Adv. Appl. Clifford Algebras 27 (2017), 2067–2082
© 2016 Springer International Publishing
0188-7009/032067-16
published online June 23, 2016
DOI 10.1007/s00006-016-0698-2

Advances in Applied Clifford Algebras

Efficient Implementation of Inverse Kinematics on a 6-DOF Industrial Robot using Conformal Geometric Algebra

Sondre Sanden Tørdal*, Geir Hovland, and Ilya Tyapin

Abstract. This paper presents an implementation of the inverse kinematics (IK) solution for an industrial robot based on Conformal Geometric Algebra where the correct signs of the joint angles are extracted using the multivector coefficients and applying the forward kinematics. The solution presented is twice as fast as traditional IK algorithms implemented using matrix algebra, and more than 45 times faster than the IK provided by the robot manufacturer. The proposed solution has been successfully demonstrated and benchmarked in a 3-DOF motion compensation experiment. In addition to being efficient the presented solution requires less matrix operations than for the traditional IK.

Keywords. Inverse kinematics, Industrial robotics, Conformal geometric algebra, Motion compensation.

1. Introduction

Existing offshore applications may include real-time dependent applications such as motion compensated equipment. The real-time equipment has to be functional and safely operated even during extreme weather conditions. This is one of the key motivations why robust and reliable programmable logic controllers (PLCs) currently dominate the offshore industry. In addition, offshore companies possess excellent knowledge and experience using PLCs to control their equipment. PLCs are in general a robust, reliable and real-time control platform to develop control strategies. However, a drawback using the PLC is the computational performance (typically 10–20 ms cycle times), and the low-level programming languages which do not provide functionality to handle matrices directly for instance. PLCs in offshore applications

*Corresponding author.
The research presented in this article has received funding from the Norwegian Research Council, SFI Offshore Mechatronics, Project Number 237896.

are normally shared between different applications, not only kinematics calculations. Since these PLCs often reach their computational limit, it is in general important to use efficient code. As the offshore industry is currently experiencing difficulties related to low oil prices, profitable oil production is of high importance. To decrease oil production costs, a higher level of complex automation is required. Fast, intuitive and robust algorithms are one important step closer to achieving this goal.

In almost all automation engineering tasks including robotic equipment, the problem of solving the inverse kinematics (IK) of complex geometries are required. If the computational performance is limited, slow IK algorithms, dynamics and path planning, will drastically reduce the overall performance due to the time lag between each sampling interval. If the robot/equipment to be controlled is placed inside a factory, this problem can be solved by using powerful computers. On the other hand, if situated in harsh offshore conditions the computational performance may be limited due to less powerful control platforms like PLCs for instance. Current methods to solve the IK problem are in general performed by analyzing the Forward Kinematics (FK) and then finding suitable relationships to solve the joint angles for a given end-effector position and orientation [12]. IK methods like Pieper's solution [10] which was presented already in the late 1960s is not possible to implement directly on a PLC without preprocessing the matrix equations into ordinary equations. This is true since a matrix/vector library is normally not covered by the IEC61131-3 standard. In [8] an example was presented where matrix operations were preprocessed to obtain regular equations. The equations generated from conformal geometric algebra (CGA) do not require matrix operations in the first place, and could be implemented more or less directly on standard PLCs.

William Kingdom Clifford first introduced geometric algebra in the 1870s. Clifford algebra was at this time building on the earlier work of Hamilton and Grassmann. This mathematical theory has not frequently been used in engineering applications, mostly due to the lack of tools to understand and implement the algorithms. This issue has more or less been solved lately due to newly developed visualization software like CluViz (see [9]). The developed code in CluViz can easily be compiled to optimized C++ code by using Hildenbrand's Gaalop C++ compiler [2,4,7]. The specific task in this paper is aimed at solving the IK problem of an industrial robot using CGA as a mathematical framework, there is some previously published work which are presenting IK algorithms using CGA, please see [1,3,5,6,11]. The work presented in this paper differs from the previously published work with respect to the following: The proposed CGA solution is compared with traditional IK methods using DH-parameters and geometric decoupling, the signs/orientations of the four first angles are obtained using the CGA multivector components, the last two orientations are found by applying the FK. The industrial robot (Comau Smart-5 NJ-110 3.0) considered in this paper has two joint offsets (a_1 and a_3 in Fig. 1), which increase slightly the kinematic complexity compared to previous research work. As a bottom line, this paper presents a traditional IK algorithm and a CGA-based IK algorithm describing

an industrial 6-DOF robot with two joint offsets. The two IK algorithms are compared in terms of computational speed and intuitiveness of the algorithm design process.

The main goal of the research presented in this paper is not to solve the IK for very challenging machines, but rather prepare the basis for computationally efficient multiple-machine control and relative motion compensation with limited computational resources (such as industrial PLCs). The current paper considers only a single robot, but for the authors, it is the first step in this direction.

2. Traditional IK Solution using DH-Parameters and Geometric Decoupling

The traditional IK solution is divided in five sub-solutions: describing the FK using DH-parameters, calculation of the wrist center point (see. Fig. 1), geometric decoupling to solve the first three joint angles $(\theta_1, \theta_2, \theta_3)$, apply the FK to obtain the orientation matrix R_6^3, and design a proper algorithm to choose among the different set of angles $(\theta_4, \theta_5, \theta_6)$ obtained from R_6^3. An overview of the robot, link lengths and coordinate frames is shown in Fig. 1.

2.1. Forward Kinematics

The DH convention is a 4×4 homogeneous transformation matrix A_i using four parameters to describe a rigid motion from joint $(i-1)$ to joint (i). The DH transformation matrix A_i is given by Eq. (2.1).

$$A_i = \begin{bmatrix} c_{\theta_i} & -s_{\theta_i} c_{\alpha_i} & s_{\theta_i} s_{\alpha_i} & a_i c_{\theta_i} \\ s_{\theta_i} & c_{\theta_i} c_{\alpha_i} & -c_{\theta_i} s_{\alpha_i} & a_i s_{\theta_i} \\ 0 & s_{\alpha_i} & c_{\alpha_i} & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.1}$$

where θ_i is the revolution angle of joint i, α_i is the link twist, a_i is the joint offset, and d_i is the link length. In matrix A_i, c_k and s_k are abbreviations for cosine and sine expressions, where the subscript k defines the respective angle argument. These four DH-parameters are usually structured in a DH-table as given in Table 1.

The DH table lays the foundation for understand and solve the IK problem. By using Eq. (2.1) and the parameters described in Table 1, a complete homogeneous transformation between coordinate frame 0 and the end effector's coordinate frame 6 is given by Eq. (2.2).

$$T_6^0(\theta_1, \ldots, \theta_6) = \prod_{i=1}^{6} A_i = \begin{bmatrix} r_{11} & r_{12} & r_{13} & x \\ r_{21} & r_{22} & r_{23} & y \\ r_{31} & r_{32} & r_{33} & z \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.2}$$

where transformation matrix T_6^0, consists of the rotation matrix R_6^0's scalar components r_{ij} and the end effector's Cartesian position given by x, y and z.

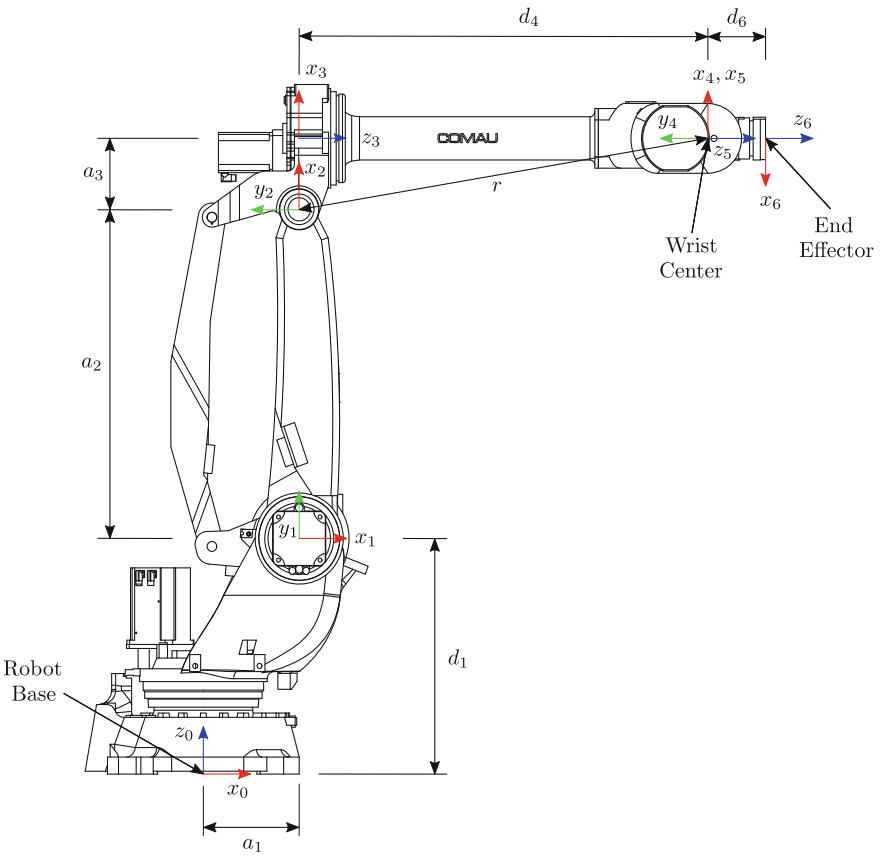

FIGURE 1. Geometric overview of Comau Smart-5 NJ-110 3.0 industrial robot ($\theta_2 = 90°$)

TABLE 1. Comau Smart-5 NJ-110 3.0 DH-table

Link i	a_i	α_i	d_i	θ_i
1	a_1	$90°$	d_1	θ_1
2	a_2	$0°$	0	θ_2
3	a_3	$90°$	0	θ_3
4	0	$-90°$	d_4	θ_4
5	0	$90°$	0	θ_5
6	0	$0°$	d_6	$\theta_6 + 180°$

The rotation matrix R_6^0's components r_{ij} are described using the ZYZ-Euler angle transformation which is given in Eq. (2.3).

$$R_6^0 = R_z(\phi)R_y(\theta)R_z(\psi) = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} \tag{2.3}$$

The rotation angles ϕ, θ and ψ are given as successive rotations relative to the current coordinate frame. The FK is now fully defined by the six input parameters $(x, y, z, \phi, \theta, \psi)$. The IK objective is to solve all the six joint angles θ_i as a function of the six input parameters $(x, y, z, \phi, \theta, \psi)$, as illustrated by Eq. (2.4).

$$\theta_i = f_i(x, y, z, \phi, \theta, \psi) \tag{2.4}$$

2.2. Wrist Center Calculation

The second step is to calculate the wrist center located in joint 4 and 5. The Cartesian coordinates of the wrist center are given by Eq. (2.5).

$$\begin{bmatrix} x_c^0 \\ y_c^0 \\ z_c^0 \end{bmatrix} = \begin{bmatrix} x - d_6 r_{13} \\ y - d_6 r_{23} \\ z - d_6 r_{33} \end{bmatrix} = \begin{bmatrix} x \\ y \\ z \end{bmatrix} + R_6^0 \begin{bmatrix} 0 \\ 0 \\ -d_6 \end{bmatrix} \tag{2.5}$$

where x_c^0, y_c^0 and z_c^0 are the wrist center position relative to the robot base coordinate system shown in Fig. 1.

2.3. Geometric Decoupling

The first three joint angles are obtained by geometric decoupling since the wrist center position is known from Eq. (2.5). The first joint angle θ_1 is simply found using Eq. (2.6).

$$\theta_1 = \text{Atan2}\left(y_c^0, x_c^0\right) \tag{2.6}$$

Equation (2.6) introduces the Atan2 function which is commonly available in most programming languages. Further θ_2 is calculated using Eqs. (2.7–2.9).

$$D = \frac{(x_c^0)^2 + (y_c^0)^2 - a_2^2 - r^2}{2a_2 r} \tag{2.7}$$

$$\alpha = \text{Atan2}\left(-\sqrt{1 - D^2}, D\right) \tag{2.8}$$

$$\theta_2 = \text{Atan2}\left(y_c^1, x_c^1\right) - \text{Atan2}\left(rs_\alpha, a_2 + rc_\alpha\right) \tag{2.9}$$

where x_c^1 and y_c^1 are the wrist center coordinates given relative to coordinate frame 1, which is placed in the origin of revolution joint 2 (see Fig. 1). To obtain the wrist center position relative to coordinate frame 1, the DH-matrix A_1 is used as shown in Eq. (2.10).

$$P_c^1 = A_1^{-1} P_c^0 \tag{2.10}$$

where P_c^0 and P_c^1 are the homogeneous representations of the wrist center coordinates relative to coordinate frames 0 and 1 respectively. The third joint angle is obtained using Eq. (2.8) together with Eq. (2.11).

$$\theta_3 = \alpha + \frac{\pi}{2} - \text{Atan2}\left(a_3, d_4\right) \tag{2.11}$$

2.4. The Orientation Angles

Rotation matrix R_3^0 is found by using joint angle variables θ_1, θ_2 and θ_3. By knowing R_3^0 and R, the numerical version of the rotation matrix R_6^3 can be calculated using Eq. (2.12).

$$R_6^3 = (R_3^0)^T R_6^0 \tag{2.12}$$

The analytical solution of the rotation matrix R_6^3 is found by multiplying the DH matrices A_4, A_5 and A_6. The resulting rotation matrix is given by Eq. (2.13).

$$R_6^3 = \begin{bmatrix} s_4 s_6 - c_4 c_5 c_6 & c_6 s_4 + c_4 c_5 s_6 & c_4 s_5 \\ -c_4 s_6 - c_5 c_6 s_4 & c_5 s_4 s_6 - c_4 c_6 & s_4 s_5 \\ c_6 s_5 & -s_5 s_6 & c_5 \end{bmatrix} \tag{2.13}$$

From this resulting matrix, the possibility arises to obtain analytical expressions for joint angle θ_4, θ_5 and θ_6. The first angle to be solved is θ_5 using Eq. (2.14).

$$\theta_5 = \cos^{-1}(R_6^3(3,3)) \tag{2.14}$$

Joint angle θ_5 described in Eq. (2.14) is calculated using an inverse cosine expression, meaning that the orientation of this angle has to be determined by an intelligent algorithm, or it has to be defined by the user to fit the specific application. As soon as this angle has been determined, the last two joint angles can be found using Eqs. (2.15) and (2.16).

$$\theta_4 = \text{Atan2}(\text{sign}(\theta_5) R_6^3(2,3), \text{sign}(\theta_5) R_6^3(1,3)) \tag{2.15}$$

$$\theta_6 = \text{Atan2}(-\text{sign}(\theta_5) R_6^3(3,2), \text{sign}(\theta_5) R_6^3(3,1)) \tag{2.16}$$

Now, all the six revolution angles have been found, the remaining step is to write a suitable algorithm to choose among the two possible sets of angles for the last three joint angles $\theta_4, \theta_5, \theta_6$.

3. Conformal Geometric Algebra Solution

In Sect. 2, the IK problem was solved using matrix algebra and trigonometric decoupling. In this section, the CGA is used to solve the IK problem in an intuitive and elegant way. One of the main benefits of using CGA is that it presents a nice and elegant way of describing geometric entities such as points, spheres, planes, circles, lines and point pairs. This gives the ability to work with complex geometric problems in a fast and intuitive way, especially if compared to traditional methods using loads of trigonometric expressions and matrices.

3.1. Short CGA Background Theory

The nature of geometric algebra and CGA are that the algebraic operations are non-commutative. In addition to being non-commutative, the algebra uses three different products. The three different products are given in Table 2.

CGA is based upon the three Euclidean basis vectors $\{e_1, e_2, e_3\}$, and two additional basis vectors $\{e_0, e_\infty\}$, where e_0 represent the origin, and e_∞ represent the infinity dimension. Using this five basis vectors, the most common geometric entities can be expressed using Table 3.

Table 2. The products of geomtric algebra

Notation	Meaning
AB	Geometric Product
$A \wedge B$	Outer Product
$A \cdot B$	Inner Product

Table 3. Conformal geometric entities

Entity	IPNS Rep.	OPNS Rep.
Point	$P = \mathbf{x} + \frac{1}{2}\mathbf{x}^2 e_\infty + e_0$	
Sphere	$S = P - \frac{1}{2}r^2 e_\infty$	$S^* = P_1 \wedge P_2 \wedge P_3 \wedge P_4$
Plane	$\pi = \mathbf{n} + de_\infty$	$\pi^* = P_1 \wedge P_2 \wedge P_3 \wedge e_\infty$
Circle	$Z = S_1 \wedge S_2$	$Z^* = P_1 \wedge P_2 \wedge P_3$
Line	$L = \pi_1 \wedge \pi_2$	$L^* = P_1 \wedge P_2 \wedge e_\infty$
Point pair	$P_P = S_1 \wedge S_2 \wedge S_3$	$P_P^* = P_1 \wedge P_2$

In Table 3, $\mathbf{x}$ is the Euclidean 3D representation of a point, r is the sphere radius, $\mathbf{n}$ is the Euclidean normal vector of a plane, and d is the normal distance between the origin and the plane hereby named π. In Table 3, all the geometric entities except from the point are represented using both Inner Product Null Space (IPNS) and Outer Product Null Space (OPNS) representations. For instance, if a plane π is constructed by taking the outer product between three points and e_∞, the resulting plane π is represented using OPNS. To obtain the IPNS representation of the plane, the dual of the plane has to be calculated. The dual of a multivector is found through dividing the multivector D with the pseudoscalar I, as demonstrated in Eq. (3.1).

$$D^* = \frac{D}{I} = \frac{D}{e_1 \wedge e_2 \wedge e_3 \wedge e_\infty \wedge e_0} \tag{3.1}$$

where D^* is the notation for the dual of multivector D. The fundamentals of handling CGA expressions are now discussed briefly. In practical implementation and development of the code, it is highly recommended to use a visualization software like CluViz created by Perwass see [9]. Another important equation is needed to extract the desired point P of a point pair P_P which is defined in Table 3. This is carried out by introducing Eq. (3.2), which extracts the desired point by changing the sign located before the root expression.

$$P = \frac{\pm\sqrt{P_P^* \cdot P_P^*} + P_P^*}{e_\infty \cdot P_P^*} \tag{3.2}$$

Now the point where CGA shows it strength and weakness are to be discussed. In CGA, there is an elegant and universal way to calculate the angle between two geometric entities such as planes and line segments. In Eq. (3.3), the angle θ between two geometric objects o_1 and o_2 is defined.

$$\theta = \angle(o_1, o_2) = \cos^{-1}\left(\frac{o_1^* \cdot o_2^*}{|o_1^*||o_2^*|}\right) \tag{3.3}$$

However, the weakness in real life engineering tasks is the lack of orientation (sign) in the computed angle θ. To obtain a signed angle one might consider applying signs depending on the position of related points in the model. In some situations, the sign may also be obtained by studying the resulting multivector coefficients. In this paper, the signs are extracted using the multivector coefficients and two scalar components of the FK orientation matrix R_6^4. The solution is elaborated in the next section.

CGA also offers operations to describe rigid motions using rotors and translators. These can be combined to a motor which is quite similar to homogeneous transformations. A rotor in CGA is described using Eq. (3.4).

$$R = e^{-\frac{\phi}{2}L} \tag{3.4}$$

where L is a normalized bivector representing the desired rotation axis, and ϕ is the rotation angle around bivector L. In order to perform translations, a translator is introduced as described in Eq. (3.5).

$$T = e^{-\frac{1}{2}(t_1e_1+t_2e_2+t_3e_3)e_\infty} \tag{3.5}$$

where $\{t_1, t_2, t_3\}$ are the translations in directions $\{e_1, e_2, e_3\}$ respectively. Rotations and translations can be combined as a motor M given by Eq. (3.6)

$$\begin{aligned} M &= RT \quad \text{(Rotation relative to a fixed coordinate frame)} \\ M &= TR \quad \text{(Rotation relative to the current coordinate frame)} \end{aligned} \tag{3.6}$$

As we can see, there are clear similarities between rigid motions described in CGA and homogeneous transformations in Euclidean space. In order to actually apply a rigid motion to an object o, the motor M is applied using Eq. (3.7).

$$o_M = Mo\tilde{M} \tag{3.7}$$

where M is the motor and $\tilde{M}$ is the reverse of the rotor. More information about the reverse operation is given in [2].

3.2. Proposed CGA IK Solution

The proposed solution follows the technique used by Hildenbrand and Bayro Corrochano with intersecting spheres and planes to obtain the fundamental points in all revolution joints. The resulting joint angles are then found by calculating the angle between the resulting line segments along the links. However, in our solution also the sign of the four first joint angles are obtained using the sign of the multivector elements.

First the point at origin P_0, two points in the x- and z-directions, and an assisting point P_A is needed. The points are given by Eqs. (3.8–3.11).

$$P_0 = e_0 \tag{3.8}$$

$$P_{0x} = e_1 + \frac{1}{2}e_\infty + e_0 \tag{3.9}$$

$$P_{0z} = e_3 + \frac{1}{2}e_\infty + e_0 \tag{3.10}$$

$$P_A = d_1e_1 + \frac{1}{2}d_1^2e_\infty + e_0 \tag{3.11}$$

To establish the end effector point P_6 and wrist center point P_c, a motor consisting of a rotor and translator are used. The rotor will describe the orientation of the end effector, and the translator will define the position relative to origin. The rotor, translator and the resulting motor are given in Eqs. (3.12–3.14).

$$R_{06} = e^{-\phi\frac{1}{2}(e_1\wedge e_2)}e^{-\theta\frac{1}{2}(e_3\wedge e_1)}e^{-\psi\frac{1}{2}(e_1\wedge e_2)} \tag{3.12}$$

$$T_{06} = e^{-\frac{1}{2}(xe_1+ye_2+ze_3)e_\infty} \tag{3.13}$$

$$M_{06} = T_{06}R_{06} \tag{3.14}$$

The resulting motor M_{06} can then be applied to calculate both the end effector point P_6 and the wrist center point P_c. These two points are defined in Eqs. (3.15–3.16).

$$P_6 = M_{06}P_0\tilde{M}_{06} \tag{3.15}$$

$$P_c = M_{06}(-d_6e_3 + \frac{1}{2}d_6^2e_\infty + e_0)\tilde{M}_{06} \tag{3.16}$$

where d_6 is the length of the last link connecting P_c and P_0. So far the required points P_0, P_c and P_6 are established. The next step is to establish point P_1 by intersecting two planes and one sphere. The first plane π_{0AC} is intersecting with point P_0, P_A and P_c, the plane is defined by Eq. (3.17).

$$\pi_{0AC} = (P_0 \wedge P_A \wedge P_c \wedge e_\infty)^* \tag{3.17}$$

The second plane $(e_3 + d_1e_\infty)$ has the normal in the z-direction and intersects with point P_A. The sphere S_0 is placed in the origin and has the radius equal to the absolute length between points P_0 and P_1. The sphere S_0 is defined in Eq. (3.18).

$$S_0 = P_0 - \frac{1}{2}\left(a_1^2 + d_1^2\right)e_\infty \tag{3.18}$$

The point pair P_{P1} is then found by intersecting the two planes and the sphere located in origin. The point pair expression is then given by Eq. (3.19).

$$P_{P1} = S_0 \wedge \pi_{0AC} \wedge (e_3 + d_1e_\infty) \tag{3.19}$$

As only one of the points in point pair P_{P1} is of interest, the desired point P_1 is found by applying Eq. (3.2). To illustrate the process, Fig. 2 shows the two planes (red and blue) and the sphere (green) located in the origin.

Figure 2 shows all the points located in all six revolution joints. So far, the points P_0, P_1, P_c and P_6 are defined. As the figure illustrates, also points P_2 and P_3 are needed to completely describe the robot kinematic chain. These remaining points are found using the same procedure as for point P_1, the only difference is that the remaining points are found by intersecting two spheres and one plane instead.

Point pair P_{P2} is found by constructing a sphere in point P_1 with radius a_2, and a sphere in point P_c with radius r. These two spheres are given by Eqs. (3.20) and (3.21).

$$S_1 = P_1 - \frac{1}{2}a_2^2e_\infty \tag{3.20}$$

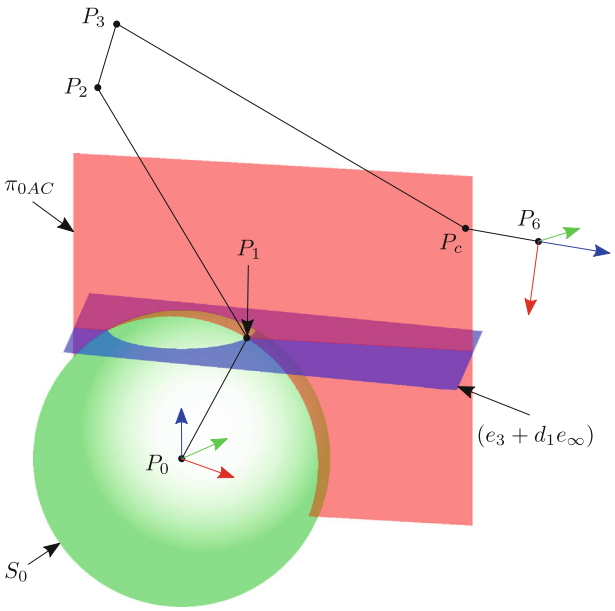

FIGURE 2. Illustration of the intersecting planes and sphere used to construct point P_1

$$S_r = P_c - \frac{1}{2} r^2 e_\infty \tag{3.21}$$

These two spheres are then intersected with the plane π_{0AC} as given in Eq. (3.22).

$$P_{P2} = S_1 \wedge S_r \wedge \pi_{0AC} \tag{3.22}$$

The same procedure is then applied to establish point pair P_{P3} where the two spheres with center in point P_2 and P_c are defined by Eqs. (3.23) and (3.24).

$$S_2 = P_2 - \frac{1}{2} a_3^2 e_\infty \tag{3.23}$$

$$S_c = P_c - \frac{1}{2} d_4^2 e_\infty \tag{3.24}$$

P_{P3} is found by again intersecting these two spheres with the plane π_{0AC}.

$$P_{P3} = S_2 \wedge S_c \wedge \pi_{0AC} \tag{3.25}$$

Similar to point pair P_{P1}, the desired point is chosen by applying Eq. (3.2). All the fundamental points found in all six revolution joints are now established. The remaining steps required to obtain the angles between the rigid links require some additional line and plane segments. These additional segments enable the use of Eq. (3.3) to calculate the respective angle between them. The additional required line segments are given in Eqs. (3.26–3.30).

$$L_{01} = (P_0 \wedge P_1 \wedge e_\infty)^* \tag{3.26}$$

$$L_{12} = (P_1 \wedge P_2 \wedge e_\infty)^* \tag{3.27}$$

$$L_{23} = (P_2 \wedge P_3 \wedge e_\infty)^* \tag{3.28}$$
$$L_{3C} = (P_3 \wedge P_C \wedge e_\infty)^* \tag{3.29}$$
$$L_{C6} = (P_C \wedge P_6 \wedge e_\infty)^* \tag{3.30}$$

The additional necessary plane segments are given in Eqs. (3.31–3.34).

$$\pi_{3C6} = (P_3 \wedge P_c \wedge P_6 \wedge e_\infty)^* \tag{3.31}$$
$$\pi_{6xz} = (P_6 \wedge P_{6x} \wedge P_{6z} \wedge e_\infty)^* \tag{3.32}$$
$$\pi_{012} = (P_0 \wedge P_1 \wedge P_2 \wedge e_\infty)^* \tag{3.33}$$
$$\pi_{123} = (P_1 \wedge P_2 \wedge P_3 \wedge e_\infty)^* \tag{3.34}$$

All the required geometric entities are now defined, and all six rotational angles should be obtained using Eq. (3.3). In real life engineering problems like IK, the orientation/sign has to be considered as well. Obtaining these signs using geometric computations like the inner product between a point and a plane could have been considered to define the angle orientations/signs. However, in this article, the sign of the multivector coefficients are observed to define these signs. A multivector in CGA is defined as a linear combination of the 32 basis elements $\{1, e_1, e_2, e_3, e_\infty, e_0, \ldots, e_1e_2e_3e_\infty e_0\}$. An example of a multivector u is given by Eq. (3.35)

$$u = u(3)e_2 + u(5)e_\infty + u(30)(e_3 \wedge e_1) \tag{3.35}$$

where $u(3)$ is the notation used to describe the scalar coefficient associated with multivector basis element e_2. The same apply for $u(5)$ and $u(30)$, these are the scalar coefficients of basis elements e_∞ and $(e_3 \wedge e_1)$ respectively. In general $u(i)$ is the ith coefficient of multivector u in CluViz. By observing the signs of the multivector coefficients $u(i)$, the signs could be defined for angles $\theta_1 \cdots \theta_4$. These four signed angles are given by Eqs. (3.36–3.39).

$$\theta_1 = \text{sign}(P_1(3))\angle(e_2^*, \pi_{0AC}^*) \tag{3.36}$$
$$\theta_2 = \text{sign}(\pi_{012}^*(30))\angle(L_{01}^*, L_{12}^*) + \tan^{-1}(d_1/a_1) \tag{3.37}$$
$$\theta_3 = \text{sign}(\pi_{123}^*(30))\angle(L_{12}^*, L_{23}^*) \tag{3.38}$$
$$\theta_4 = \text{sign}(-\pi_{3C6}(5))\angle(\pi_{0AC}^*, \pi_{3C6}^*) \tag{3.39}$$

In Eq. (3.36), $\tan^{-1}(d_1/a1)$ is added to match the angle offset in the traditional IK algorithm described in Sect. 2. To obtain the remaining two signed angles θ_5 and θ_6, the FK has to be applied in order to find the rotation matrix R_6^4 since such information could not be found in the multivector components. However, only $R_6^4(1,3)$ and $R_6^4(3,1)$ are required since these two matrix components describe the sign of the last two angles. The analytical version of R_6^4 is given by Eq. (3.40) to illustrate why only two of the matrix components are required to describe the sign of joint angles 5 and 6.

$$R_6^4 = \begin{bmatrix} -c_5c_6 & -c_5s_6 & s_5 \\ -c_6s_5 & s_5s_6 & -c_5 \\ -s_6 & -c_6 & 0 \end{bmatrix} \tag{3.40}$$

To obtain the numerical values of rotation matrix R_6^4, the four first signed joint angles and R_6^0 are used to obtain R_6^4 using Eq. (3.41).

$$R_6^4 = (R_4^0)^T R_6^0 \tag{3.41}$$

Symbolic equations for $R_6^4(1,3)$ and $R_6^4(3,1)$ can be found analytically to enable for low level PLC language implementation. The calculated values of $R_6^4(1,3)$ and $R_6^4(3,1)$ can then be used to describe the signs of joint angle θ_5 and θ_6 as shown in Eqs. (3.42) and (3.43).

$$\theta_5 = \text{sign}(R_6^4(1,3))\angle(L_{3C}^*, L_{C6}^*) \tag{3.42}$$

$$\theta_6 = \text{sign}(-R_6^4(3,1))\angle(\pi_{6xz}^*, \pi_{3C6}^*) \tag{3.43}$$

The signs of the first four joint angles were found by analyzing the multivector coefficients. The remaining two signs of joint 5 and 6 were found by applying the FK scheme. The authors believe that it should be possible to solve the correct signs for all six joint angles by a further study of the CGA properties in general. The method presented in this paper is valid as long as the wrist center point P_c is located at $x_c^0 > 0$. A separate solution for $x_c^0 \leq 0$ would have to be developed, following the same principles as outlined in this paper.

4. Gaalop Implementation and Benchmarking

In the previous Sects. 2 and 3, two different methods to solve the IK problem were presented, where the latter one used CGA as a mathematical framework. In this section, a benchmark between these two algorithms and the IK algorithm supplied with the robot are presented. The benchmark program written in C++ loops through the trajectory depicted in Fig. 3 for a high number of iterations and then measures the elapsed time for each of the three algorithms respectively.

The result of applying the benchmark is given in Table 4 where the resulting average iteration time is given in microseconds.

Table 4 indicates the superior performance of the CGA-based IK algorithm implemented using the Gaalop compiler. Since the algorithm supplied with the Comau robot and the traditional IK algorithm can only be implemented using serial computing, also the CGA-based algorithm is implemented serially. However, the CGA-based algorithm can be parallelized, which again most likely would speed up the algorithm even further. Another benefit of using CGA is that matrix computations are avoided, which simplifies the implementation of the final code on platforms like industrial PLCs.

The resulting CGA IK algorithm is implemented in an experimental setup in the MotionLab located in the Mechatronics lab at the University of Agder (see https://www.motion-lab.no). Here the algorithm has been benchmarked and used to carry out a 3-DOF motion compensation, where the robot base is moving in x, y and z-directions. A YouTube video of the 3-DOF experimental motion compensation can be seen at https://www.youtube.com/watch?v=WDpFD1c37T4. The experimental setup used in the YouTube

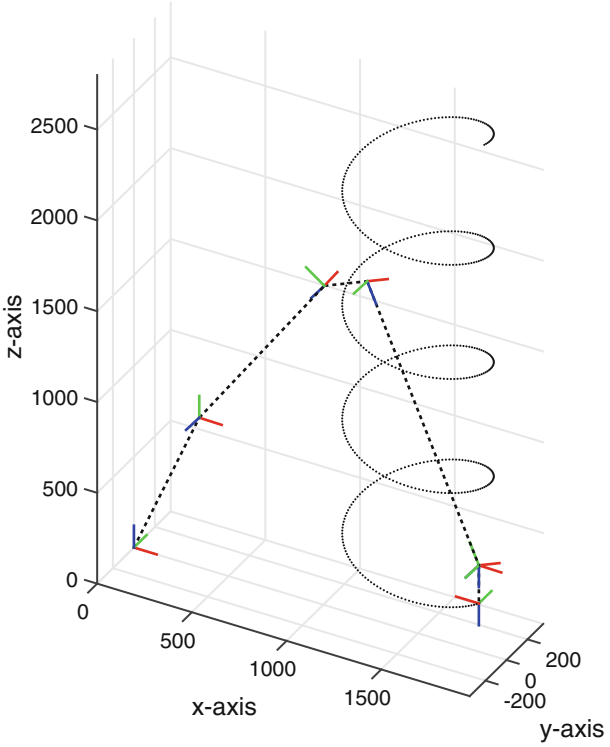

FIGURE 3. Spiral trajectory used to benchmark the three IK algorithms

TABLE 4. Average iteration time for the three different IK algorithms using an Intel i7-3555LE CPU

Comau IK	Traditional IK	CGA based IK
71.77 μs	2.93 μs	1.59 μs

video is depicted in Fig. 4, and the number indicators are described in more detail below the figure.

1. Industrial 6-DOF robot delivered by Comau Robotics (Smart-5 NJ-110 3.0).
2. Robot control cabinet featuring Comau's C5GOpen functionality.
3. Bosch Rexroth Stewart Platform (E-Motion 8000) capable of lifting 8 tons and move with 6-DOF.
4. Industrial PC featuring Linux Real-Time and PowerLink communication with the control cabinet (2). This is where the IK algorithms are implemented using optimized C++ code generated using Gaalop.
5. Handheld controller used to start up and execute commands to the industrial robot.
6. Laptop featuring human machine interface (HMI) and 3D visualization of the current robot pose.

FIGURE 4. Lab setup at the norwegian MotionLab at the University of Agder, Campus Grimstad.

5. Discussion and Conclusion

In this paper, two IK algorithms have been proposed. The first algorithm uses traditional matrix operations to solve the joint angles as a function of the input reference. The second algorithm is based on CGA, this algorithm has been implemented on a real-life robot application using optimized C++ code, which was generated by using the Gaalop compiler. The performance benchmark clearly states the benefits of using CGA over traditional matrix algebra. The CGA implementation is almost twice as fast as the traditional IK algorithm, and about 45 times faster than the IK algorithm supplied with the industrial robot. Even though the results are already convincing, the benchmarks in this paper have only been using serial processing. Since the CGA algorithm allows for parallel processing, even higher performance should be realistic. Another advantage of using CGA is that the resulting C++ code does not require any matrix operations, which means that the code has great potential to be easily implemented on an industrial PLC for instance.

Even if the results in this paper show that the resulting equations based on CGA are faster than traditional IK methods using matrix algebra, it should be noted that both the CGA and the traditional IK equations can be further optimized. For example, the following code in Maple

```
with(CodeGeneration):
x1:= ... :
x2:= ... :
C(['x1'=x1,'x2'=x2],optimize):
```

will result in significantly faster code, both for CGA and traditional IK. In this way, the requirement for a matrix library on a PLC can be removed for the traditional IK.

Another benefit of using CGA in complex geometric tasks is the more intuitive and elegant way of describing geometric entities compared to homogeneous transformations or quaternions. CGA may increase engineers ability to solve tasks involving complex geometry much more efficiently than before, both regarding computational performance and time spent in developing the algorithms. This is especially true if the visualization software CluViz is used to visualize and understand CGA.

On the downside, CGA does not offer an elegant way of describing the orientation of the angles obtained. The signs of the first four joint angles were found by analyzing the multivector coefficients. The remaining two signs of joint 5 and 6 had to be described using the FK. The authors believe that it should be possible to solve the correct signs for all six joint angles by a further study of the mathematical tools provided by GCA in general. Further work is therefore needed to describe the joint angle signs in an elegant, robust and efficient way. Such further work may strengthen the relevance of CGA in real-life engineering problems like IK.

References

[1] Chaoqun, W., Hongtao, W., Qunhua, M.: Inverse kinematics computation in robotics using conformal geometric algebra. IET Conf. Proc. (2009). doi:10.1049/cp.2009.1527

[2] Hildenbrand, D., Simos, T.E., Psihoyios, G., Tsitouras, Ch., Anastassi, Z.: Foundations of Geometric Algebra Computing. Springer (2013). doi:10.1007/978-3-642-31794-1

[3] Hildenbrand, D., Fontijne, D., Wang, Y., Alexa, M., Dorst, L.: Competitive runtime performance for inverse kinematics algorithms using conformal geometric algebra. In: Eurographics Conference of Vienna, 2006

[4] Hildenbrand, D., Koch, A.: Gaalop-High Performance Computing Based on Conformal Geometric Algebra. In: Proceedings of the AGACSE Conference Grimma near Leipzig, Germany, August 2008 (2008)

[5] Hildenbrand, D., Lange, H., Stock, F., Koch, A.: Efficient inverse kinematics algorithm based on conformal geometric algebra using reconfigurable hardware. Citeseer (2008)

[6] Hildenbrand, D., Zamora, J., Bayro-Corrochano, E.: Inverse kinematics computation in computer graphics and robotics using conformal geometric algebra. Adv. Appl. Cliff Algebra **18**(3–4), 699–713 (2008)

[7] Lange, H., Stock, F., Koch, A., Hildenbrand, D.: Acceleration and energy efficiency of a geometric algebra computation using reconfigurable computers and GPUs. In: Proceedings of the 17th IEEE Symposium on Field Programmable Custom Computing Machines, 2009. FCCM'09, pp. 255–258. IEEE (2009)

[8] Megahed, S.M.: Inverse kinematics of spherical wrist robot arms: analysis and simulation. J. Intell. Robot. Syst. **5**(3), 211–227 (1992)

[9] Perwass, C.: Geometric Algebra with Applications in Engineering, 1st edn (2009)

[10] Pieper, D.L.: The kinematics of manipulators under computer control. Stanford, Calif.: Computer Science Department, Stanford University (1968)

[11] Pitt, J., Hildenbrand, D., Stelzer, M., Koch, A.: Inverse kinematics of a humanoid robot based on conformal geometric algebra using optimized code generation. In: Proceedings of the Humanoids 2008-8th IEEE-RAS International Conference on Humanoid Robots, pp. 681–868. IEEE (2008). doi:10.1109/ICHR.2008.4756025

[12] Spong, M.W., Hutchinson, S., Vidyasagar, M.: Robot Modeling and Control, vol. 3. Wiley, New York (2006)

Sondre Sanden Tørdal, Geir Hovland and Ilya Tyapin
The University of Agder
Jon Lilletuns vei 9
4876 Grimstad
Norway
e-mail: sondre.tordal@uia.no

Geir Hovland
e-mail: geir.hovland@uia.no

Ilya Tyapin
e-mail: ilya.tyapin@uia.no

Received: February 21, 2016.
Accepted: June 11, 2016.