Blind Search inverse kinematics for controlling all types of serial-link robot arms

Samuel N Cubero

University of Southern Queensland, Toowoomba, Australia

1 Introduction

The main objective of "Inverse Kinematics" (IK) is to find the joint variables of a serial-link manipulator to achieve a desired position and orientation relationship between the end-effector frame and a base (or reference) frame. This paper describes a general purpose Inverse Kinematics (IK) method for solving all the joint variables for any type of serial-link robotic manipulator using its Forward Kinematic (FK) solution. This method always succeeds in solving the IK solution for any design of articulated, serial-link robot arm. It will always work on any design of serial-link manipulator, regardless of the number or types of joints or degrees of freedom (rotary and/or translational) and it is simple and easy enough to be implemented into robot arm design and simulation software, even automatically, without any need for complex mathematics or custom derived equations. Known as the "Blind Search" method, it also works on robots with redundant joints and with workspace internal singularities and will not become unstable or fail to achieve an IK solution. Robot arm design and 3D simulation software has been written and has successfully demonstrated that the "Blind Search" algorithm can be used as a general-purpose IK method that is capable of controlling all types of robot arm designs and even 3D animated objects and characters. The speed of solving IK solutions numerically is dependent on software design, selected search parameters and processing power.

2 Background of inverse kinematics

The majority of modern serial-link robotic arms and manipulators are designed to manipulate an end-effector tool to follow a predefined trajectory or path in 3D space (ie. A series of known or calculated cartesian-coordinate positions and roll-pitch-yaw angular orientations of the tool). For example, a 6 degree-of-freedom welding robot must have its 6 individual joint variables (or joint angles) controlled

simultaneously so that the tip of a MIG welding tool attached at its tool-plate can be placed accurately at known positions along a welding path. (eg. a straight line)

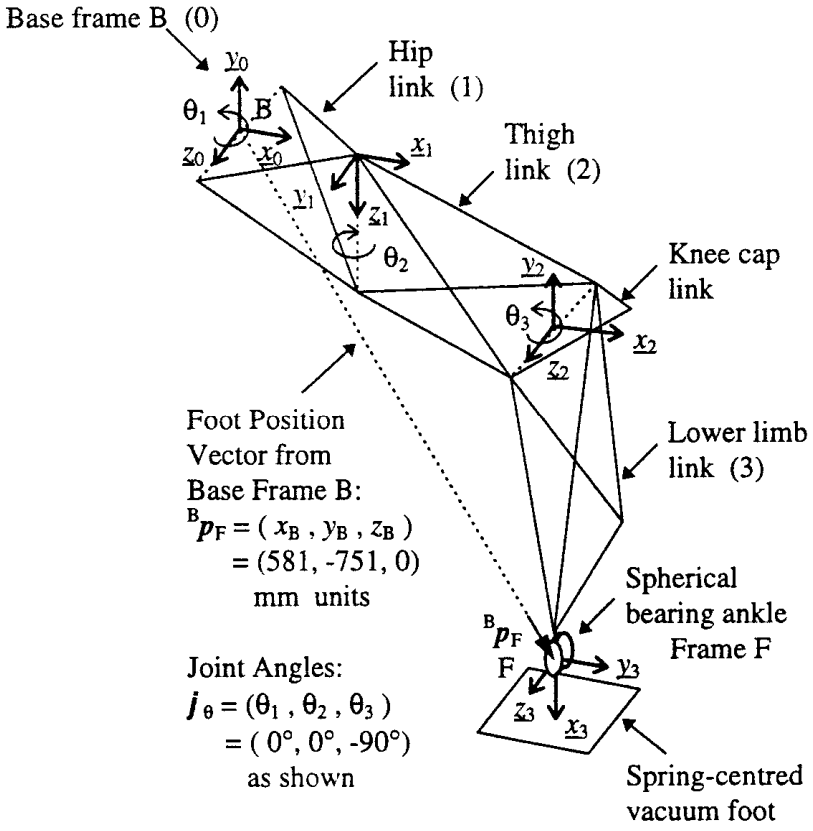

Fig. 1. Coordinate frames for the links of a 3 degree-of-freedom robot leg (example)

Using the Denavit Hartenberg (D-H) convention for obtaining an A-matrix for each link, the Forward Kinematic (FK) solution for the leg shown in Fig. 1 is given by the x_B , y_B , z_B coordinates for the foot, measured from Frame B's origin.

$$x_B = \cos\theta_1 \cos\theta_2 (l_3 \cos\theta_3 + l_2) + \sin\theta_1 (d_2 - l_3 \sin\theta_3) + l_1 \cos\theta_1 \quad (1)$$

$$y_B = \sin\theta_1 \cos\theta_2 (l_3 \cos\theta_3 + l_2) - \cos\theta_1 (d_2 - l_3 \sin\theta_3) + l_1 \sin\theta_1 \quad (2)$$

$$z_B = \sin\theta_2 (l_3 \cos\theta_3 + l_2) \quad (3)$$

(where l_1 , l_2 and l_3 are link lengths according to the D-H convention)

These FK equations were derived by multiplying together three A-matrices:

$^{B}\mathbf{T}_{F} = {}^{0}\mathbf{T}_{3} = {}^{0}\mathbf{A}_{1}\ {}^{1}\mathbf{A}_{2}\ {}^{2}\mathbf{A}_{3}$ = transform relating frame 3 to frame 0 (4)

The two most common "general-purpose" methods for obtaining the IK solution of serial-link robotic arms or manipulators, are: (1) Using "Inverse Transforms": Paul's method [1] of obtaining a "closed-form" explicit solution for each joint variable; and (2) Using the "Jacobian Inverse" method to obtain an incremental IK solution [2]. Using Paul's method, we face the problem of being unable to express a joint angle as an explicit function of (x_B , y_B , z_B) and the D-H link parameters only, independent of the other joint angles. Hence, there appears to be no algebraic, closed-form IK solution for this example manipulator since the joint angles are mathematically dependent on each other. Closed-form IK solutions cannot be found for several other designs of robot manipulators using this method. Hence, Paul's "Inverse Transform" method is not a truly "general purpose" IK method that works for all manipulator designs.

A commonly used technique for obtaining an IK solution is by inverting the manipulator's Jacobian matrix, either symbolically or numerically. The Jacobian, $\mathbf{J}$, is a special type of transformation matrix which relates incremental changes in joint angles, $\Delta \boldsymbol{j}_\theta = (\Delta\theta_1 , \Delta\theta_2 , \Delta\theta_3)$, to the robot end-effector's incremental movements in cartesian space, $\Delta^{B}\boldsymbol{p}_F = (\Delta x_B , \Delta y_B , \Delta z_B)$, relative to the base or reference frame. The relationship between incremental 3D cartesian displacements of the end-effector frame, $\Delta^{B}\boldsymbol{p}_F$, and incremental changes in joint angles, $\Delta \boldsymbol{j}_\theta$, is given as

$$\Delta^{B}\boldsymbol{p}_F = {}^{B}\mathbf{J}_\theta \, \Delta \boldsymbol{j}_\theta \tag{5}$$

from McKerrow [2], where the Jacobian matrix, $\mathbf{J}$, is given by

$$ {}^{B}\mathbf{J}_\theta = \begin{bmatrix} \frac{\partial x_B}{\partial \theta_1} & \frac{\partial x_B}{\partial \theta_2} & \frac{\partial x_B}{\partial \theta_3} \\ \frac{\partial y_B}{\partial \theta_1} & \frac{\partial y_B}{\partial \theta_2} & \frac{\partial y_B}{\partial \theta_3} \\ \frac{\partial z_B}{\partial \theta_1} & \frac{\partial z_B}{\partial \theta_2} & \frac{\partial z_B}{\partial \theta_3} \end{bmatrix} \tag{6}$$

If we know the small changes in joint variables for all links of a robot, $\Delta \boldsymbol{j}_\theta = (\Delta\theta_1, \Delta\theta_2 , \Delta\theta_3)$, the Jacobian allows us to calculate the small "incremental" changes in the position of the end-effector or foot, $\Delta^{B}\boldsymbol{p}_F = (\Delta x_B , \Delta y_B , \Delta z_B)$. Hence, the "incremental" IK solution is:

$$\Delta \boldsymbol{j}_\theta = {}^{B}\mathbf{J}_\theta^{-1} \, \Delta^{B}\boldsymbol{p}_F \tag{7}$$

The Jacobian matrix cannot be inverted when the foot or the end-effector frame is near a workspace internal singularity, which is a position where there are an infinite number of possible joint variable solutions. There are two types of singularities. (1) A workspace internal singularity occurs inside the 3D workspace (volume of reachable positions), often when two or more joint axes or link frame axes

line up; and (2) A workspace boundary singularity occurs at or beyond the outermost surface of the robot's 3D workspace, when the manipulator is fully extended or fully retracted and attempts to move outside of its workspace (eg. outside of the joint angle ranges defined by the minimum and maximum permissible joint angles for each link). In Fig. 2, there are infinite possible values for the solution of θ_2 when the robot's foot (end-effector frame origin) is situated on the "internal singularity curve", where θ_3= -132.4°, or if the foot position (origin of frame F or 3) is in line with the $\underline{z}_1$ basis axis as shown in Fig. 1.

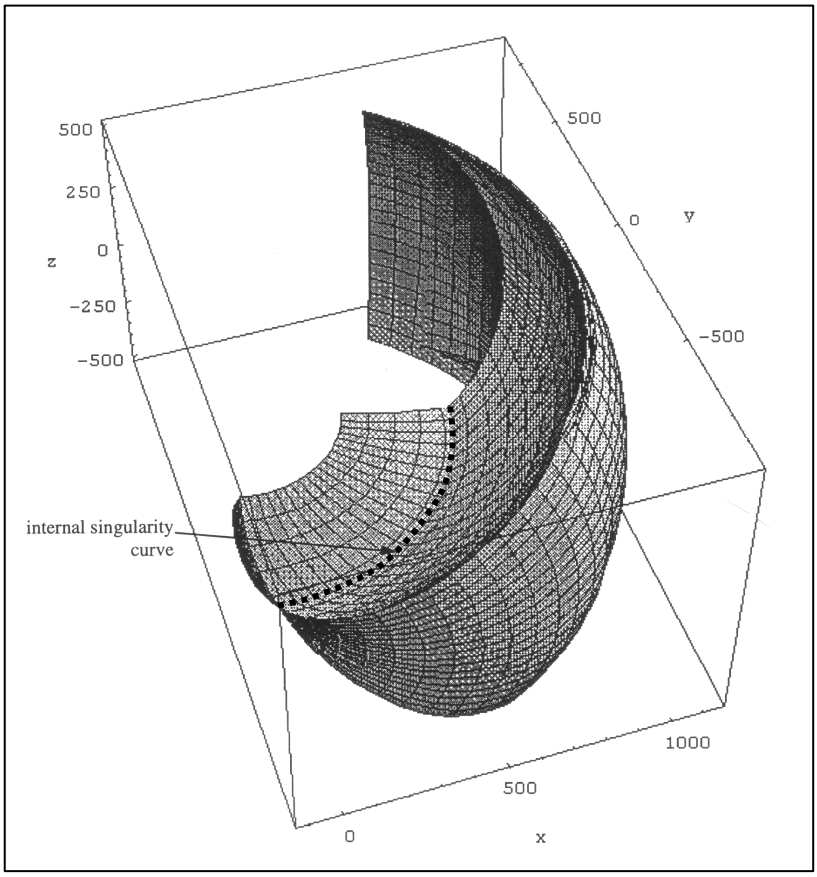

Fig. 2. Workspace of robot in Fig. 3 showing internal singularity curve [Cubero, 13]

At target positions near or on this internal singularity curve, the Jacobian matrix cannot be inverted and hence, no IK solution can be found unless additional algorithms are implemented to deal with this problem. Such singularity features can be identified by plotting the surfaces described by the FK equations. Most programmers of robot controllers are aware of the problems caused by internal singularities and, hence, do their best to avoid letting goal points or trajectory paths of

end-effectors go anywhere near these problem areas. The stability and accuracy of this IK method depends on the sensitivity of **J** to particular joint angle values.

There are several other IK techniques that have been proposed in the past, many more than can be described in detail here. The reader may wish to examine and compare other IK methods, such as: (a) "Screw algebra" by Kohli & Sohni [6], "Dual matrices" by Denavit [7], "Dual quaternian" by Yang & Freudenstein [8], "Iterative method" by Uicker et al [9] and Milenkovic & Huang [10], geometric or vector approaches by Lee & Ziegler [11], "Iterative method" by Goldenberg & Lawrence [14, 15] and Pieper's approach for solving Euler angles [12]. Not all of these IK methods work for every possible type of serial-link robot manipulator design, especially those with one or more redundant degrees of freedom or those with one or more internal singularities. Also, most of these IK methods require a great deal of complicated trigonometric analysis and equation derivations which are unique for each type of robot manipulator design, hence, their approaches cannot be universally applied to all types of serial-link manipulators. The above methods can be demonstrated to work on a few, particular examples of manipulators, but they do not offer a simple, methodical, procedural approach that will always achieve an IK solution for every type of serial-link manipulator. Many of these methods are also not simple enough to automate or use immediately in 3D manipulator design and control software and they require the derivation of unique IK equations or algorithms for different classes of robot arms. In many cases, IK solutions cannot be found because the end-effector frame is located near an internal singularity within the workspace, causing IK equations or Jacobian inversions to become unsolvable. Such problems are described by McKerrow [2]. The "Blind Search" IK method overcomes all of these limitations.

3 A new "Blind Search" incremental IK method

After considering the limitations of popular IK methods, the author [Cubero, 13] proposed a general purpose IK method in 1997 which relies heavily on a "trial and error" or error minimizing approach to solving the joint variables of any robot arm, given the desired origin position and orientation of the end-effector frame. Only the Forward Kinematic (FK) solution for a manipulator or its overall manipulator transform matrix is needed, hence, this method can be applied to any type of robot with one degree of freedom per link. This IK method solves all the correct joint variables needed which correspond to an incremental change in the displacement and axis orientations of a robot's end-effector frame in 3D space.

How the "Blind Search" IK method works

Consider a 3 dof serial-link manipulator with only rotary joints, such as the example robot leg in Fig. 1. Assuming that a small change in foot position is required, each joint angle (or displacement or joint variable) can do one of the following:

1. Increase (+) by a small displacement $\Delta\theta$ (rotary), or Δd (sliding)
2. Decrease (-) by a small displacement $\Delta\theta$ (rotary), or Δd (sliding); or
3. Remain the same (0) , no change

With forward kinematic equations, we can calculate the end-effector frame position error and/or basis axes alignment errors which arise for all possible combinations of small joint variable adjustments. For any serial-link manipulator with n degrees of freedom, there are a total of 3^n -1 different possible combinations for joint variable adjustments which can be made for a given $\Delta\theta$ (for rotary joints), or Δd for translational joints (excluding the "no changes" solution). For example, a 2 link robot arm would have 3^2-1= 8 possible combinations (excluding "no changes"). In the following discussion, we will consider solving the 3 joint angles for the example robot leg in Fig. 1, given the desired target position ${}^{B}\boldsymbol{t}_{T} = (x_B , y_B , z_B)$ for the foot relative to the base frame B. The following analysis can also be extended to 6 degree of freedom industrial manipulators, where the position of the end-effector (or tool) frame origin position and basis axes orientations are specified and must be achieved through small changes in joint variables. For simplicity, we will limit our discussion to an IK solution for achieving a desired (target) end-effector frame origin position. Later, we will extend this discussion to obtaining IK solutions which achieve a desired end-effector frame orientation and a desired (target) end-effector frame origin position, simultaneously. For the example robot shown in Fig. 1, there are N = 3^3-1 = 26 possible FK solutions for the foot position if each joint angle changes by $+\Delta\theta$, $-\Delta\theta$ or 0° (no change at all). We can calculate all the resulting position errors of these FK solutions from the desired foot position using simple vector subtraction (ie. Target position vector subtract the current position vector of the foot). We can then select the "best" combination of joint angle changes which produces the smallest computed foot position error as our incremental IK solution after doing a simple "minimum value search" on an array of error magnitudes from all combinations of joint angle changes. If the computed position error from this "best" combination of joint angle changes is not satisfactory, then the entire process can be repeated using the "best" or smallest error vector found so far (which tends to become smaller with more iterations), until the error tolerance is met. Note that the incremental change in foot position to the next target position must be small enough to be achieved, within tolerance, so that the target may be reached by any one or more of the possible combinations of joint angle changes. The test joint angles (subscript small "t") will tend to converge towards the correct solution if we continually select the combination of joint angle changes that minimizes foot position error relative to the target position. This algorithm will now be considered in detail.

As shown in Fig. 3, the target position vector for the foot, ${}^{B}\boldsymbol{t}_{T}$, starts from the base frame origin B and points to the target frame origin T. The foot position vector ${}^{B}\boldsymbol{p}_{F}$ starts from the base frame B and points to the foot F. The starting error vector, ${}^{F}\boldsymbol{e}_{T} = {}^{B}\boldsymbol{t}_{T} - {}^{B}\boldsymbol{p}_{F}$, starts from the foot F and points to the target frame origin T. The goal is to find a suitable set of joint angle changes to get the foot position F to end up at the target position T, within an acceptable error range (or tolerance). We need to find the combination of joint angle changes that achieves an error vector magnitude $|{}^{F}\boldsymbol{e}_{T}|$ less than an acceptable error tolerance within a reasonably short time, for quick real-time performance. We may need to search the FK solu-

tion for each and every possible combination of joint angle changes and select the combination of changes that gives us the smallest position error from the target position, which satisfies the required error tolerance.

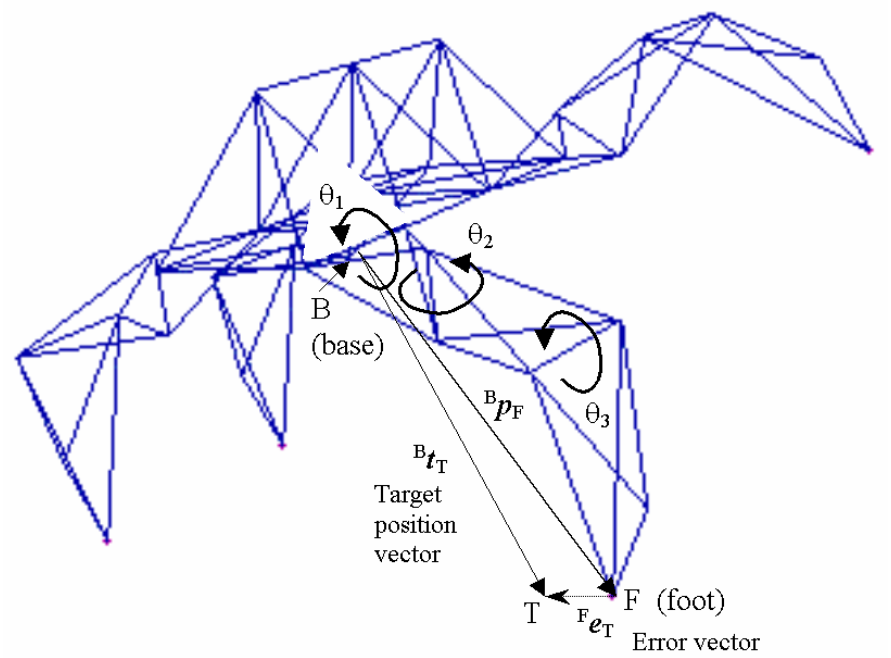

Fig. 3. Position vectors for foot position F and target point T

Combinations for test angles (index i)				***Position***	***Magnitude***
Test 1: $\theta_{1t1} = \theta_1+\Delta\theta$, $\theta_{2t1} = \theta_2+\Delta\theta$, $\theta_{3t1} = \theta_3+\Delta\theta$ (+,+,+)				${}^{B}\boldsymbol{p}_{Ft}$ (1)	$\lvert{}^{F}\boldsymbol{e}_{Tt}\rvert$ (1)
Test 2: $\theta_{1t2} = \theta_1+\Delta\theta$, $\theta_{2t2} = \theta_2+\Delta\theta$, $\theta_{3t2} = \theta_3-\Delta\theta$ (+, +, -)				${}^{B}\boldsymbol{p}_{Ft}$ (2)	$\lvert{}^{F}\boldsymbol{e}_{Tt}\rvert$ (2)
Test 3: $\theta_{1t3} = \theta_1+\Delta\theta$, $\theta_{2t3} = \theta_2-\Delta\theta$, $\theta_{3t3} = \theta_3+\Delta\theta$ (+, -, +)				${}^{B}\boldsymbol{p}_{Ft}$ (3)	$\lvert{}^{F}\boldsymbol{e}_{Tt}\rvert$ (3)
+, -, - (i = 4)	+, 0, +	-, -, 0	+, 0, -	(arrays	continue)
-, +, + (i =5)	+, 0, 0	-, 0, -	+, -, 0	etc.	etc.
-, +, - (etc)	0, +, +	-, 0, 0	-, 0, +		
-, -, +	0, +, 0	0, -, -	-, +, 0		
-, -, -	0, 0, +	0, -, 0	0, +, -	(test positions)	(test errors)
+, +, 0	*Ignore* 0, 0, 0	0, 0, -	0, -, + (i=26)	${}^{B}\boldsymbol{p}_{Ft}$ (26)	$\lvert{}^{F}\boldsymbol{e}_{Tt}\rvert$ (26)

Table 1. Angular displacement combinations for the example robot of Fig. 1

Table 1 and Fig. 4 show all the possible displacements that can be made for the example robot leg shown in Figures 1 and 4. The "Blind Search" IK algorithm that will now be discussed is suitable for finding accurate joint angle solutions as long as the end-effector position always remains within its workspace, as shown in Fig. 2, and the search parameters $\Delta\theta$ and $\lvert{}^{F}\boldsymbol{e}_{T}\rvert$ are kept small enough so that the magnitude of the error vector converges towards 0. If, after many iterations none of the error magnitudes calculated from all of the combinations satisfy the accepted error tolerance (ie. the "best" IK error magnitude $\lvert{}^{F}\boldsymbol{e}_{Tb}\rvert$ > etol), then the

initial incremental step (magnitude of the foot error vector) may need to be lowered because the target position T could be too far from the current foot position F.

In summary, the following steps are executed:

{1} A new target end-effector position/orientation is issued and this must be quite near to the current position/orientation of the end-effector frame.

{2} All the FK solutions and position/angle errors for each of the displacement combinations shown in Fig. 4 are calculated and stored in an array.

{3} A minimum value search is conducted to find out which displacement combination produced the smallest error and this combination is marked as the current "best IK solution". {4} If this error satisfies the acceptable error tolerance for a match, then the IK solution is found, otherwise, the process is repeated from step {2} where the most recent "best IK solution" is used as the starting or current robot configuration, until the best FK error set (joint angles and/or displacements) satisfies the error tolerance criteria. In effect, this is a form of algorithmic or iterative "feedback control" method being used to eliminate the difference between the actual current position/ orientation of the robot manipulator and the target position/orientation. This method searches for small joint angle/displacement changes so that the current joint angles/displacements move closer and closer to the correct IK solution (within a set tolerance).

The "Blind Search" algorithm can also be extended to find the IK solution that aligns the axes of both the end-effector frame and the target frame, while bringing together both their origins so that they coincide, within an acceptable error tolerance. Orienting the end-effector frame's x, y and z axes will be described later, so for now, we will consider achieving an IK solution for position only.

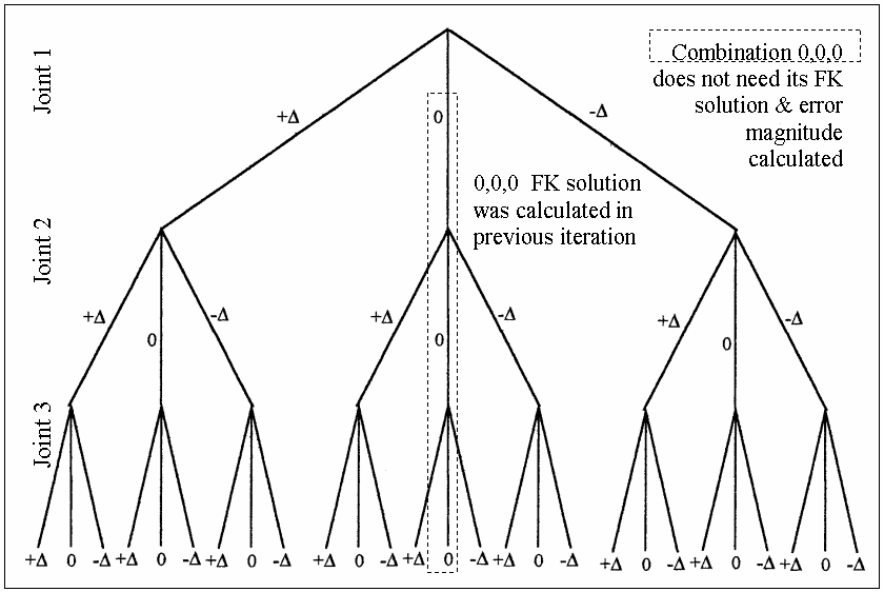

Fig. 4. Displacement combinations for all joints of a 3 degree-of-freedom manipulator

Algorithm 1. Blind Search IK method for the example robot shown in Fig. 1

1. Set the etol variable to an acceptable tolerance value for computed foot position error (eg. Let 0.02 mm FK error be acceptable for an "exact" position match, ie. Position of F must be within this distance from the target position T for accepting an IK solution. This is the worst possible error for the IK solution. Set $\Delta\theta$ to be proportional to the magnitude of the starting position error $| {}^{F}\boldsymbol{e}_{T} |$ or expected displacement of the end-effector (foot), where $\Delta\theta = k | {}^{F}\boldsymbol{e}_{T} |$, eg. k = 0.2 ° / mm (scale factor k depends on lengths of robot's links. The longer the l_n link lengths, the larger k should be). Try to keep displacements small, eg. $\Delta\theta <$ 2° but non-zero. (Algorithm is continued on following page…)
2. Initialize iteration counter variable c = 0 and set the counter limit cmax = 50 , which is the maximum number of iterations that will be executed in search of an IK solution whose error satisfies etol (error tolerance). Clear error flag *ef* = 0. (The error flag is only used if the magnitudes of the search parameters (initial error vector $|{}^{F}\boldsymbol{e}_{T}|$ or $\Delta\theta$) are too large, or if the target position T is outside the robot's workspace). These values must be carefully selected.
3. Trajectory planning algorithm supplies the next target foot position ${}^{B}\boldsymbol{t}_{T}$. Trajectory planning algorithm defines 3D path points for foot F. Keep all target positions close or within 10mm of each other, or perhaps even smaller, depending on size of $\Delta\theta$. *Note*: Initial step size $| {}^{F}\boldsymbol{e}_{T} |$ should be proportional to $\Delta\theta$. The error flag *ef* is noted, acted upon and cleared.
4. The initial "best" error $|{}^{F}\boldsymbol{e}_{Ttb}| = | {}^{F}\boldsymbol{e}_{T} | = |{}^{B}\boldsymbol{t}_{T} - {}^{B}\boldsymbol{p}_{F}|$ and initial "best" test IK solution is $\boldsymbol{j}_{\theta tb} = (\theta_{1tb}, \theta_{2tb}, \theta_{3tb}) = \boldsymbol{j}_{\theta} = (\theta_1, \theta_2, \theta_3)$ = currently incorrect model joint angles. A test foot position vector ${}^{B}\boldsymbol{p}_{Ft}$ (model of position) and the test error vector and its magnitude $|{}^{F}\boldsymbol{e}_{Tt}|$ are calculated for each combination and stored. ($|{}^{F}\boldsymbol{e}_{Ttb}|$ should decrease as c increases).
5. This step may have to be repeated $3^3 - 1 = 26$ times for each possible combination of joint angle changes or no changes. eg. FOR i = 1 to 26 (Loop). Calculate the test angles for each combination of joint angle changes, as shown in Table 1 (eg. $\theta_{1t1}, \theta_{2t1}, \theta_{3t1}$, where i =1) and check that each test joint angle lies within its valid range of motion (between the joint's minimum and maximum displacement limits, θ_{min} and θ_{max} for rotary joints, or d_{min} and d_{max} for translational joints). eg. If any test joint angle exceeds its minimum or maximum acceptable limit, then set it to that nearest limit. ie. do not allow any test angle outside of its valid range of motion (ie. every θ must always stay between θ_{min} and θ_{max} for that joint). Calculate a test foot position, ${}^{B}\boldsymbol{p}_{Fti}$, for this combination "i" using the test joint angles just found $\boldsymbol{j}_{\theta ti} = (\theta_{1ti}, \theta_{2ti}, \theta_{3ti})$ from Table 1 with FK (Eqs. 1-3) and store each test position into an array. Also store the test joint angles $\boldsymbol{j}_{\theta ti}$. Calculate a test position error vector for this combination, ${}^{F}\boldsymbol{e}_{Tti} = {}^{B}\boldsymbol{t}_{T} - {}^{B}\boldsymbol{p}_{Fti}$, and its magnitude $|{}^{F}\boldsymbol{e}_{Tti}|$ and record this into an array. Run an optional check to see if the magnitude of the test error for each combination satisfies the error tolerance. If error $| {}^{F}\boldsymbol{e}_{Tti} | \leq$ etol , this is an acceptable IK solution, so record this as $|{}^{F}\boldsymbol{e}_{Ttb}|$ and store the "best" joint angles $\boldsymbol{j}_{\theta tb} = (\theta_{1tb}, \theta_{2tb}, \theta_{3tb}) = (\theta_{1ti}, \theta_{2ti}, \theta_{3ti})$ then jump to Step 8, skipping steps 6 and 7 to save time. If not, repeat Step 5 (Next "i") and test the next combination of changes.

Algorithm 1 continued on next page…

Algorithm 1 continued (Blind Search IK method):

6. Search for the smallest error magnitude in the $|^F\boldsymbol{e}_{Tt}|$ (1 to 26 element) array using a simple "minimum value search" algorithm. The smallest error in this complete array of 26 error magnitudes is $|^F\boldsymbol{e}_{Tt}|(s)$, it's index number is remembered as "s" in the array and its joint variables (test IK solution) are recorded $\boldsymbol{j}_{\theta ts} = (\theta_{1ts}, \theta_{2ts}, \theta_{3ts})$. The smallest error magnitude found so far from all the iterations (since c = 0) is recorded as the "best" error, $|^F\boldsymbol{e}_{Ttb}|$. If $|^F\boldsymbol{e}_{Tt}|(s)$ from this array is less than the previously calculated "best error", then set the new "best error" to equal the current error, ie. if $|^F\boldsymbol{e}_{Tt}|(s) < |^F\boldsymbol{e}_{Ttb}|$, then set $|^F\boldsymbol{e}_{Ttb}| = |^F\boldsymbol{e}_{Tt}|(s)$, and update the best joint angles $\boldsymbol{j}_{\theta tb} = (\theta_{1tb}, \theta_{2tb}, \theta_{3tb}) = (\theta_{1ts}, \theta_{2ts}, \theta_{3ts})$.
7. Check that the test error is converging towards 0. If smallest error $|^F\boldsymbol{e}_{Tt}|(s) > |^F\boldsymbol{e}_{Ttb}|$, the last "minimum value search" did not find a better IK solution which produced a smaller "best" error than the one found from the previous pass of this algorithm. This could be caused by any of the search parameters $\Delta\theta$, k or the displacement $|^F\boldsymbol{e}_T|$, being too large, so they may be reduced or halved here if necessary. If c > cmax (eg. after 50 iterations) and error is still greater than the acceptable error tolerance, $|^F\boldsymbol{e}_{Ttb}| >$ etol , then set the "error flag" $ef = 1$ to inform the trajectory planning algorithm to take corrective action or modify its planned trajectory, eg. print "No solution found because the initial movement of the end-effector $|^F\boldsymbol{e}_T|$ was too large or $\Delta\theta$ or k are too large, or the target point T is outside the robot's workspace.", set c = 0 and go to Step 3 to retry this incremental move again. (The search parameters may be reduced automatically to avoid this error message) Otherwise, if c<<cmax and the "best" error sofar $|^F\boldsymbol{e}_{Ttb}| >$ etol , another iteration is necessary to hopefully find a smaller $|^F\boldsymbol{e}_{Ttb}| \leq$ etol, so calculate $^B\boldsymbol{p}_{Ft}$ using FK (Equations 1-3) with the best "test" angles sofar $\boldsymbol{j}_{\theta tb} = (\theta_{1tb}, \theta_{2tb}, \theta_{3tb})$, increment the loop counter c = c+1 and go to Step 5. If $|^F\boldsymbol{e}_{Ttb}| \leq$ etol , then the IK solution is found so proceed to Step 8.
8. If the error flag is clear, $ef = 0$, and the best test angles found so far produces an error magnitude $|^F\boldsymbol{e}_{Ttb}| \leq$ etol , then update the joint angles by equating them with the best test angles: $\theta_1 = \theta_{1tb}$, $\theta_2 = \theta_{2tb}$, $\theta_3 = \theta_{3tb}$. (etol must be large enough to obtain a fast solution).
9. Send this angle data to the position controllers of the actuators for links 1, 2 and 3, then Return to Step 3 to get the next target position $^B\boldsymbol{t}_T$. (Note that the new FK solution for $^B\boldsymbol{p}_F$ is calculated with the newest joint angles $\boldsymbol{j}_\theta = (\theta_1, \theta_2, \theta_3)$ which are accurate joint variables. The best test position $^B\boldsymbol{p}_{Ftb}$ should converge towards $^B\boldsymbol{t}_T$, or tend to get closer to the target with more passes of this algorithm, until the best error $|^F\boldsymbol{e}_{Ttb}| \leq$ etol (error tolerance) *Note*: This example describes "position control" only. To implement orientation control, use $|^F\boldsymbol{e}_{Ttb}| = e_{total}$ from Eq. (16) to also force the orientation of the end-effector's frame E to match the x, y & z basis axes of the target frame T. Additional code is necessary to control or guide the shape of redundant links towards a preferred posture or to approach the target from a particular 3D direction.

If the "best" error does not keep on improving and the error tolerance is not met after several iterations have completed, search parameters may be reduced in magnitude and the entire procedure can be repeated. This is not a problem that is normally encountered if these search parameters are selected carefully.

Forcing the End-effector frame to match the Target frame's origin position and orientation

The discussion in the previous section dealt mainly with finding an IK solution for a serial-link manipulator given a target position (origin of target frame T) which is very close to the origin of the end-effector frame E. We have so far only described how to move the origin of end-effector frame E to the origin of target frame T. We will now consider an extension to this incremental IK method and try to orient the $\underline{x}_E$ and $\underline{y}_E$ basis axes of end-effector frame E so that they point in the same directions as the corresponding $\underline{x}_T$ and $\underline{y}_T$ basis axes of the target frame T respectively. Note that only two of the three corresponding basis axes need alignment as long as both frames are "right handed". If we can find the robot's joint variables to move the origin of frame E to the origin of frame T, which is only a short distance away, while the corresponding x and y (and consequently z) basis axes of both frames are made to point in the same directions respectively (ie. $\underline{x}_E$ becomes colinear with $\underline{x}_T$ and $\underline{y}_E$ becomes colinear with $\underline{y}_T$) within an acceptable error tolerance, then the complete incremental IK solution is found for any type of serial-link arm. Consider the end-effector frame E for any type of multi-degree-of-freedom robot arm manipulator, as shown in Fig. 5.

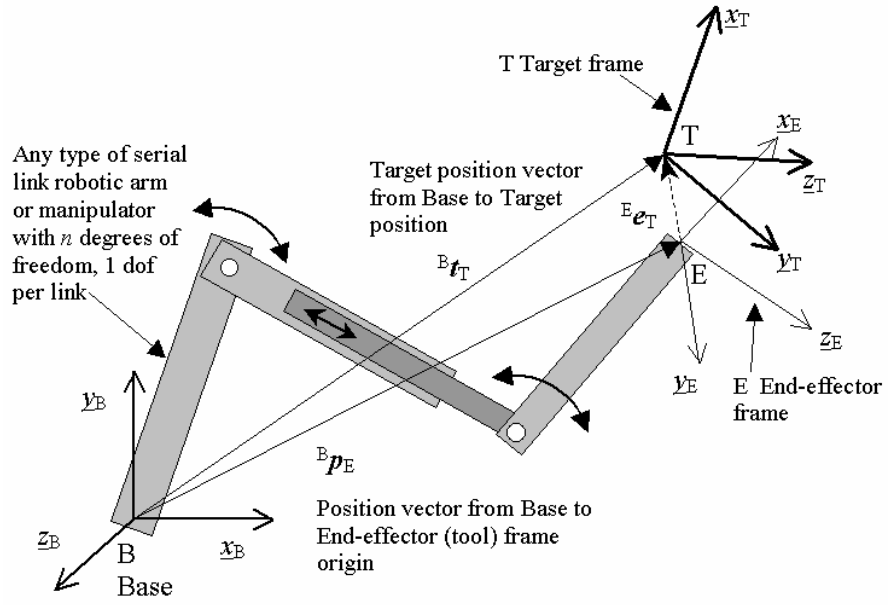

Fig. 5. Base, End-effector and Target frames for a serial-link manipulator

The robot arm shown in Fig. 5 is just for illustrative purposes only and this discussion applies to any serial-link robot arm design with one degree of freedom per link. Note that spherical joints (or "ball joints") can be treated like two rotary links, where one link has a zero link length ($l = 0$), so each link has one rotation/

displacement about one axis. eg. the human shoulder joint has two rotary degrees of freedom and can be considered as two links: one with a shoulder-to-elbow link length attached to an invisible link with a zero link length, each with a different rotation.

The Target frame can be specified relative to the Base frame of the robot using a standard 4x4 transformation matrix ${}^{B}\mathbf{T}_{T}$. Note that the $\underline{x}_T$, $\underline{y}_T$ and $\underline{z}_T$ unit vectors representing the basis axes of frame T relative to the base frame B are obtained from the first 3 vertical columns of the ${}^{B}\mathbf{T}_{T}$ "target" frame 4x4 matrix.

$$
{}^{B}\mathbf{T}_{T} = \begin{bmatrix} a_T & d_T & g_T & p_T \\ b_T & e_T & h_T & q_T \\ c_T & f_T & i_T & r_T \\ 0 & 0 & 0 & 1 \end{bmatrix}
= \begin{bmatrix} \underline{x}_T & \underline{y}_T & \underline{z}_T & {}^{B}\boldsymbol{t}_T \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{8}
$$

where the direction vector of the $\underline{x}_T$ basis axis with respect to frame B is given by
$\underline{x}_T = a_T\,\underline{x}_B + b_T\,\underline{y}_B + c_T\,\underline{z}_B$ and likewise
$\underline{y}_T = d_T\,\underline{x}_B + e_T\,\underline{y}_B + f_T\,\underline{z}_B$
$\underline{z}_T = g_T\,\underline{x}_B + h_T\,\underline{y}_B + i_T\,\underline{z}_B$

The point vector of the frame T origin point relative to the frame B origin is
${}^{B}\boldsymbol{t}_T = p_T\,\underline{x}_B + q_T\,\underline{y}_B + r_T\,\underline{z}_B$

Similarly, the $\underline{x}_E$, $\underline{y}_E$ and $\underline{z}_E$ unit vectors representing the basis axes of frame E relative to the base frame B are obtained from the first 3 columns of the ${}^{B}\mathbf{T}_{E}$ matrix which is the FK transformation matrix of the entire manipulator (similar to the type found in Eq. 4, obtained by combining all the A-matrices for the manipulator). The manipulator transform for an n-link manipulator is thus

${}^{B}\mathbf{T}_{E} = {}^{0}\mathrm{A}_{1}\ {}^{1}\mathrm{A}_{2}\ldots.\ {}^{n-1}\mathrm{A}_{n}$
(for any serial-link manipulator with n links ≥ 3)

$$
{}^{B}\mathbf{T}_{E} = \begin{bmatrix} a_E & d_E & g_E & p_E \\ b_E & e_E & h_E & q_E \\ c_E & f_E & i_E & r_E \\ 0 & 0 & 0 & 1 \end{bmatrix}
= \begin{bmatrix} \underline{x}_E & \underline{y}_E & \underline{z}_E & {}^{B}\boldsymbol{p}_E \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{9}
$$

where the direction vector of the $\underline{x}_E$ basis axis with respect to frame B is

$\underline{x}_E = a_E \underline{x}_B + b_E \underline{y}_B + c_E \underline{z}_B$ and likewise,
$\underline{y}_E = d_E \underline{x}_B + e_E \underline{y}_B + f_E \underline{z}_B$
$\underline{z}_E = g_E \underline{x}_B + h_E \underline{y}_B + i_E \underline{z}_B$

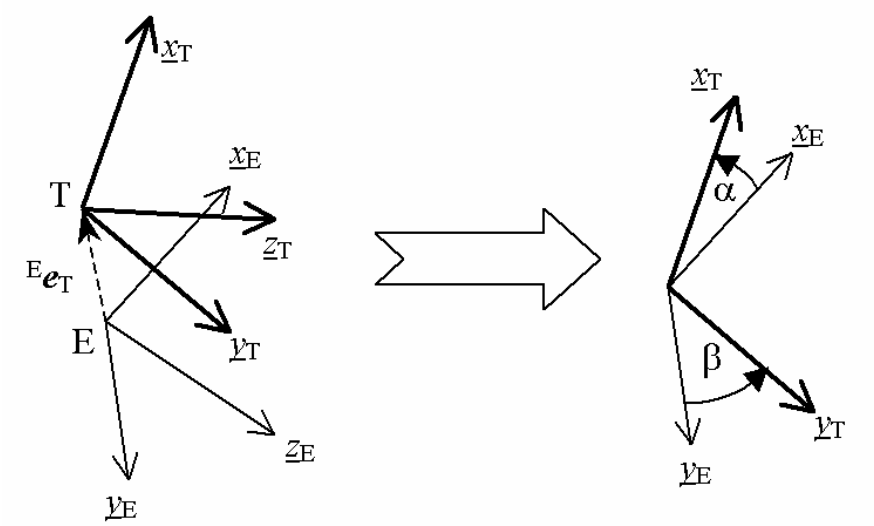

Fig. 6. Superimposing the origins of both E and T frames to measure α & β

The point vector of the frame E origin point relative to the frame B origin is

${}^B\boldsymbol{p}_E = p_E \underline{x}_B + q_E \underline{y}_B + r_E \underline{z}_B$

Fig. 6 shows the error vector and angular differences between the x and y basis axes of frames E and T, the magnitudes of which all need to be "driven towards 0" or reduced below an acceptable error tolerance in order to obtain an acceptable IK solution. In oder to achieve a suitable IK solution, the magnitude of the error vector $|{}^E\boldsymbol{e}_T|$ and the angles between pairs of corresponding $\underline{x}_E$, $\underline{x}_T$ and $\underline{y}_E$, $\underline{y}_T$ basis axes (ie. α & β respectively) must be calculated and then combined into a "total error" value $|e_{total}|$ which can be used to search for the best combination of joint variable changes. We may use the "Scalar" or "Dot Product" operation on basis vectors $\underline{x}_E$ and $\underline{x}_T$ to find the angle between them, α. Likewise, we can perform the same operation on vectors $\underline{y}_E$ and $\underline{y}_T$ to find β. There is no need to do this for the 3rd pair of axes, vectors $\underline{z}_E$ and $\underline{z}_T$, because if the other two axes line up, the z axes will automatically be aligned relative to the x-y planes because both frames are "right handed". The solutions for both α and β can each range anywhere from 0° to 180° . Also, the magnitude of any basis (unit) vector is 1, thus, $|\underline{x}_E| = |\underline{x}_T| = |\underline{y}_E| = |\underline{y}_T| = 1$. The inner or "Dot Product" operations are now used to find α and β.

$$\underline{x}_E \bullet \underline{x}_T = |\underline{x}_E|\,|\underline{x}_T| \cos\alpha = \cos\alpha \tag{10}$$

$$\underline{y}_E \bullet \underline{y}_T = |\underline{y}_E|\,|\underline{y}_T| \cos\beta = \cos\beta \tag{11}$$

It is useful to note that if α or β lies between 90° and 180°, the cosine function will return a negative value. The "worst case" alignment between any two basis axes vectors is 180°, which gives cos (180°) = -1. If α or β are between 0° and 90°, the cosine function will return a positive value. The "best case" for alignment between any two basis axis vectors is 0°, which gives cos(0°) = +1. Hence, angular alignment "error" between the $\underline{x}_E$ and $\underline{x}_T$ axes can be measured using a positive value like 1-cos (α). If the angle $\alpha = 0$, cos (0°) = 1, so the alignment error is 1-1 = 0 (meaning zero alignment error). If $\alpha = 180°$, cos (180°) = -1 so 1-(-1) = +2 which gives the largest or maximum value for alignment error. Note that "alignment error" is an artificial term that ranges from 0 (perfect alignment) to +2 (worst alignment) and is simply used as a measure of how poorly a pair of basis axes line up. We will designate e_{ax} as the angular alignment error between $\underline{x}_E$ and $\underline{x}_T$, and e_{ay} as the angular alignment error between $\underline{y}_E$ and $\underline{y}_T$. These values are useful for calculating an overall "total error" which also includes position error.

$$\begin{aligned} e_{ax} &= 1 - \cos\alpha = 1 - \underline{x}_E \bullet \underline{x}_T \\ &= 1 - (a_E\, a_T + b_E\, b_T + c_E\, c_T) \end{aligned} \tag{12}$$

$$\begin{aligned} e_{ay} &= 1 - \cos\beta = 1 - \underline{y}_E \bullet \underline{y}_T \\ &= 1 - (d_E\, d_T + e_E\, e_T + f_E\, f_T) \end{aligned} \tag{13}$$

An equation for "total error", e_{total}, can be created to combine the position error and angular alignment error terms so that the best combination of joint variable changes can be found to minimize this "total error". Total error can now be formulated using two "weighting factors" which can be adjusted to scale the importance of each source of error. K_p is the factor which adjusts the contribution of the initial error vector magnitude (or incremental step size to the next target position) $|^E\boldsymbol{e}_T|$ towards the "total error". K_a is the factor which adjusts the contribution of both e_{ax} and e_{ay} angular misalignment (error) values. These weighting factors are like "gains" for a PID algorithm, but they must always remain positive. ie. e_{total}, e_{ax} and e_{ay} are always positive. We will call e_{pos} the error term due to position error (distance between E and T) and e_{ang} will be the error term due to the sum of angular alignment error values.

$$e_{pos} = K_p\ |^E\boldsymbol{e}_T| \qquad (> 0 \text{ positive}) \tag{14}$$

$$e_{ang} = K_a\,(e_{ax} + e_{ay}) \qquad (> 0 \text{ positive}) \tag{15}$$

$$\begin{aligned} e_{total} &= e_{pos} + e_{ang} \\ &= K_p\ |^E\boldsymbol{e}_T| + K_a\,(e_{ax} + e_{ay}) \qquad (> 0) \end{aligned} \tag{16}$$

The values for K_p and K_a need to be adjusted so that a fair balance can be obtained between the contribution of position error and the contribution of axis misalignment errors. The worst value for e_{pos} should be equal to the worst case value of e_{ang} if position error is just as important as alignment error for the axes. Accuracy of the "Blind Search method" depends largely on the value of the error tolerance for an acceptable IK solution, however, higher precision solutions may require more iterations. The variable e_{total} can be used instead of a "test" error vector $|{}^{F}\boldsymbol{e}_{Tt}|$, which can be calculated for each and every possible combination of joint variable changes. (See Step 9 in Algorithm 1)

The same methods used in the previous section may be used to search for the best combination of joint variable changes which produce smaller e_{total} values as more iterations are executed. A smallest "total error" value can be found for each iteration using a minimum value search and this can be compared to the best "total error" found so far. The "Blind Search" algorithm searches for incremental IK solutions by reducing the "total error" e_{total} with more passes of the algorithm, until the e_{total} value is below a satisfactory error tolerance value, e_{tol} (or `etol`) for an acceptable FK and IK solution. Hence, an IK solution is found when $e_{total} \leq e_{tol}$. The error tolerance for the IK solution must be set by the programmer along with carefully selected values for search parameters like initial position displacement or step $|{}^{E}\boldsymbol{e}_{T}|$ and $\Delta\theta$ (or Δd for translational joints), but this can even be automated.

4 Conclusion

This paper has presented a practical and robust inverse kinematics method which can solve the joint variables for any type of serial-link robot arm or manipulator, regardless of the number and types of degrees of freedom the manipulator has or the number or location of workspace internal singularities inherent within the workspace of the robot. The "Blind Search" IK method described in this paper will search for small joint angle changes that are necessary to minimise the origin position error and/or alignment error between the axes of the End-effector frame and the Target frame. The speed, stability and reliability of "Blind Search" IK solutions depends heavily on the selection of suitable search parameters, such as step size to the next target point $|{}^{E}\mathbf{e}_{T}|$, $\Delta\theta$ or Δd incremental displacement magnitudes for each link, K_p and K_a "weighting factors" for calculating "total error", and an error tolerance e_{tol} defining the permissible tolerance or acceptable error of an IK solution. These variables need to be adjusted for each type of robot manipulator. Performance of these algorithms can be "tuned" by trial and error, or perhaps automatically, to achieve a satisfactory balance between solution accuracy, search stability (ie. reliability of convergence toward an acceptable solution), and computation time for real-time control. Much time and effort can be saved by using this "Blind Search" IK method because complicated mathematical derivations are avoided and only the FK solution or overall manipulator transform matrix is needed. The "Blind Search" method has been tested successfully for controlling the end-effectors of 3D simulated robot arms assembled from standard generic link types, as defined by McKerrow [2], without the need to derive any equations.

5 References

1. Paul R. P. (1981). *Robot manipulators – Mathematics, programming and control.* Massachusetts Institute of Technology USA, ISBN 0-262-16082-X
2. McKerrow P. J. (1991). *Introduction to Robotics.* (Chapters 3 and 4) Addison-Wesley, ISBN 0-201-18240-8
3. Klafter R. D., Chmielewski T. A., Negin M. (1989). *Robotic engineering – an integrated approach.* Prentice-Hall, ISBN 0-13-782053-4
4. Fu K. S., Gonzalez R. C., Lee C. S. G. *Robotics: Control, Sensing, Vision and Intelligence*. McGraw-Hill, ISBN 0-07-100421-1
5. Ranky P. G., Ho C. Y. (1985*). Robot modelling: control and applications with software.* IFS Publications Ltd. UK, ISBN 0-903608-72-3, and Springer-Verlag, ISBN 3-540-15373-X
6. Kohli D., Soni A. H. (1975). *Kinematic Analysis of Spatial Mechanisms via Successive Screw Displacements*. J. Engr. For Industry, Trans. ASME, vol. 2, series B pp. 739-747.
7. Denavit J. (1956). *Description and Displacement Analysis of Mechanisms Based on 2x2 Dual Matrices*. Ph.D thesis, Mechanical Eng'g, Northwestern University, Evanston, Ill.
8. Yang A. T., Frudenstein R. (1964). *Application of Dual Number Quaternian Algebra to the Analysis of Spatial Mechanisms*. Trans. ASME, Journal of Applied Mechanics, vol. 31, series E, pp. 152-157.
9. Uicker J. J. Jr, Denavit J., Hartenberg R. S. (1964). *An Iterative Method for the Displacement Analysis of Spatial Mechanisms.* Trans. ASME, Journal of Applied Mechanics, vol. 31, Sereies E, pp. 309-314
10. Milenkovic V., Huang B. (1983). *Kinematics of Major Robot Linkages*. Proc. 13th Intl. Symp. Industrial Robots, Chicago, Ill, pp. 16-31 to 16-47.
11. Lee C. S. G., Ziegler M. (1984). *A Geometric Approach in Solving the Inverse Kinematics of PUMA Robots*. IEEE Trans. Aerospace and Electronic System, vol. AES-20, No. 6, pp. 695-706.
12. Pieper D. L. (1968). *The Kinematics of Manipulators under Computer Control*. Artificial Intelligence Project Memo No. 72, Computer Science Department, Stanford University, Palo Alto, Calif., USA
13. Cubero S. N. (1997). *Force, Compliance and Position Control for a Pneumatic Quadruped Robot*. Ph.D Thesis, Faculty of Engineering, University of Southern Queensland, Australia.
14. Goldenberg A. A., Benhabib B., Fenton R. G. (1985). *A complete generalized solution to the inverse kinematics of robots*. IEEE Journal of Robotics and Automation, RA-1, 1, pp. 14-20.
15. Goldenberg A. A., Lawrence D. L. (1985). *A generalized solution to the inverse kinematics of redundant manipulators*. Journal of Dynamic Systems, Measurement and Control, ASME, 107, pp. 102-106.