Implementing HuPf Algorithm for the Inverse Kinematics of General 6R/P Manipulators

Jose Capco, Saraleen Mae Manongsong

Implementing HuPf Algorithm for the Inverse Kinematics of General 6R/P Manipulators

Jose Capco[1⋆] and Saraleen Mae Manongsong[2⋆⋆]

[1] Research for Symbolic Computation, Johannes Kepler University, Linz, Austria
jcapco@risc.jku.at
[2] Institute of Mathematics, University of the Philippines Diliman, 1101 Quezon City, Philippines
smmanongsong@gmail.com

Abstract. We reformulate and extend the HuPf algorithm (see [7]), which was originally designed for a general 6R manipulator (i.e. 6 jointed open serial chain/robot with only rotational joints), to solve the inverse kinematic (IK) problem of 6R/P manipulators (6-jointed open serial robot with joints that are either rotational or prismatic/translational). For the algorithm we identify the kinematic images of 3R/P chains with a quasi-projective variety in $\mathbb{P}^7$ via dual quaternions. More specifically, these kinematic images are projections of the intersection of a Segre variety with a linear 3-space to an open subset of $\mathbb{P}^7$ (identified with the special Euclidean group SE(3)). We show an easy and efficient algorithm to obtain the linear varieties associated to 3R/P subchains of a 6R/P manipulator. We provide examples showing the linear spaces for different 3R/P chains (a full list of them is available in an upcoming paper). Accompanying the extended HuPf algorithm we provide numerical examples showing real IK solutions to some 6R/P manipulators.

Keywords: Inverse Kinematics · Elimination Theory · Serial Manipulator

1 Introduction

The *Study quadric* is given by the $\sum_{i=0}^{3} x_i x_{i+4}$ for points

$$(x_0 : \cdots : x_7) \in \mathbb{P}^7(\mathbb{C})$$

There is a bijection between the special linear group SE(3) and a quasiprojective subset of the quadric via dual-quaternions. In this article we will give a rough idea what this means:

Let $\mathbb{H}$ be the classical (Hamiltonian) skew field of quaternions. One first shows a bijection between the quotient of multiplicative group $\mathbb{H}^*/\mathbb{R}^*$ and the special orthogonal group SO(3) (which is well-known, see [7,10]). The ring $\mathbb{H}[x]/\langle x^2\rangle$ (indeterminate x commuting with coefficients in $\mathbb{H}$) is called the ring of *dual quaternions* and denoted $\mathbb{D}$. The elements in $\mathbb{D}$ are often written as $p+\epsilon q$ where ϵ is the equivalence class of x and p, q are quaternions in $\mathbb{H}$ (we compare this construction to construction of *dual numbers* in algebraic number theory). Clearly $\mathbb{H}$ is a subring of $\mathbb{D}$ and there is an injection from the elements of SE(3) to the quotient of multiplicative group $\mathbb{D}^*/\mathbb{R}^*$ and it is well-defined if we know how pure translations and pure rotations are mapped. Pure rotations are mapped to $\mathbb{H}$ which is in $\mathbb{D}$. Pure translations are mapped via

$$\boldsymbol{t} \mapsto 0 + \epsilon\left(\frac{\boldsymbol{t}}{2}\right)$$

where $\boldsymbol{t} \in \mathbb{R}^3\backslash\{\boldsymbol{0}\}$ is a non-zero translation vector and we regard elements in $\mathbb{H}$ as 4-tuples ($\mathbb{H}$ is a four-dimensional $\mathbb{R}$-algebra). One shows that this fully defines a group homomorphism $\mathrm{SE}(3) \to \mathbb{D}^*$

⋆ Supported and funded by the Austrian Science Fund (FWF): Project P28349-N32 and W1214-N14 Project DK9

⋆⋆ Supported by the Office of the Chancellor of the University of the Philippines Diliman, through the Office of the Vice Chancellor for Research and Development, for funding support through the Outright Research Grant.

and if we compose it with the canonical quotient $\mathbb{D}^* \to \mathbb{D}^*/\mathbb{R}^*$ we even have an injective group homomorphism. Since $\mathbb{D}$ is an eight-dimensional $\mathbb{R}$-algebra we can identify elements in $\mathbb{D}^*/\mathbb{R}^*$ with points in a subset of $\mathbb{P}^7(\mathbb{R})$. In fact the image of the composed map we just described $\mathrm{SE}(3) \hookrightarrow \mathbb{D}^*/\mathbb{R}^*$ can be identified with points

$$\{(x_0 : \cdots : x_7) \in \mathbb{P}^7(\mathbb{R}) \,:\, x_0 \neq 0, \ldots, x_7 \neq 0 \text{ and } \sum_{i=0}^{3} x_i x_{i+4} = 0\}$$

which is a quasi-projective variety in $\mathbb{P}^7(\mathbb{R})$. For more details, we refer to [7,10] or our upcoming paper.

For engineering and applications, we deal with the real points of the Study quadric, but we also consider complex points because we discuss finiteness and existence of solutions to the inverse kinematic (IK) problem which involve some basic intersection theory where an algebraic closed base field (e.g. when using Hilbert's Nullstellensatz) is important. This allows us to give a general statement whether HuPf algorithm will work or not. We note that, we only assume that we are solving the inverse kinematic of regular values for a general 6R/P manipulator (finite solutions), complicated algorithms in real algebraic geometry like cylindrical algebraic decomposition (CAD, see [3]) to solve the real solutions is not necessary (there are at most 16 solutions, see [9]). Using CAD to describe real higher dimensional solutions (like the inverse kinematic of a redundant, e.g. 7 jointed, robot or the solutions within a kinematic singularity) is however very attractive but is beyond the scope of this work (we plan to investigate this in the future). Knowing if there are real solutions is usually done in the middle of the HuPf algorithm e.g. when solving for the roots of a resultant to find the coordinates of the middle link of a 6R/P-chain. Thus, our objective in this manuscript is threefold:

1. We describe the algorithm to compute all the parameterized hyperplanes needed for the HuPf algorithm for different permutations of 3R/P joint-types and for different parameterization of joints in the 3R/P sub-chains (as described in [9,7,8]).
2. We discuss efficient choice of parameterized linear spaces when computing IK. We also discuss preprocessing (e.g. the linear spaces for the left 3-chain can be fully preprocessed) to make real-time IK computation possible.
3. We discuss special cases, i.e. cases when, for a given 3R/P chain, all the parameterized linear spaces defined by each of the parameterizing joints are inside the Study quadric (i.e. the HuPf algorithm may fail because there are infinite solution to the IK problem).

We shall use the Denavit-Hartenberg (DH) convention when describing relationship between two frames. More precisely, the transformation between the frames (of joints) is given by the following rule:

- The z-axis of the reference frame will be the axis of rotation if the joint is revolute or the translational direction if the joint is prismatic
- To obtain the next frame, one starts with a rotation about the z-axis of the reference frame, called the *rotation*, followed by
- a translation along the z-axis of the reference frame, called the *offset*, followed by
- a translation along the x-axis, called the *distance*, followed by
- a rotation about the x-axis, called the *twist*

It is worth mentioning that, in order to solve IK from a system of polynomial equations, all rotations are parameterized by tangent of half-angles. In short, the transformation between frame i to frame $i+1$ is given by

$$R_z(v_i)T_z(d_i)T_x(a_i)R_x(l_i)$$

where R_z, T_z, T_x, R_x are rotations or translations with respect to z- or x-axis parameterized by tangent of half-angle rotation v_i, offset d_i, distance a_i and tangent of half-angle twist l_i of the i-th frame (i-th joint). More thorough discussion on DH-parameters and the DH-convention is given in [11].

In the first step of the HuPf algorithm, one wants to compute a set of at least four hyperplanes which describes a parameterized linear space whose projection to $\mathbb{P}^7$ contains the image of the kinematic map. The way to compute this for a 6R manipulator is discussed in [9]. We give this algorithm when prismatic joints are also involved:

Algorithm 1: Computing 3-chain hyperplane parameterized by joint $k \in \{1, 3\}$

Input: DH-parameter of a 3-chain, $k \in \{1, 3\}$, **boolean** *parameterized*
Output: Hyperplane of 2-chains (non-parameterized) or hyperplane pencils parameterized by joint k in $\mathbb{P}^7$ (3-chain)

Let ν_i, ν_j, ν_k be the three joint parameters with $i, j \in \{1, 2, 3\}$
Compute the forward kinematic image $\sigma = (s_0 : \cdots : s_7) \in \mathbb{P}^7$ of the 2-chain not containing k, with coordinates parameterized by ν_i, ν_j
Write each coordinate of σ as a 2-variate polynomial in $\mathbb{R}[\nu_i, \nu_j]$ with multi-degree at most $(1, 1)$
Create a 4×8 matrix M with l-th column having entries which are coefficients of $1, \nu_i, \nu_j, \nu_i\nu_j$ respectively in s_l
Compute the null space of the matrix M spanned by linearly independent vectors $w_1, \ldots, w_m$ each of length 8
if not *parameterized* **then**
    return the linear forms $H_l : \sum_{n=0}^{7} w_{ln} x_n \in \mathbb{R}[x_0, \ldots, x_7]$, where w_{ln} is the n-th coordinate of a representative of w_l, with $l = 1, \ldots, m$
end
else
    if $k = 1$ **then**
        Regard $(x_0 : \cdots : x_7)$ as a dual quaternion and pre-multiply it with a dual quaternion defined by motion of ν_1 to get $(x'_0 : \cdots : x'_7)$
    end
    else
        Regard $(x_0 : \cdots : x_7)$ as a dual quaternion and post-multiply it with a dual quaternion defined by motion of ν_3 to get $(x'_0 : \cdots : x'_7)$
    end
    return the linear forms $H_l : \sum_{n=0}^{7} w_{ln} x'_n(x_0, \ldots, x_7) \in \mathbb{R}[x_0, \ldots, x_7]$, with $l = 1, \ldots, m$
end

In Algorithm 1 (and further algorithms), the joint parameter ν_i is d_i if the i-th joint is prismatic otherwise it is v_i (i.e. if the i-th joint is revolute).

Algorithm 2: Computing 3-chain hyperplane parameterized by second joint

Input: DH-parameter of a 3-chain
Output: Hyperplane parameterized by 2nd joint in $\mathbb{P}^7$ corresponding to 3-chain motion

Let $\boldsymbol{\nu} := (\nu_1, \nu_2, \nu_3)$ be the three joint parameters
Compute the forward kinematic $\sigma = (s_0 : \cdots : s_7) \in \mathbb{P}^7$ of the 3-chain with coordinates parameterized by all the joint parameters
Write each coordinate of σ as a 3-variate polynomial $\mathbb{R}[\nu_1, \nu_2, \nu_3]$ each of multi-degree at most $(1, 1, 1)$
Create an empty 12×16 matrix M
Set the $l \leq 8$ column of M have entries which are coefficients of $\boldsymbol{\nu}^\alpha$, where $\alpha \leq (1, 1, 2)$, of s_l
Set the $l > 8$ columns of M have entries which are coefficients of $\boldsymbol{\nu}^\alpha$, where $\alpha \leq (1, 1, 2)$, of $\nu_2 s_l$
Compute the null space of the matrix M spanned by vectors $w_1, \ldots, w_m$ each of length 16
Choose independent linear forms from the linear forms $H_l : \sum_{n=0}^{7} (w_{l,n} + w_{l,n+8}\nu_2) x_n$ with $l = 1, \ldots, m$
return these independent linear forms

To compute the special cases, i.e. case for which hyperplane parameterized by any of the joints lie inside the Study quadric, one follows almost the same procedure as Algorithm 1 and 2. However,

since now the hyperplane may not lie in general position (i.e. the DH-parameters may be very specific for them to all lie in the Study quadric), care must be taken when computing the null space of the coefficient matrices.

2 The Hyperplanes

For an efficient algorithm it is vital to choose the linear space that is described by linear forms with least complexity. Usually the linear space parameterized by the second joint is the most complex case. So one first computes linear space parameterized by the first joint and look for (DH parameter) conditions for which this linear space may lie in the Study quadric (say for RRR, this condition is when the twist half-angle tangent satisfies $l_2 = 0$ or the offset satisfies $a_2 = 0$). In the case that the linear space parameterized by ν_1 (see Algorithm 1) cannot be chosen, we look at the third joint and immediately apply the condition (so in our example if $l_2 = 0$ or $a_2 = 0$ then we immediately apply this in the set of equations), we then use Algorithm 1 to find the hyperplanes parameterized by ν_3. Finally if for this case the linear space lies in the Study quadric (in our example, this would be either $a_1 = 0$ or $l_1 = 0$ with the additional condition from the first investigation that $a_2 = 0$ or $l_2 = 0$), we immediately apply these conditions and look at Algorithm 1 to get hyperplane parameterized by ν_2. To summarize, the hyperplanes we provide cannot be used in general case but in cases of increasing level of complexity. One should use these steps in order when chosing the computed linear spaces in our work:

1. Check if $T(\nu_1)$ is applicable (for RRR we check if $a_2 \neq 0$ and $l_2 \neq 0$)
2. If not, check if $T(\nu_3)$ is applicable (for RRR, we check if $(a_2 = 0 \vee l_2 = 0) \wedge (a_1 \neq 0 \wedge l_2 \neq 0)$)
3. If not, supposing we are not in a *special case* (i.e. kinematic image of the 3-chain does not describe a planar, pure translation or spherical motion) then we use $T(\nu_2)$.

Because of the level of complexity of the equations describing the general $T(\nu_2)$, inverse kinematic computation using the algorithm will be much slower. So one tries to avoid this if either $T(\nu_1)$ or $T(\nu_3)$ is possible. Due to the length of the equations, we do not show all of the simplified $T(\nu_2)$ for all 3R/P chains. But we can show for all the subcases in the 3R-chain (for $T(\nu_1)$ and $T(\nu_3)$ one can look at other literatures which will have them in detail).

For the 3R/P chain these are the conditions to not choose $T(\nu_1)$ (in Step 1. above) respectively $T(\nu_3)$ (in Step 2. above, assuming we cannot choose $T(\nu_1)$):

3R/P	✗$T(\nu_1)$	✗$T(\nu_3)$
RRP	$l_2 = \pm 1$	$a_1 = 0$ or $l_1 = 0$
RPR	$l_2 = \pm 1$	$l_1 = \pm 1$
RPP[1]	All	$l_1 = \pm 1$
PRR	$a_2 = 0$ or $l_2 = 0$	$l_1 = \pm 1$
PRP	$l_2 = \pm 1$	$l_1 = \pm 1$
PPR[1]	$l_2 = \pm 1$	All
RRR	$a_2 = 0$ or $l_2 = 0$	$a_1 = 0$ or $l_1 = 0$

Table 1: DH conditions for not choosing the linear space. For a given 3R/P if the condition in column 2 is satisfied, we have to look at column 3 and if that is also satisfied we may possibly choose $T(\nu_2)$

Notice in the above table we disregarded PPP because this 3-chain describes a purely translational motion so its kinematic image lies in a 3-space living in the Study quadric. In fact, increasing prismatic joints should theoretically make it easier for us to compute inverse kinematics.

[1] we exclude the case that the two consuctive prismatic joints allow movement in the same direction (i.e. twist angle is 0). This is a degenerate case that is not interesting and in this case a solution to the IK problem imply infinite solutions so that classical HuPf algorithm is not applicable.

Here we show simplified parameterized (by a selected joint parameter) linear spaces of some 3R/P chains. By *simplified* we mean the following: the coordinates $(x_0 : \cdots : y_3)$ are given up to post- or pre- multiplication by some fixed element in SE(3). For instance with the DH-parameter the kinematic image of an 3R chain parameterized by v_1 is given by:

$$R_z(v_1)T_z(d_1)T_x(a_1)R_x(l_1)M$$

where M is the kinematic image of a 2R-chain. But we can simply commute R_z and T_z and consider the image of

$$R_z(v_1)T_x(a_1)R_x(l_1)M$$

reducing the number of variables so we are able to display simpler linear forms (the actual hyperplane can be obtained by an easy transformation, in this case by a premultiplication of $T_z(d_1)$). Though, in all our cases, we assume for brevity that d_1 is always 0 (otherwise another simple transformation will yield the inverse kinematic).

Often $T(\nu_2)$ is very complicated and involves many subcases (in order to improve efficiency in the `C++` implementation and the algorithm). So we will only show this in RRR and RRP case. For RRR the linear spaces $T(v_1)$ and $T(v_3)$ are well-studied (see [9,7,8]) so we will not show this.

2.1 RRR Hyperplanes, $T(v_2)$

For $T(v_2)$ with $[a_1, a_2] = [0, 0]$:

$$\begin{aligned}
H_1 &: d_2l_1l_2v_2x_3 + d_2l_1l_2x_0 - d_2v_2x_3 + d_2x_0 - 2l_1l_2v_2x_4 +_2 l_1l_2x_7 - 2v_2x_4 - 2x_7\\
H_2 &: d_2l_1v_2x_2 + d_2l_1x_1 + d_2v_2x_2l_2 - d_2x_1l_2 - 2l_1v_2x_5 +_2 l_1x_6 +_2 v_2l_2x_5 +_2 l_2x_6\\
H_3 &: -d_2l_1v_2x_1 + d_2l_1x_2 - d_2v_2x_1l_2 - d_2x_2l_2 - 2l_1v_2x_6 - 2l_1x_5 +_2 v_2l_2x_6 - 2l_2x_5\\
H_4 &: -d_2l_1l_2v_2x_0 + d_2l_1l_2x_3 + d_2v_2x_0 + d_2x_3 - 2l_1l_2v_2x_7 - 2l_1l_2x_4 - 2v_2x_7 +_2 x_4
\end{aligned}$$

For $T(v_2)$ with $[a_1, l_2] = [0, 0]$:

$$\begin{aligned}
H_1 &: a_2l_1v_2x_0 - a_2l_1x_3 - v_2x_3d_2 - 2v_2x_4 + x_0d_2 - 2x_7\\
H_2 &: -a_2v_2x_1 - a_2x_2 + v_2x_2d_2l_1 - 2v_2l_1x_5 + x_1d_2l_1 +_2 l_1x_6\\
H_3 &: -a_2v_2x_2 + a_2x_1 - v_2x_1d_2l_1 - 2v_2l_1x_6 + x_2d_2l_1 - 2l_1x_5\\
H_4 &: a_2l_1v_2x_3 + a_2l_1x_0 + v_2x_0d_2 - 2v_2x_7 + x_3d_2 +_2 x_4
\end{aligned}$$

For $T(v_2)$ with $[l_1, a_2] = [0, 0]$:

$$\begin{aligned}
H_1 &: a_1l_2v_2x_0 - a_1l_2x_3 - v_2x_3d_2 - 2v_2x_4 + x_0d_2 - 2x_7\\
H_2 &: -a_1v_2x_1 + a_1x_2 - v_2x_2d_2l_2 - 2v_2l_2x_5 + x_1d_2l_2 - 2l_2x_6\\
H_3 &: -a_1v_2x_2 - a_1x_1 + v_2x_1d_2l_2 - 2v_2l_2x_6 + x_2d_2l_2 +_2 l_2x_5\\
H_4 &: a_1l_2v_2x_3 + a_1l_2x_0 + v_2x_0d_2 - 2v_2x_7 + x_3d_2 +_2 x_4
\end{aligned}$$

For $T(v_2)$ with $[l_1, l_2] = [0, 0]$:

$$\begin{aligned}
H_1 &: -x_1\\
H_2 &: -x_2\\
H_3 &: -d_2x_3 - 2x_4\\
H_4 &: d_2x_0 - 2x_7
\end{aligned}$$

2.2 RRP Hyperplanes

For $T(v_1)$:

$$
\begin{aligned}
H_1 &: l_2x_0 - x_1 \\
H_2 &: -l_2x_3 + x_2 \\
H_3 &: a_2l_2^2x_0 - a_2x_0 - 2l_2x_4 - 2x_5 \\
H_4 &: a_2l_2^2x_2 - a_2x_2 - 2l_2^2x_7 - 2l_2x_6
\end{aligned}
$$

For $T(d_3)$:

$$
\begin{aligned}
H_1 &: a_1l_1x_0 - 2x_4 \\
H_2 &: -a_1x_1 - 2l_1x_5 \\
H_3 &: -a_1x_2 - 2l_1x_6 \\
H_4 &: a_1l_1x_3 - 2x_7
\end{aligned}
$$

For $T(v_2)$ with $[a_1, l_2] = [0, 1]$:

$$
\begin{aligned}
H_1 &: -l_1v_2x_0 - l_1v_2x_1 + l_1x_2 + l_1x_3 + v_2x_0 - v_2x_1 - x_2 + x_3 \\
H_2 &: -l_1v_2x_2 - l_1v_2x_3 - l_1x_0 - l_1x_1 - v_2x_2 + v_2x_3 - x_0 + x_1 \\
H_3 &: 2a_2l_1^2v_2x_0 - 2a_2l_1^2x_2 - 2a_2l_1v_2x_0 + 2a_2l_1x_2 + l_1^3v_2x_2d_2 + l_1^3v_2x_4 - l_1^3v_2x_5 + l_1^3d_2x_1 + l_1^3x_6 \\
&\quad - l_1^3x_7 + l_1^2v_2x_2d_2 - l_1^2v_2x_4 - l_1^2v_2x_5 + 2l_1^2x_0d_2 + l_1^2d_2x_1 - l_1^2x_6 - l_1^2x_7 + l_1v_2x_2d_2 - l_1v_2x_4 \\
&\quad + l_1v_2x_5 + 2l_1x_0d_2 - l_1d_2x_1 - l_1x_6 + l_1x_7 + v_2x_2d_2 + v_2x_4 + v_2x_5 - d_2x_1 + x_6 + x_7 \\
H_4 &: 2a_2l_1^2v_2x_2 + 2a_2l_1^2x_0 + 2a_2l_1v_2x_2 + 2a_2l_1x_0 - l_1^3v_2x_0d_2 + l_1^3v_2x_6 - l_1^3v_2x_7 + l_1^3d_2x_3 - l_1^3x_4 \\
&\quad + l_1^3x_5 + l_1^2v_2x_0d_2 + l_1^2v_2x_6 + l_1^2v_2x_7 - 2l_1^2x_2d_2 - l_1^2d_2x_3 - l_1^2x_4 - l_1^2x_5 - l_1v_2x_0d_2 - l_1v_2x_6 \\
&\quad + l_1v_2x_7 + 2l_1x_2d_2 - l_1d_2x_3 + l_1x_4 - l_1x_5 + v_2x_0d_2 - v_2x_6 - v_2x_7 + d_2x_3 + x_4 + x_5
\end{aligned}
$$

For $T(v_2)$ with $[a_1, l_2] = [0, -1]$:

$$
\begin{aligned}
H_1 &: l_1v_2x_0 - l_1v_2x_1 + l_1x_2 - l_1x_3 + v_2x_0 + v_2x_1 + x_2 + x_3 \\
H_2 &: l_1v_2x_2 - l_1v_2x_3 - l_1x_0 + l_1x_1 - v_2x_2 - v_2x_3 + x_0 + x_1 \\
H_3 &: -2a_2l_1^2v_2x_0 - 2a_2l_1^2x_2 - 2a_2l_1v_2x_0 - 2a_2l_1x_2 + l_1^3v_2x_2d_2 - l_1^3v_2x_4 - l_1^3v_2x_5 + l_1^3d_2x_1 + l_1^3x_6 \\
&\quad + l_1^3x_7 - l_1^2v_2x_2d_2 - l_1^2v_2x_4 + l_1^2v_2x_5 + 2l_1^2x_0d_2 - l_1^2d_2x_1 + l_1^2x_6 - l_1^2x_7 + l_1v_2x_2d_2 + l_1v_2x_4 \\
&\quad + l_1v_2x_5 - 2l_1x_0d_2 - l_1d_2x_1 - l_1x_6 - l_1x_7 - v_2x_2d_2 + v_2x_4 - v_2x_5 + d_2x_1 - x_6 + x_7 \\
H_4 &: -2a_2l_1^2v_2x_2 + 2a_2l_1^2x_0 + 2a_2l_1v_2x_2 - 2a_2l_1x_0 - l_1^3v_2x_0d_2 - l_1^3v_2x_6 - l_1^3v_2x_7 + l_1^3d_2x_3 - l_1^3x_4 \\
&\quad - l_1^3x_5 - l_1^2v_2x_0d_2 + l_1^2v_2x_6 - l_1^2v_2x_7 - 2l_1^2x_2d_2 + l_1^2d_2x_3 + l_1^2x_4 - l_1^2x_5 - l_1v_2x_0d_2 + l_1v_2x_6 \\
&\quad + l_1v_2x_7 - 2l_1x_2d_2 - l_1d_2x_3 + l_1x_4 + l_1x_5 - v_2x_0d_2 - v_2x_6 + v_2x_7 - d_2x_3 - x_4 + x_5
\end{aligned}
$$

For $T(v_2)$ with $[l_1, l_2] = [0, 1]$:

$$
\begin{aligned}
H_1 &: x_0 - x_1 \\
H_2 &: x_2 - x_3 \\
H_3 &: -d_2x_2 - x_4 - x_5 \\
H_4 &: d_2x_0 - x_6 - x_7
\end{aligned}
$$

For $T(v_2)$ with $[l_1, l_2] = [0, -1]$:

$$H_1 : -x_0 - x_1$$
$$H_2 : -x_2 - x_3$$
$$H_3 : -d_2x_2 + x_4 - x_5$$
$$H_4 : d_2x_0 + x_6 - x_7$$

The other linear forms for other 3-chain joint types can be similarly computed. They (esp. RPR and PRR chains) can found in [1] and in an upcoming paper.

2.3 The Right Chain

We have so far displayed linear forms describing the linear space for the left chain. Our point of reference is the base frame. Hyperplane from parameters of right chain can also be computed using the following algorithm:

Algorithm 3: Computing hyperplanes parameterized by a joint in the right reversed 3-chain

Input: DH-parameter of the right 3-chain, end-effector transformation τ, $T(\nu_i)$
Output: $T(\nu_{7-i})$ described by DH-parameter of the right-chain

1 Make the following substitutions in the equations of $T(\nu_i)$ (say for RRP):

$$[a_1, a_2, a_3, l_1, l_2, l_3, v_1, v_2, v_3, d_1, d_2, d_3] \to [-a_5, -a_4, 0, -l_5, -l_4, 0, -v_6, -v_5, -v_4, -d_6, -d_5, -d_4]$$

2 Replace $\sigma := (x_0 : \cdots : x_7)$ with $\tau\sigma$
3 **return** new linear forms describing $T(\nu_{7-i})$ having reversed joint-types (say for PRR)

Due to lack of space, we will not show the linear forms for the right-chain. This is available in the dataset [1]. In [2] we also include a Giac implementation of the algorithms (we us the giacpy python wrapper of Giac, see [5,6]).

Clearly, in the algorithms presented the computation of resultant is the 'bottleneck' (the resultant that one compute is that of two bivariate polynomials, with maximum total degree 14). However, we do not focus on complexity analysis because neither the degree (at most 14) nor the number of variables (two) will vary when computing the inverse kinematics of a general 6R/P manipulator using the HuPf algorithm. Moreover, for the magnitude of our problem even a naive resultant computation (e.g. using Sylvester matrix) would suffice and be fast enough. In fact, HuPf runtime is fast. For instance, the maximum runtime for solving one inverse kinematic query (`C++` parallelized implementation) using HuPf in an Intel Core i5-6200U processor is 35ms. This is a tolerable number even by industrial standards. However in the future, we plan to study an extended version of HuPf algorithm devised to solve inverse kinematics of redundant manipulators where we will need to focus on time complexity as joint number varies.

3 The Inverse Kinematic Algorithm with an Example

Finally we can show HuPf algorithm for solving inverse kinematics of general 6R/P manipulators. We assume that the end-effector pose is reachable (i.e. the inverse kinematics has a real solution) and that the IK solutions are finite (for 6-jointed manipulators we only want solutions of regular values).

Algorithm 4: (HuPf) Computing IK of a general 6R/P manipulator for a general pose

Input: A reachable end-effector pose $\sigma \in \mathrm{SE}(3)$ whose IK solution is finite. Parameterized linear spaces for the left-chain (see Algorithms 1 and 2) and right-chain (dependent on σ, see Algorithm 3) by joint-parameters μ and ν.

Output: IK solutions to σ

1 Let $\mathcal{L} := \{l_i\}_{i=1}^{8}$ be the linear forms describing the two linear spaces in the input
2 **for** $j = 1, \ldots, 8$ **do**
3 | Solve for $(x_0 : \cdots : x_7) \in \mathbb{P}^7(\mathbb{C}(\mu, \nu))$ satisfying all linear forms in $\mathcal{L} \backslash \{l_j\}$
4 | **if** $x_0 \neq 0$ *or* $x_1 \neq 0$ *or* $x_2 \neq 0$ *or* $x_3 \neq 0$ **then** break
5 **end**
6 Without loss of generality we may assume $x_0, \ldots, x_7 \in \mathbb{C}[\mu, \nu]$
7 Substitute $x_0, \ldots, x_7$ into l_j to obtain a polynomial $f(\mu, \nu) \in \mathbb{C}[\mu, \nu] \backslash \mathbb{C}$
8 Set $g := \sum_{i=0}^{3} x_i x_{i+3}$ which is generally a polynomial in $\mathbb{C}[\mu, \nu] \backslash \mathbb{C}$
9 Common zeros of f and g are computed via resultant and elimination theory
10 **foreach** *common zero* (μ', ν') *of* f *and* g **do**
11 | Let $x_0', \ldots, x_7'$ be the evaluations of $x_0, \ldots, x_7$ (pose of the middle link)
12 | **foreach** *joint* λ *not* μ *and* ν **do**
13 | | There is a unique solution of λ via backsubstition of $x_0', \ldots, x_7'$ into a linear form paramaterized by λ from Algorithms 1, 2 and 3
14 | **end**
15 **end**
16 **return** all joint values

One can show, with assumption that the input is a reachable end-effector pose with finite IK solution, the algorithm ends successfully. This reasoning is also used to prove that the for loop in Line 2 will break successfully with one of the x_0, x_1, x_3, x_4 non-zero. Finally the finiteness of the IK solution will also guarantee us that f and g in Lines 7 and 8 are not identically 0 (it is not a constant because we have a solution to the IK problem).

We now show an example of an IK problem that we solve using Algorithm 4. Consider a 2R2P2R manipulator (i.e. a serial manipulator consisting of first two joints that are revloute, third and fourth joints that are prismatic and last two joints that are revolute) with DH-parameters given in the table below.

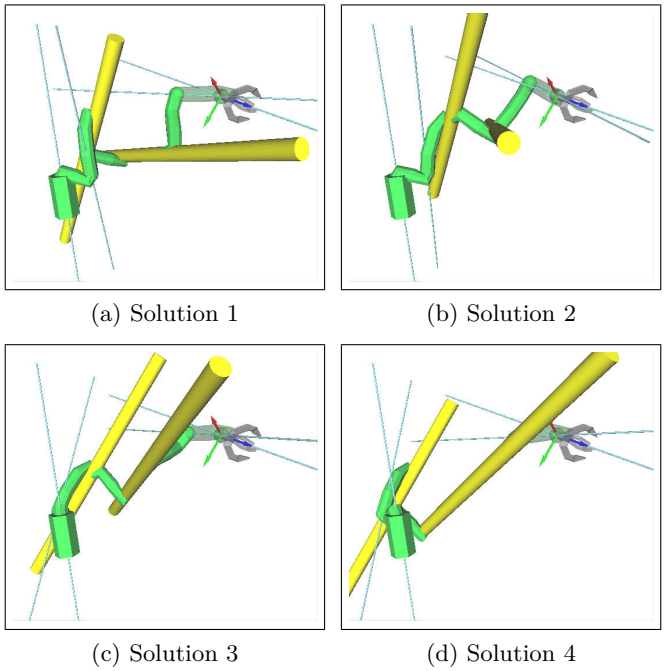
If we apply Algorithm 4 we obtain 12 solutions to the IK problem for a generic pose of the end-effector. In our case we chose a pose given by joint values $10, 30, 0.1, -0.1, 31, 55$ (revolute joint values given in degree) or $0.0875, 0.1763, 0.1, -0.1, 0.2773, 0.5206$ (revolute joint values given as tangent of half-angles). Generally only 4 of these solutions are real, the real solutions are given in the table below and it is illustrated in the figure below

i	v_i	d_i	a_i	l_i
1	*	0	0.2	0.2035
2	*	0.3	0.2	0.2035
3	-0.4142	*	0.3	0.4142
4	0.7133	*	0.4	0.3153
5	*	0.3	0	0.1763
6	*	0	0	0

Table 2: DH parameters for a 2R2P2R manipulator in our example. Parameters involving twist or rotation are tangent of half-angles.

	Solution 1	Solution 2	Solution 3	Solution 4
v_1	-0.0374	0.0875	1.2875	2.0551
v_2	0.7075	0.1763	-0.6262	-0.7273
d_3	-0.3786	0.1	0.0119	-0.3336
d_4	0.6391	-0.1	0.5796	1.03798
v_5	-1.4516	0.2773	0.0878	-0.3247
v_6	-9.3357	0.5206	1.3693	5.1233

Table 3: Real inverse kinematics solutions to the given 2RP3R manipulator. Revolute joint values are given in tangent of half-angles.



(a) Solution 1 (b) Solution 2

(c) Solution 3 (d) Solution 4

Fig. 1: Real inverse kinematic solutions to a certain end-effector pose of the 2R2P2R chain given in Table 2. The (light-blue) lines correspond to axis of rotations of the revolute joints and the (yellow) thick rods are the direction where the prismatic joint translates the links.

References

1. **J. Capco, S.M. Manongsong**, *Linear Spaces Associated to 3R/P Kinematic Image [Data set].* Zenodo 2019. doi: 10.5281/zenodo.3147394
2. **J. Capco, S.M. Manongsong**, *Code: Implementing HuPf Algorithm for the inverse Kinematics of General 6R/P Manipulators.* Zenodo 2019. doi: 10.5281/zenodo.3157441
3. **G.E. Collins**, *Quantifier Elimination by Cylindrical Algebraic Decomposition – Twenty Years of Progress.* In: Quantifier Elimination and Cylindrical Algebraic Decomposition (Editor: B.F. Caviness, J.R. Johnson), Springer-Verlag 1998, p.8–23
4. **D. Cox, J. Little, D. O'Shea**, *Ideals, Varieties and Algorithms*, 3rd Edition, Springer 2007.
5. **B. Parisse, R. De Graeve**, *Giac/Xcas.*
   Online: https://www-fourier.ujf-grenoble.fr/~parisse/giac.html, Accessed: 02.2019
6. **F. Han**, *giacpy.*
   Online: https://gitlab.math.univ-paris-diderot.fr/han/giacpy, Accessed: 02.2019
7. **M. Husty, M. Pfurner, H.-P. Schröcker**, *A new and efficient algorithm for the inverse kinematics of a general serial 6R manipulator*, Mech. Mach. Theory 2007, Vol. 42, p.66–81
8. **M. Husty, H.-P. Schröcker**, *Kinematics and Algebraic Geometry*, 21st Century Kinematics (Editor: J.M. McCarthy), Springer 2012, p. 85–123.
9. **M. Pfurner**, *Analysis of spatial serial Manipulators using kinematic mapping*, Doctoral Thesis, Institute for Basic Sciences in Engineering, Unit Geometry and CAD, University of Innsbruck, Oct. 2006
10. **J.M. Selig**, *Geometric Fundamentals of Robotics*, Monographs in Computer Science (Ed.: D. Gries, F.B. Schneider), 2nd Edition, Springer 2005.
11. **M.W. Spong, S. Hutchinson, M. Vidyasagar**, *Robot Modeling and Control*, John Wiley & Sons 2005.