMATICS SOLUTION OF 6R DECOUPLED ROBOTS

The analytical inverse kinematics for the ABB IRB 120, as well as many other 6R decoupled robots, can be obtained as explained in this sections 1 and 2. This section takes for granted that the nominal parameters of the robot are used, so all errors (or "deltas") from Table 2 are omitted.

The position of the center of the wrist (the origin of the 4th frame using the DH representation) will allow us to find θ_1, θ_2 and θ_3, which can be represented as

$$\mathbf{p}_4^0 = \mathbf{A}_1^0\,\mathbf{A}_2^1\,\mathbf{A}_3^2\,\mathbf{A}_4^3 \begin{bmatrix} o \\ 0 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} p_{x4} \\ p_{y4} \\ p_{z4} \\ 1 \end{bmatrix} = \begin{bmatrix} \mathrm{c}_1(a_2 + a_3\mathrm{s}_2 + d_4\mathrm{c}_{23} + a_4\mathrm{s}_{23}) \\ \mathrm{s}_1(a_2 + a_3\mathrm{s}_2 + d_4\mathrm{c}_{23} + a_4\mathrm{s}_{23}) \\ d_1 + a_3\mathrm{c}_2 - d_4\mathrm{s}_{23} + a_4\mathrm{c}_{23} \\ 1 \end{bmatrix} = \begin{bmatrix} p_x - d_6 a_x \\ p_y - d_6 a_y \\ p_z - d_6 a_z \\ 1 \end{bmatrix}. \tag{15}$$

Combining the two first equations in (18), the solution for θ_1 is quite straightforward if the term $a_2 + a_3\mathrm{s}_2 + d_4\mathrm{c}_{23} + a_4\mathrm{s}_{23}$ is not equal to zero. There are two solutions for θ_1

$$\theta_{1,1} = \operatorname{atan2}(p_y - d_6 a_y, p_x - d_6 a_x), \tag{16}$$

$$\theta_{1,2} = \operatorname{atan2}(-p_y + d_6 a_y, -p_x + d_6 a_x), \tag{17}$$

The shoulder singularity occurs when $p_y = d_6 a_y$ and $p_x = d_6 a_x$, which means that the term $a_2 + a_3\mathrm{s}_2 + d_4\mathrm{c}_{23} + a_4\mathrm{s}_{23}$ is zero. This is the only case where we cannot apply (18). Once θ_1 is known, the solution for θ_3 can be obtained using the following equation derived from (17)

$$a_3^2 + d_4^2 + a_4^2 - 2a_3 d_4 \mathrm{s}_3 + 2a_4 a_3 \mathrm{c}_3 = (p_{z_4} - d_1)^2 + (\mathrm{c}_1 p_{x_4} + \mathrm{s}_1 p_{y_4} - a_2)^2. \tag{18}$$

This equation is in the form $A\sin(x) + B\cos(x) = C$ and the explicit solutions are

$$\theta_{3,1} = \operatorname{atan2}(-B\sqrt{B^2 + A^2 - C^2} + AC, A\sqrt{B^2 + A^2 - C^2} + CB), \tag{19}$$

$$\theta_{3,2} = \operatorname{atan2}(\sqrt{B^2 + A^2 - C^2} + AC, -A\sqrt{B^2 + A^2 - C^2} + CB). \tag{20}$$

Finally, θ_2 can be obtained from (21) and (22) since θ_3 and θ_3 are known. Rewriting these equations

$$\mathrm{s}_2(22) + \mathrm{c}_2(23): \qquad a_3 - d_4\mathrm{s}_3 + a_4\mathrm{c}_3 = k_1\mathrm{s}_2 + k_2\mathrm{c}_2, \tag{21}$$

$$\mathrm{c}_2(22) - \mathrm{s}_2(23): \qquad d_4\mathrm{c}_3 + a_4\mathrm{s}_3 = \mathrm{c}_2 k_1 - \mathrm{s}_2 k_2, \tag{22}$$

where $k_1 = \mathrm{c}_1 p_{x_4} + \mathrm{s}_1 p_{y_4} - a_2$ and $k_2 = p_{z_4} - d_1$. Then, joint θ_2 is completely defined

$$\theta_2 = \operatorname{atan2}((a_3 - d_4\mathrm{s}_3 + a_4\mathrm{c}_3)k_1 - (d_4\mathrm{c}_3 + a_4\mathrm{s}_3)k_2, \quad (a_3 - d_4\mathrm{s}_3 + a_4\mathrm{c}_3)k_2 + (d_4\mathrm{c}_3 + a_4\mathrm{s}_3)/k_1). \tag{23}$$

To calculate θ_4, θ_5 and θ_6 we must use the information contained in $\mathbf{R}_6^3$, where

$$\mathbf{R}_6^3 = (\mathbf{R}_1^0\,\mathbf{R}_2^1\,\mathbf{R}_3^2)^{-1}\,\mathbf{R}_6^0 = \mathbf{R}_4^3\,\mathbf{R}_5^4\,\mathbf{R}_6^5 = \begin{bmatrix} r_{1,1} & r_{1,2} & r_{1,3} \\ r_{2,1} & r_{2,2} & r_{2,3} \\ r_{3,1} & r_{3,2} & r_{3,3} \end{bmatrix} = \begin{bmatrix} * & * & -c_4 s_5 \\ -s_5 c_6 & s_5 s_6 & c_5 \\ * & * & s_4 s_5 \end{bmatrix}. \tag{24}$$

We omitted the expressions marked with $*$ as they are not of interest. Two solutions for θ_5 can be directly obtained:

$$\theta_{5,1} = \operatorname{atan2}(\sqrt{1 - r_{2,3}^2}, r_{2,3}), \tag{25}$$

$$\theta_{5,2} = \operatorname{atan2}(-\sqrt{1 - r_{2,3}^2}, r_{2,3}). \tag{26}$$

Once θ_5 is known, θ_4 and θ_6 are completely defined if θ_5 is not equal to 0

$$\theta_4 = \operatorname{atan2}\left(\frac{r_{3,3}}{s_5}, \frac{-r_{1,3}}{s_5}\right), \tag{27}$$

$$\theta_6 = \operatorname{atan2}\left(\frac{r_{2,2}}{s_5}, \frac{-r_{2,1}}{s_5}\right) \tag{28}$$

Otherwise, if $\theta_5 = 0$ we have a wrist singularity and the equations can be simplified by forcing $\theta_4 = 0$. If all joints are limited in the range $[-180^\circ, 180^\circ]$, there are eight possible solutions (see http://youtu.be/xjjAiAmV05I). If this range is extended there can be more solutions.