Differential Kinematics Control Using Circles as Bivectors of Conformal Geometric Algebra

Julio Zamora-Esquivel[1][a], Alberto Jaimes Pita[1][b], Edgar Macias-Garcia[1][c], Javier Felip-Leon[1][d], David Gonzalez-Aguirre[1][e] and Eduardo Bayro-Corrochano[2][f]

[1]*Intelligent System Research, Intel Labs, Zapopan, Jalisco, Mexico*

[2]*Poznan University of Technology, Poznan, Poland*

Keywords: Grasping, Kinematics, Manipulation Planning.

Abstract: In this paper, we propose a modern mathematical framework to model a robotic arm and compute the differential kinematics of its end effector, which is represented as a circle in a three-dimensional space. This circle is described using a bi-vector within the context of conformal geometric algebra. By utilizing a circle to characterize the grasping pose on the object and the pose of the end-effector, we develop a differential kinematics-based control law that guides the end-effector to minimize the error between both circles. The circle representation offers three degrees of freedom for the center, two degrees for orientation, and one degree for the radius, allowing us to effectively describe the end-effector pose using a single geometric primitive. Our approach allows for simultaneous adjustment of both the position and orientation of the end effector.

1 INTRODUCTION

Visually guided grasping is a well-established problem that has been addressed through various approaches (Fang et al., 2020), (Wei et al., 2021), (Fourmy et al., 2023), (Farias et al., 2021). Typically, it involves developing a control law to adjust the joint angles of a robotic arm to minimize the error between the end-effector and the target pose, defined by the orientation and position, where most of the state-of-the-art algorithms often tackle this problem by controlling position and orientation independently (Ma et al., 2020). In contrast, our proposed algorithm employs a single geometric primitive to simultaneously describe both the position and orientation of the end-effector and the target goal. This geometric primitive is efficiently represented as a bivector within the Conformal Geometric Algebra (CGA) framework.

As a practical example, we model the kinematics and differential kinematics of the 7-DoF Franka robot arm and the Unitree G1 humanoid robot. Our control law, grounded in the differential kinematics of circles, enables the efficient displacement of both end effectors to a set of predefined target poses with a cylindrical geometry $p_{obj} = [x, y, z, \alpha, \beta]$ (Figure 1).

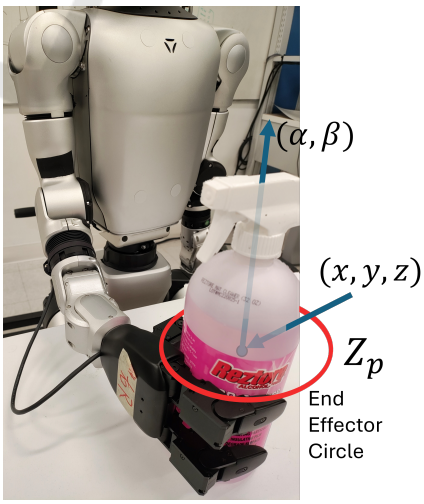

Figure 1: End-effector pose of an object described with a circle $Z_p \in G_{4,1}$ in conformal geometric algebra.

[a] https://orcid.org/0000-0002-0226-0047
[b] https://orcid.org/0009-0009-8339-6605
[c] https://orcid.org/0000-0003-2571-9460
[d] https://orcid.org/0000-0002-2115-4610
[e] https://orcid.org/0000-0002-5032-8261
[f] https://orcid.org/0000-0002-4738-3593

Zamora-Esquivel, J., Pita, A. J., Macias-Garcia, E., Felip-Leon, J., Gonzalez-Aguirre, D. and Bayro-Corrochano, E.
Differential Kinematics Control Using Circles as Bivectors of Conformal Geometric Algebra.
DOI: 10.5220/0013945500003982

In *Proceedings of the 22nd International Conference on Informatics in Control, Automation and Robotics (ICINCO 2025) - Volume 2*, pages 585-592
ISBN: 978-989-758-770-2; ISSN: 2184-2809

The rest of the paper is organized as follows: Section 2 provides a brief introduction to Conformal Geometric Algebra and the representation of circles as geometric entities. In Section 3, we present our proposed methodology to model and solve the differential kinematics of end-effectors using circles in Conformal Geometric Algebra. Section 4 introduces a novel approach for applying a control law based on these techniques, with different practical implementations on Section 5. Finally, Section 6 discusses the conclusions and potential directions for future work.

2 CLIFFORD ALGEBRA AND CONFORMAL GEOMETRY

Geometric algebra ($G_{4,1}$) provides an elegant framework for expressing conformal geometry. To illustrate this, we adopt the formulation presented in (Li et al., 2001) and demonstrate how the Euclidean vector space ($\mathbb{R}^3$) is represented within ($\mathbb{R}^{4,1}$). This space is characterized by an orthonormal vector basis $(e_1, e_2, e_3, e_4, e_5)$, with the properties of the Clifford product detailed in Table 1.

Table 1: Clifford product for blades in Conformal Geometric Algebra ($G_{4,1}$).

Basis	e_1	e_2	e_3	e_4	e_5
e_1	1	e_{12}	$-e_{31}$	e_{14}	e_{15}
e_2	$-e_{12}$	1	e_{23}	e_{24}	e_{25}
e_3	e_{31}	$-e_{23}$	1	e_{34}	e_{35}
e_4	e_{41}	e_{42}	e_{43}	1	E
e_5	e_{51}	e_{52}	e_{53}	$-E$	-1

Where $e_{ij} = e_i \wedge e_j$ is a bivectorial basis, while e_{23}, e_{31} and e_{12} are the Hamilton basis. A unit Euclidean pseudo-scalar I_e, a pseudo-scalar I_c and the bivector E can be defined by:

$$I_e : = e_1 \wedge e_2 \wedge e_3, \quad (1)$$
$$E : = e_4 \wedge e_5 = e_4 e_5, \quad (2)$$
$$I_c : = I_e \wedge E = I_e E. \quad (3)$$

2.1 Geometric Entities

Within this mathematical framework, it is possible to represent various geometric entities such as points, lines, planes, circles, and spheres in a 3D space (For a detailed description, please refer to Bayro (Bayro-Corrochano, 2018)). This framework enables the representation of a point at infinity (e_∞), and the origin (e_o):

$$e_\infty = e_4 + e_5, \quad (4)$$
$$e_0 = \frac{1}{2}(e_5 - e_4), \quad (5)$$

which can be used to define other geometric entities, and apply transformations between them. A 3D Euclidean Point ($x_e \in G_3$) can be mapped to a conformal point ($x_c \in G_{4,1}$) through the transformation:

$$x_c = x_e + \frac{1}{2}x_e^2 e_\infty + e_0. \quad (6)$$

2.2 Spheres and Planes

The equation of a sphere of radius ρ centered at point $p_e \in \mathbb{R}^n$ can be written as:

$$(x_e - p_e)^2 = \rho^2, \quad (7)$$

since $x_c \cdot y_c = -\frac{1}{2}(\mathbf{x}_e - \mathbf{y}_e)^2$, we can rewrite the formula above in terms of homogeneous coordinates as:

$$x_c \cdot p_c = -\frac{1}{2}\rho^2, \quad (8)$$

by considering that $x_c \cdot e_\infty = -1$, we can factor the expression above to:

$$x_c \cdot (p_c - \frac{1}{2}\rho^2 e_\infty) = 0. \quad (9)$$

Which finally yields the simplified equation for the sphere as $s = p_c - \frac{1}{2}\rho^2 e_\infty$ (note from this equation that a point is just a sphere with zero radius). Alternatively, the dual of the sphere is represented as $(n+1)$-vectors $s^* = sI^{-1}$. The advantage of the dual form is that the sphere can be directly computed from four points (in 3D) as:

$$s^* = x_{c_1} \wedge x_{c_2} \wedge x_{c_3} \wedge x_{c_4}. \quad (10)$$

If we replace one of these points for the point at infinity we get:

$$\pi^* = x_{c_1} \wedge x_{c_2} \wedge x_{c_3} \wedge e_\infty. \quad (11)$$

So, as π becomes in the standard form:

$$\pi = \pi^* I^{-1}, \quad (12)$$
$$\pi = n + d e_\infty, \quad (13)$$

where n is the normal vector and d represents the Hesse distance.

2.3 Circles and Lines

A circle z can be regarded as the intersection of two spheres s_1 and s_2 as $z = (s_1 \wedge s_1)$. The dual form of the circle can be expressed by three points:

$$z^* = x_{c_1} \wedge x_{c_2} \wedge x_{c_3}. \quad (14)$$

Similar to the case of planes, lines can be defined by circles passing through the point at infinity as:

$$L^* = x_{c_1} \wedge x_{c_2} \wedge e_\infty, \tag{15}$$

while the standard form of the line (in 3D) can be expressed by:

$$L = l + e_\infty (t \cdot l). \tag{16}$$

The line in the standard form is a bivector, and it has six parameters (Plucker coordinates), but just four degrees of freedom. A full list of entities is described in Tables 2 and 3.

Table 2: Representation of conformal geometric using the standard representation.

Entity	Representation
Sphere	$s = p - \frac{1}{2}\rho^2 e_\infty$
Point	$x_c = x_e + \frac{1}{2}x_e^2 e_\infty + e_0$
Line	$L = \pi_1 \wedge \pi_2$
Plane	$\pi = n + d e_\infty$
Circle	$z = s_1 \wedge s_2$
Pair of P.	$P_p = s_1 \wedge s_2 \wedge s_3$

Table 3: Representation of conformal geometric entities using the dual representation.

Entity	Dual Representation
Sphere	$s^* = x_1 \wedge x_2 \wedge x_3 \wedge x_4$
Point	$x^* = s_1 \wedge s_2 \wedge s_3 \wedge s_4$
Line	$L^* = x_1 \wedge x_2 \wedge e_\infty$
Plane	$\pi^* = x_1 \wedge x_2 \wedge x_3 \wedge e_\infty$
Circle	$z^* = x_1 \wedge x_2 \wedge x_3$
Pair of P.	$P_p^* = x_1 \wedge x_2$

3 DIFFERENTIAL KINEMATICS OF CIRCLES

In conformal geometric algebra, the forward kinematics of the end-effector circle $z_p \in R_{4,1}$ is given by (Hildenbrand et al., 2008):

$$z'_p = \prod_{i=1}^{n} M_i z_p \prod_{i=1}^{n} \widetilde{M}_{n-i+1}. \tag{17}$$

In this equation motors M are used to represent $3D$ rigid transformations as the motor algebra does (E. and Kähler, 2000):

$$M = \cos\left(\frac{\theta}{2}\right) - \sin\left(\frac{\theta}{2}\right) L = e^{-\frac{\theta}{2}L}, \tag{18}$$

now we produce an expression for differential kinematics through the total differentiation of (17) as follows:

$$dz'_p = \sum_{j=1}^{n} \partial_{q_j} \left(\prod_{i=1}^{n} M_i z_p \prod_{i=1}^{n} \widetilde{M}_{n-i+1} \right) dq_j, \tag{19}$$

where $\widetilde{M}$ is the motor conjugate. Each term of the sum is the product of two functions in q_j, then the differential yields:

$$dz'_p = \sum_{j=1}^{n} \left[\partial_{q_j} \left(\prod_{i=1}^{j} M_i \right) \prod_{i=j+1}^{n} M_i z_p \prod_{i=1}^{n} \widetilde{M}_{n-i+1} + \prod_{i=1}^{n} M_i z_p \prod_{i=1}^{n-j} \widetilde{M}_{n-i+1} \partial_{q_j} \left(\prod_{i=n-j+1}^{n} \widetilde{M}_{n-i+1} \right) \right] dq_j. \tag{20}$$

Since $M = e^{-\frac{1}{2}qL}$, the differential of the motor can be defined as $d(M) = -\frac{1}{2}MLdq$. Thus, we can write the partial differential of the motor's product as follows:

$$\begin{aligned} \partial_{q_j} \left(\prod_{i=1}^{j} M_i \right) &= -\frac{1}{2} \prod_{i=1}^{j} M_i L_j \\ &= -\frac{1}{2} \left(\prod_{i=1}^{j-1} M_i \right) L_j M_j. \end{aligned} \tag{21}$$

In a similar approach, the differential of the $\widetilde{M} = e^{\frac{1}{2}qL}$ give us $d(\widetilde{M}) = \frac{1}{2}MLdq$, and the differential of the product is:

$$\partial_{q_j} \left(\prod_{i=n-j+1}^{n} \widetilde{M}_{n-i+1} \right) = \frac{1}{2} \widetilde{M}_j L_j \prod_{i=n-j+2}^{n} \widetilde{M}_{n-i+1}, \tag{22}$$

by replacing (21) and (22) in (20) we get:

$$dz'_p = -\frac{1}{2} \sum_{j=1}^{n} \left[\prod_{i=1}^{j-1} M_i \left(L_j \left(\prod_{i=j}^{n} M_i z_p \prod_{i=1}^{n-j+1} \widetilde{M}_{n-i+1} \right) - \left(\prod_{i=j}^{n} M_i z_p \prod_{i=1}^{n-j+1} \widetilde{M}_{n-i+1} \right) L_j \right) \prod_{i=1}^{j-1} \widetilde{M}_{j-i} \right] dq_j. \tag{23}$$

By definition the product "$\circ$" of two bivectors is given by:

$$A \circ B = AB - BA \tag{24}$$

Using the equation (24) we can simplify (23), since L and Z_p are bivectors then we can rewrite (23) as:

$$dz'_p = \sum_{j=1}^{n} \left[\left(\prod_{i=1}^{j-1} M_i \right) \left(\left(\prod_{i=j}^{n} M_i z_p \prod_{i=1}^{n-j+1} \widetilde{M}_{n-i+1} \right) \circ L_j \right) \prod_{i=1}^{j-1} \widetilde{M}_{j-i} \right] dq_j. \tag{25}$$

The product of $i = [1, j-1]$ and $i = [j, n]$ is equal to the product of $i = [1, n]$. Also for $\widetilde{M}$, equation (25) can be written as:

$$dz'_p = \sum_{j=1}^{n} \left[\left(\prod_{i=1}^{n} M_i z_p \prod_{i=1}^{n} \widetilde{M}_{n-i+1} \right) \circ \left(\prod_{i=1}^{j-1} M_i L_j \prod_{i=1}^{j-1} \widetilde{M}_{j-i} \right) \right] dq_j. \tag{26}$$

Using the equation of the direct kinematics (17), we can simplify (26) as:

$$dz'_p = \sum_{j=1}^{n} \left[z'_p \circ \left(\prod_{i=1}^{j-1} M_i L_j \prod_{i=1}^{j-1} \widetilde{M}_{j-i} \right) \right] dq_j. \tag{27}$$

The equation of forward kinematics of circles also applies for Lines, in this way we can use (17) to define the transformed line L' in terms of L as follows:

$$L'_j = \prod_{i=1}^{j-1} M_i L_j \prod_{i=1}^{j-1} \widetilde{M}_{j-i}. \tag{28}$$

Finally, by replacing (28) on (27) we get a very compact expression of differential kinematics:

$$dz'_p = \sum_{j=1}^{n} \left[z'_p \circ L'_j \right] dq_j. \tag{29}$$

4 KINEMATIC CONTROL OF A ROBOT ARM

In this section, we first define a control law based on the orientation of the end effector, followed by a control law based on its position. These two aspects are then integrated into a unified control law that simultaneously addresses both position and orientation, utilizing the circle representation to model the pose of the end-effector and the desired position (Figure 2).

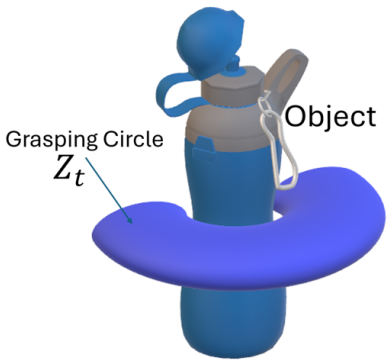

Figure 2: Target circle estimated in the manipulated object.

This kinematic control is formulated as a training rule for a neural network, utilizing gradient descent to adjust the weights, specifically the joint angles (q). The primary objective is to adjust the values of q to minimize the error, defined as the difference between the end-effector's orientation and the target orientation, as expressed by the following equation:

$$E_o = \frac{1}{2} (Z_t - Z_p)^2. \tag{30}$$

In this context, (Z_t) represents the target circle, while (Z_p) denotes the circle generated by the gripper as the end-effector, which describes the pose of the robotic arm defined by the joints q. This relationship is established through the direct kinematics equation, as given by equation (17). To adjust the joint angles (q) and minimize the error (E), the partial derivative is computed as follows:

$$\frac{\partial E_o}{\partial q_i} = -(Z_t - Z_p) \frac{\partial Z_p}{\partial q_i}. \tag{31}$$

In previous sections the differential kinematics was formulated in terms of the rotation axis L_i as:

$$\frac{\partial E_o}{\partial q_i} = -(Z_t - Z_p) \cdot \left[Z_p \circ L'_i \right], \tag{32}$$

now, to create a control law for the position of the end-effector the error given by the delta in position will be computed as:

$$E_p = \frac{1}{2} (P_t - X_p)^2, \tag{33}$$

where P_t represents the target position and X_p represents the position of the end effector. Here X_p is also the center of the circle and it can be also replaced by a sphere S_p with the center in the center of the circle:

$$S_p = Z_p / \pi_p, \tag{34}$$

where π_p is the plane of the circle given by $\pi_p^* = Z_p^* \wedge e_\infty$. Then the error can be rewritten as:

$$E_p = \frac{1}{2} (P_t - S_p)^2. \tag{35}$$

The training rule to minimize the error E_p by adjusting the joint angles q can be written as:

$$\frac{\partial E_p}{\partial q_i} = -(P_t - S_p) \frac{\partial S_p}{\partial q_i}. \tag{36}$$

According to (Bayro-Corrochano and Zamora-Esquivel, 2007) the differential kinematics of points and spheres can be computed using $[S_p \cdot L'_i]$. Then we can simplify equation (36) as:

$$\frac{\partial E_p}{\partial q_i} = -(P_t - S_p) \cdot \left[Z_p \pi_p^{-1} \cdot L'_i \right]. \tag{37}$$

Thus, we can merge the equations (37) and (32), as they are vectors and bivectors, respectively. This merge is achieved through a weighted sum by incorporating the learning rates (η_o) and (η_p), which, in control terms, represent the gain of the control law.

$$\frac{\eta_p \partial E_p}{\partial q_i} + \frac{\eta_o \partial E_o}{\partial q_i} = -\eta_p (P_t - S_p) \cdot \left[Z_p \pi_p^{-1} \cdot L_i' \right] - \eta_o (Z_t - Z_p) \cdot \left[Z_p \circ L_i' \right], \quad (38)$$

by reordering the terms:

$$\frac{\eta_p \partial E_p}{\partial q_i} + \frac{\eta_o \partial E_o}{\partial q_i} = -(\eta_p (P_t - S_p) + \eta_o (Z_t - Z_p)) \cdot \left[Z_p (\pi_p^{-1} + 1) L_i' \right], \quad (39)$$

the rule to update the joint angles can be written as:

$$\Delta q_i = (\eta_p (P_t - S_p) + \eta_o (Z_t - Z_p)) \cdot \left[Z_p (\pi_p^{-1} + 1) L_i' \right], \quad (40)$$

where S_p, S_t are the spheres, and P_t is the center of the sphere S_t.

5 APPLICATIONS

In this section, we implement the differential kinematics and control laws described in the previous sections to model and manipulate the position and orientation of the end-effector for different robots, including both single and bi-manual grasping scenarios.

5.1 7DOF Franka Emika Robot Arm

As a first experiment, we consider the 7-DoF Franka Emika robot arm. By employing the equation (17) we can model the forward kinematics of the end-effector by employing a set of lines and motors over each joint, with a circle entity placed on the end-effector (Figure 3). This formulation allows to describe the position and orientation of the end-effector in terms of the joints position.

To define the desired position and orientation of the end effector, we developed a Mixed Reality Environment called Roomersive. This environment has the capability to display robots and virtual objects in augmented reality, facilitating real-time interactions with other users within the same scene (Figure 4).
To enable the interaction between users and the simulated robot, we implemented a detection system composed of four Deep Learning Neural Networks

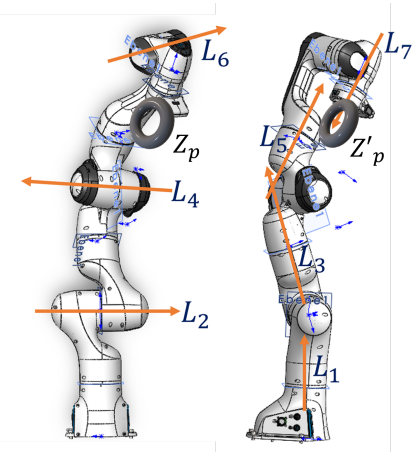

Figure 3: Robot arm Axis represented by lines L_i, each one of those lines is given by a Bivector of conformal geometry.

Models running in parallel, trained for scene understanding; human pose estimation (based on (Cao et al., 2017)), face re-identification (based on FaceNet (Schroff et al., 2023)), object recognition (based on (Wang et al., 2023)), and action recognition (Archana and Hareesh, 2021). Each model predicts a set of 2D keypoints, which are then converted to 3D and mapped into the immersive space using data from a RealSense depth camera. Additionally, this innovative simulator enables the connection of a real robot with its digital twin within the immersive environment, by sending the calculated joints to the real robot and reflecting the current real position on the simulator. This feature allows users to manipulate the real robot by interacting with the virtual object.

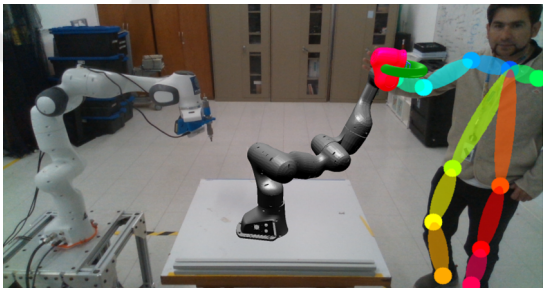

Figure 4: Roomersive creates a digital twin of a real robot within an immersive environment, allowing users to interact with it and transmit information to a real robot located elsewhere.

To enable the interactions between users and virtual objects, we employ procedural geometry; This geometry is generated using the Hull and Domain shaders of DX11/DX12 (Corporation, 2023) on the GPU. Essentially, the depth image is mapped as a texture to the GPU, and an 8×13 array of patches is created, each with four control points. In the domain shader, each

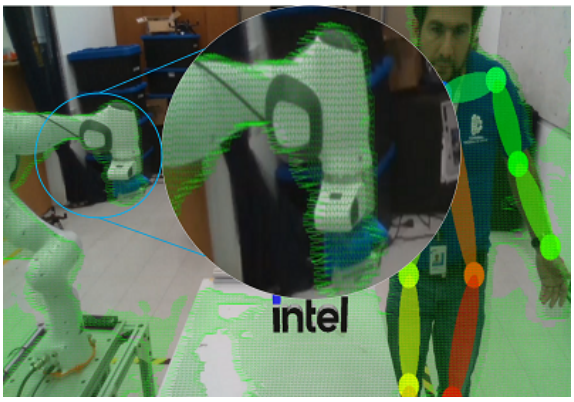

Figure 5: Procedural Geometry created using Depth information from Realsense Camera.

patch is tessellated up to 64×64, producing 4,096 vertices. For each generated vertex, we sample the depth information, which is back-propagated to generate 851,968 triangles per frame (Figure 5).

Rendering integration is achieved by creating a virtual camera with identical intrinsic parameters and positioning as the real camera. The image is then copied to the back buffer, and the procedural geometry is rendered with transparency to facilitate the visualization (Figure 6). This allows virtual objects to be occluded by or occlude real objects based on their positions. The proposed control law is applied in two scenarios: first, to remotely manipulate the robot arm by grasping and moving the virtual end-effector; and second, to grasp objects using a grasping circle located in the same position than the real objects.

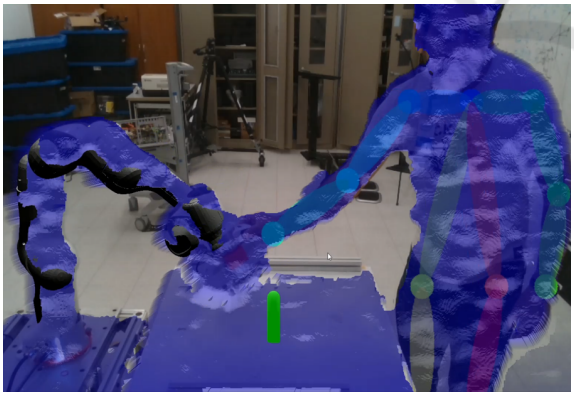

Figure 6: Procedural geometry in blue, used to set the scene depth on the 3D pipeline.

5.1.1 Immersive Remote Control

In this application we use human pose estimation and depth information to identify the wrist position in 3D, and based on the forward kinematics we estimate the position and orientation of the robot's end effector

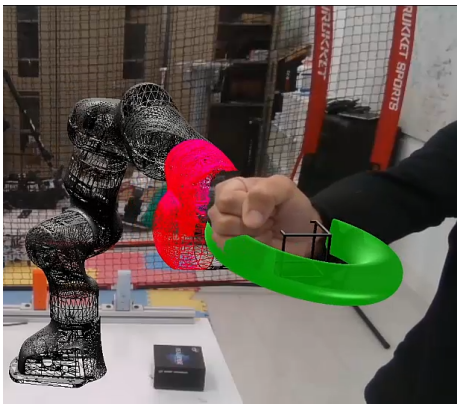

Figure 7: Wrist detection modeled with a circle, used to pull the virtual robot arm.

described with a circle, when the direct distance between the wrist circle and the end effector circle is less than a threshold and the close palm action is detected on the hand, the control law (40) is activated, this makes the robot follow the hand movement, it is important to note that we do not require the use of manual controls that allow the user to manipulate the objects.

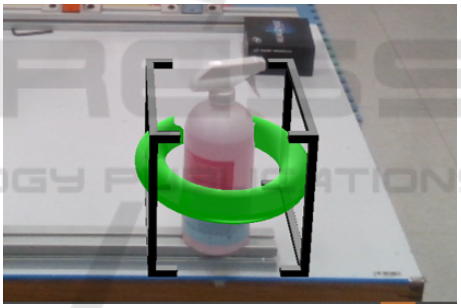

Figure 8: Object detected by our NN in 3D and the circle of grasping in the Mixed reality space.

5.1.2 Object Grasping

In this scenario, we employ a YOLO-based neural network to detect objects, using a bottle as an illustrative example (Figure 8). Once the object is located in 3D and isolated within a bounding cube, the target circle for the object is estimated. This initiates the control law 40, which guides the robot to grasp the object by simultaneously aligning its position and orientation.

5.2 Unitree G1, Humanoid Robot

As an additional experiment, we constructed the Unitree G1 humanoid robot using the same forward kinematics procedure applied to the Franka Emika robot (equation ((17))). We then implemented various in-

teraction levels in our simulator to manipulate the position of the joints based on the user's body pose information. To facilitate this, we developed models to identify when the user opens or closes their hands. Upon detecting that a joint is "grasped," we calculate a pair of circles based on the position and orientation of the selected joint and the hand. These circles are then adjusted using our control law to minimize the error between them (Figure 9).

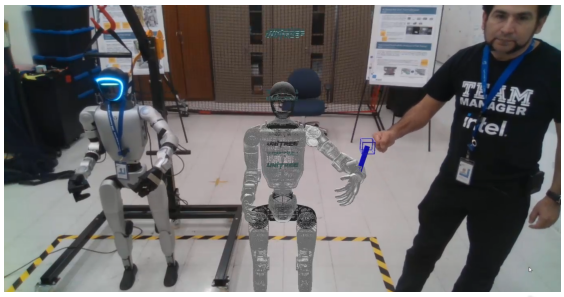

Figure 9: Interaction level 1: The user can "grasp" the joint that he wants to adjust.

Our differential kinematics algorithm can also be applied to track the human body, adjusting joint angles to accommodate multiple targets. In this approach, each joint incrementally adjusts the preceding joints to achieve the desired position (Figure 10). Additionally, a single target circle can be used to manipulate multiple end effectors simultaneously, such as in bimanual grasping. In the experiment shown in Figure 11, a single target position is defined for both hands of the robot, enabling it to "grasp" the object. In this setup, each joint receives an average of the deltas from the errors of the two end effectors.

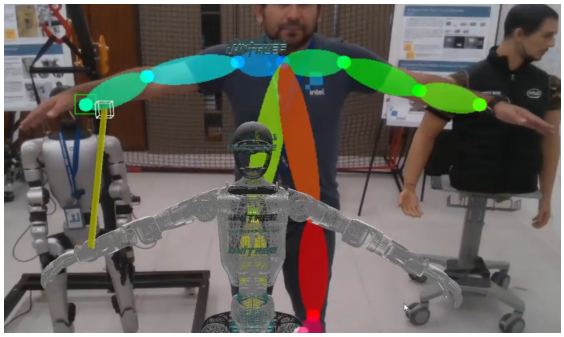

Figure 10: Interaction level 2: Humanoid robot mimicking human movements using the references of the body pose.

Additionally, other interaction levels can be implemented, for example, by choose the action to do using the immersive space (Figure 12).

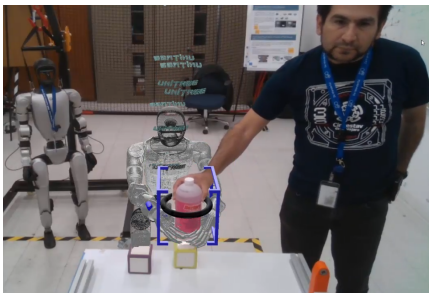

Figure 11: Interaction level 3: A single target desired pose is defined for both hands.

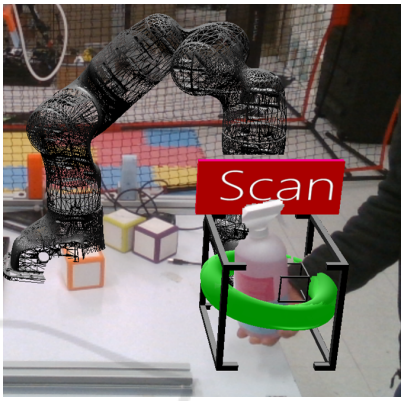

Figure 12: When the user interacts with an object, a pop-up virtual menu appears to select the desired action.

5.3 Results Discussion

The aim of this work is not to compete with the state-of-the-art in terms of speed; rather, it introduces a modern representation of the end-effector and control law using circles on the geometric algebra framework, resulting in equations that are more computationally efficient. Traditional robot kinematics, typically calculated using matrices, require 64 MAC operations to concatenate two transformations, whereas conformal geometry rotations require only 16 MAC operations for concatenation. This control law simultaneously addresses both rotation and position, enhancing efficiency. Our immersive reality system eliminates the need for handheld controllers and virtual reality glasses, offering a more natural interaction experience. Figure 13 illustrates the trajectory graph of the user's hand in 3D compared to the trajectory of the end-effector after applying the control law. Human pose detection, action recognition, and object detection operate at 60 fps, alongside the procedural generation and rendering of virtual objects and robots.

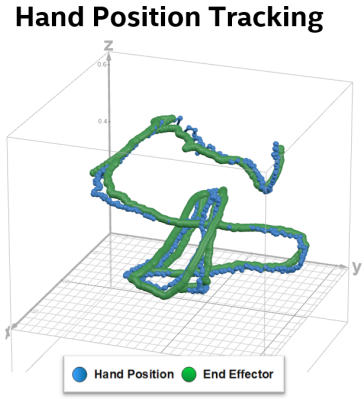

Figure 13: Internal circle components converging after 240 frames (4 seconds).

6 CONCLUSIONS

In this work, the authors aim to demonstrate the feasibility of the new mathematical representation and control algorithm. By avoiding the use of matrices for computing kinematics and control, they introduce a novel formulation based on dual quaternions as bivectors. The authors do not intend to compare performance with traditional methods, as matrix operations are typically accelerated on GPUs, whereas Clifford algebras are not. Although this method can reduce MAC operations, the lack of acceleration or parallelization means that results may be comparable in terms of time, depending on the hardware used. A separate study focusing on performance, metrics, and hardware using both approaches will be presented in a subsequent article. This work is limited to cylindrical shapes that can be gripped from any position along the circle. To ensure a unique grip pose, a parabola could be used instead of a circle, thereby adding additional degrees of freedom to describe it. This would necessitate the use of Quaternion Geometric Algebra (QGA) (Zamora-Esquivel, 2014) to calculate the kinematics and control. We are considering this approach for future work.

REFERENCES

Archana, N. and Hareesh, K. (2021). Real-time human activity recognition using resnet and 3d convolutional neural networks. In *2021 2nd International Conference on Advances in Computing, Communication, Embedded and Secure Systems (ACCESS)*, pages 173–177. IEEE.

Bayro-Corrochano, E. (2018). *Geometric algebra applications vol. I: Computer vision, graphics and neurocomputing*. Springer.

Bayro-Corrochano, E. and Zamora-Esquivel, J. (2007). Differential and inverse kinematics of robot devices using conformal geometric algebra. *Robotica*, 25(1):43–61.

Cao, Z., Simon, T., Wei, S. E., and Sheikh, Y. (2017). Real-time multi-person 2d pose estimation using part affinity fields. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7291–7299.

Corporation, I. (2023). Directx 12 ultimate on intel® arc™ graphics. Intel. https://www.intel.com/content/www/us/en/architecture-and-technology/visual-technology/directx-12-ultimate.html.

E., B.-C. and Kähler, D. (2000). Motor algebra approach for computing the kinematics of robot manipulators. *Journal of Robotics Systems*, 17(9):495–516.

Fang, W., Chao, F., Lin, C. M., Zhou, D., Yang, L., Chang, X., and Shang, C. (2020). Visual-guided robotic object grasping using dual neural network controllers. *IEEE Transactions on Industrial Informatics*, 17(3):2282–2291.

Farias, C. D., Adjigble, M., Tamadazte, B., Stolkin, R., and Marturi, N. (2021). Dual quaternion-based visual servoing for grasping moving objects. In *2021 IEEE 17th International Conference on Automation Science and Engineering (CASE)*, pages 151–158. IEEE.

Fourmy, M., Priban, V., Behrens, J. K., Mansard, N., Sivic, J., and Petrik, V. (2023). Visually guided model predictive robot control via 6d object pose localization and tracking. *arXiv preprint*, arXiv:2311.05344.

Hildenbrand, D., Zamora, J., and Bayro-Corrochano, E. (2008). Inverse kinematics computation in computer graphics and robotics using conformal geometric algebra. *Advances in applied Clifford algebras*, 18:699–713.

Li, H., Hestenes, D., and Rockwood, A. (2001). Generalized homogeneous coordinates for computational geometry. In Somer, G., editor, *Geometric Computing with Clifford Algebras*, pages 27–52. Springer-Verlag Heidelberg.

Ma, Y., Liu, X., Zhang, J., Xu, D., Zhang, D., and Wu, W. (2020). Robotic grasping and alignment for small-size components assembly based on visual servoing. *The International Journal of Advanced Manufacturing Technology*, 106:4827–4843.

Schroff, F., Kalenichenko, D., and Philbin, J. (2023). Facenet: A unified embedding for face recognition and clustering. In *The Computer Vision Foundation*. Retrieved 4 October 2023.

Wang, C. Y., Bochkovskiy, A., and Liao, H. Y. M. (2023). Yolov7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 7464–7475.

Wei, H., Pan, S., Ma, G., and Duan, X. (2021). Vision-guided hand–eye coordination for robotic grasping and its application in tangram puzzles. *AI*, 2(2):209–228.

Zamora-Esquivel, J. (2014). G 6, 3 geometric algebra; description and implementation. *Advances in Applied Clifford Algebras*, 24(2):493–514.