The Forward and Inverse Kinematics of a Delta Robot

Chapter · October 2020

DOI: 10.1007/978-3-030-61864-3_38

3 authors, including:

Hugo Hadfield
University of Cambridge
15 PUBLICATIONS **90** CITATIONS
SEE PROFILE

Joan Lasenby
University of Cambridge
200 PUBLICATIONS **3,231** CITATIONS
SEE PROFILE

The Forward and Inverse Kinematics of a Delta Robot

Hugo Hadfield[1 [0000−0003−4318−050X]], Lai Wei[1],
Joan Lasenby[1 [0000−0002−0571−0218]]

University of Cambridge, Department of Engineering, Cambridge, UK,
hh409@cam.ac.uk, letticewei@gmail.com, jl221@cam.ac.uk

Abstract. The Delta robot is one of the most popular parallel robots in industrial use today. In this paper we analyse the forward and inverse kinematics of the robot from a geometric perspective using Conformal Geometric Algebra. We calculate explicit formulae for all joints in both the forward and inverse kinematic problems as well as explicit forward and inverse Jacobians to allow for velocity and force control. Finally we verify the kinematics in Python and simulate a physical model in the Unity3D game engine to act as a test-bed for future development of control algorithms.

1 Introduction

The Delta robot [1, 2] was invented in 1985 by Raymond Clavel at EPFL after being inspired by a visit to a chocolate packing factory [3]. It has since become a particularly popular robot in industrial settings due to its good precision coupled with high speed and acceleration.

The Delta robot is a specific type of robot known as a **parallel manipulator**. Parallel manipulators, also known as parallel robots, are a class of robots that feature end-effectors driven by multiple underactuated parallel kinematic chains [4, 5]. Typically a parallel robot is designed such that all actuators remain fixed to the support structure of the robot thereby minimising the mass of the moving parts of the robot and enabling very fast accelerations. Indeed this goal of high speed/fast acceleration has been the primary driving force in the development of parallel robots for industry and today architectures such as the Delta robot are widespread in many high precision, high throughput manufacturing applications. Parallel robots, while practically very useful, are often significantly more difficult to analyse than their serial cousins due to the end-point position being a function of the configuration of multiple kinematic chains.

Conformal Geometric Algebra (CGA) is a specific 5D representation of 3D space that embeds geometric primitives and conformal transformations as elements of the same algebra [6, 7]. Our mathematics in this paper will be phrased within CGA. We will use the standard extension of the 3D geometric algebra, where our 5D CGA space is made up of the standard Euclidean basis vectors $\{e_j\}$ $j = 1, 2, 3$, where $e_j^2 = 1$, plus two additional basis vectors, e and $\bar{e}$ with signatures, $e^2 = 1$, $\bar{e}^2 = -1$. Two *null vectors* can therefore be defined as: $n_\infty = e + \bar{e}$

and $n_0 = \frac{\bar{e}-e}{2}$. The mapping of a 3D vector x to its conformal representation X is given by $X = F(x) = \frac{1}{2}(x^2 n_\infty + 2x + 2n_0)$.

In this paper we draw inspiration from the established literature of robotic analysis in GA [12–20] as well as the previous literature on Delta robot kinematics [5, 8, 11] and combine them, leveraging the representational power of CGA to illustrate the geometry of the constraints inherent in the mechanism of the Delta robot.

2 Geometry of a Delta Robot

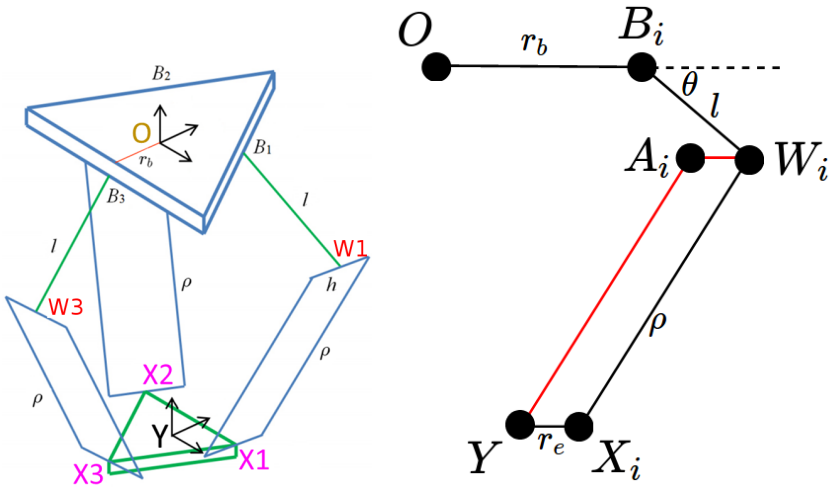

Fig. 1. Left: the 3D geometry of the delta robot. Right: The geometry of a single arm in plane.

Since its inception, there have been many variants of the Delta robot [8]. In this paper we will assume the simple robot described in this section and illustrated in Figure 1. The static part of the robot is a base plate to which three motors are rigidly attached, we will assume a space in which the origin is at the centre of this plate. Each motor shaft is rigidly attached to an 'upper arm' of length l; we will number each upper arm $i \in [1, 2, 3]$. The connection point of the motor and upper arm will be labelled B_i. The arm can only rotate in plane about the motor axis as the motor shaft and upper arm are rigidly connected. We will refer to the other end of this upper arm as the 'elbow point' and will label it W_i. At the elbow point each arm is rigidly attached to a central point of a horizontal rod we will refer to as the 'elbow rod'. At each end of the elbow rod a ball joint connects to a 'forearm' piece. The two forearm pieces for each arm are the same length and, at the other end from the elbow rod, are connected to a rigid plate that we will refer to as the end-effector plate. The point half-way between where the two forearm rods connect to the end-effector

plate is labelled X_i. We will label the point at the centre of the end-effector plate Y. Assuming the robot is infinitely stiff, the end plate is constrained, due to this specific arrangement of the forearms, to always remain parallel to the base plate and to have its in-plane orientation fixed as well. The Delta robot is therefore a purely translational mechanism.

The labels have assigned here will also serve as the notation for points in conformal space; for example, the centre of the end-effector plate has 3D position, y with $Y = F(y)$.

3 Inverse Kinematics

The inverse kinematic problem for the Delta robot is summarised as follows: To what angle relative to the base should we move the upper arms given we want the centre of the end-effector plate to be in a specific position in 3D space?

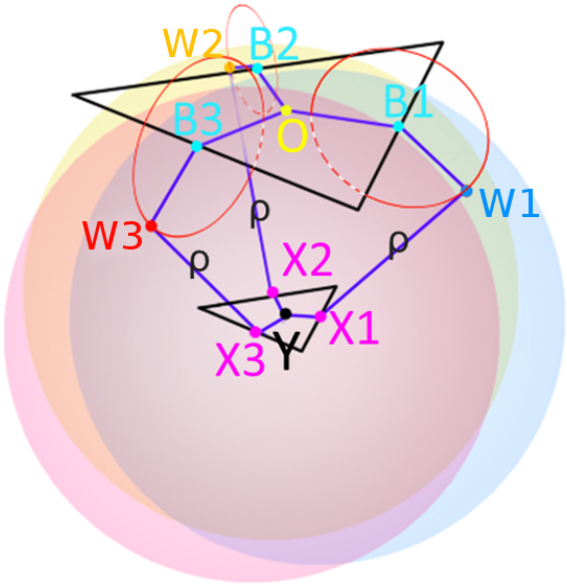

Fig. 2. The geometry of the inverse kinematic problem. There are three spheres, one for each arm of the robot, centred at the conformal points X_i. Each sphere intersects with a circle centred at B_i allowing the extraction of the conformal elbow point W_i.

To solve this problem we need to work backwards from the 3D end-effector plate position y to the motor angles θ_i considering the geometry of the robot as we go. Starting at the end-effector plate the 3D points x_i are translationally offset in plane in the direction s_i giving $x_i = y + r_e s_i$ where r_e is the radius of the end-effector plate. Due to the geometry of the robot the elbow point W_i is constrained to lie on a sphere with radius equal to the length of the forearms

ρ centred at this point x_i. We can represent this sphere as a dual sphere in conformal geometric algebra as follows:

$$\Sigma_i^* = X_i - \frac{1}{2}\rho^2 n_\infty$$

The elbow point is also simultaneously constrained to lie on a circle of radius l centred at the motor shaft to upper-arm joint, B_i. We can represent this circle as the dual circle C_i^* in CGA, where C_i^* is the intersection of a dual sphere of radius l centred at the position B_i, $\left(B_i - \frac{1}{2}l^2 n_\infty\right)$, and the dual plane through the origin, B_i and e_3 which is given by $I_3(s_i \wedge e_3)$. In CGA we calculate the intersection of objects via the 'meet' operator, as both operands are in their dual form however, here we simply need an outer product:

$$C_i^* = \left(B_i - \frac{1}{2}l^2 n_\infty\right) \wedge (I_3(s_i \wedge e_3))$$

where e_3 is the vertical unit vector. So long as y is within the reachable volume of the robot there are two possible solutions for this pair of constraints. These two solutions lie at the intersection points of the sphere and circle and the 'meet' operation of CGA provides us with a direct means to calculate these intersection points. As with the circle, our sphere and circle are in the dual form (ie. 1 and 2-vectors respectively), and so the point-pair bivector resulting from their meet is calculated as simply their outer product followed by multiplication with the 5D pseudo-scalar, I_5:

$$T_i = (C_i^* \wedge \Sigma_i^*)I_5$$

The individual solutions can be extracted from this point-pair object by projection operators [6]:

$$P_i = \frac{1}{2}\left(1 + \frac{T_i}{\sqrt{T_i^2}}\right)$$
$$W_i = -\tilde{P}_i(T_i \cdot n_\infty)P_i$$

We can then convert from the CGA to the 3D vector point:

$$w_i = -\frac{\sum_{j=1}^{j=3}(W_i \cdot e_j)e_j}{W_i \cdot n_\infty}$$

and so, with a little trigonometry we can extract the motor angles:

$$\theta_i = \text{atan2}(z_i \cdot e_3, z_i \cdot s_i), \qquad z_i = w_i - r_b s_i$$

Figure 2 illustrates the geometry of the inverse kinematic problem graphically.

4 Forward Kinematics

The forward kinematic problem is, in some sense, the opposite of the inverse kinematic one. Our goal here is to calculate the 3D vector position of the end-effector plate y given the motor angles $\theta_i, i \in [1, 2, 3]$.

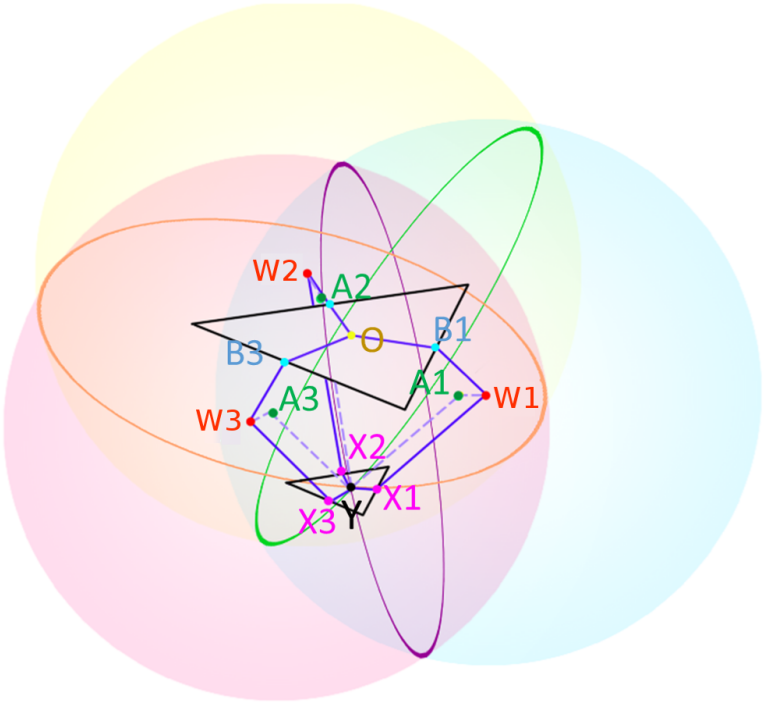

Fig. 3. The geometry of the forward kinematic problem. Each motor connects to an upper arm at position B_i. The upper arms end in the elbow point W_i. Each elbow point has an associated pseudo-elbow point A_i and forearm constraint sphere. All three constraint spheres meet at the centre of the end-point plate, Y.

To solve the forward kinematic problem we will consider the robot one arm at a time. For a given arm motor angle θ_i the 3D position of the elbow point w_i can be calculated as:

$$w_i = (r_b + l\cos(\theta_i))s_i + l\sin(\theta_i)e_3$$

For each arm we will now define a pseudo-elbow point, a_i which is offset horizontally from the true elbow point by the radius of the end effector plate and in the direction of the origin.

$$a_i = (r_b - r_e + l\cos(\theta_i))s_i + l\sin(\theta_i)e_3$$

The equivalent CGA point is then:

$$A_i = \frac{1}{2}a_i^2 n_\infty + a_i + n_0$$

Given the geometry of the robot, these pseudo-elbow points all lie a distance equal to the length of the robot's forearms, ρ, from the centre of the end-point plate Y. Geometrically these fixed distance constraints manifest themselves as

spheres, which we will label Σ_i, on which the centre of the end-point plate can lie:

$$\Sigma_i^* = A_i - \frac{1}{2}\rho^2 n_\infty$$

Each arm contributes one constraint sphere and the intersection of the three spheres produces a point-pair, T, that represents the two possible configurations of the end-plate:

$$T = I_5 \bigwedge_{i=1}^{i=3} \Sigma_i^*$$

where the $\bigwedge$ notation implies an outer product of all elements following it.

Practically only one of these possible solutions is feasible, the solution which places Y at a greater position along the e_3 axis. We can once again get the 3D position of this point, y by extracting the point with projectors.

$$P = \frac{1}{2}\left(1 + \frac{T}{\sqrt{T^2}}\right)$$

$$y = \frac{-\sum_{j=1}^{j=3}(Y \cdot e_j)e_j}{Y \cdot n_\infty}, \qquad Y = -\tilde{P}(T \cdot n_\infty)P,$$

Figure 3 illustrates the geometry of the forward kinematic problem.

5 The Inverse Jacobian

Knowing static kinematic solutions is useful but to do more advanced analysis of the Delta robot mechanism we need to look at derivatives. We will start with inverse kinematics and ask ourselves the question, if the end-point plate moves with a specific velocity what speeds must the motors be moving at to be compatible?

First we will write the 3D end-point plate position as a linear combination of basis vectors with coefficients denoted α_j, $j \in 1, 2, 3$:

$$y = \alpha_1 e_1 + \alpha_2 e_2 + \alpha_3 e_3$$

Our goal now becomes to calculate the partial derivative of each motor angle with respect to one each of these α coefficients. Taking partial derivatives of y with respect to one of the α_j coefficients trivially gives:

$$\frac{\partial y}{\partial \alpha_j} = e_j$$

For now we do not need to worry about **which** α parameter we are taking derivatives with respect to, so we will leave the derivative of the end-point written as $\frac{\partial y}{\partial \alpha}$. Using this notation, our ultimate goal in this section is to find an equation for the partial derivative of a given motor angle θ_i with respect to α, ie. $\frac{\partial \theta_i}{\partial \alpha}$. To find $\frac{\partial \theta_i}{\partial \alpha}$ we select a specific robot arm i and work back through its joints from the end-point.

The first joint position of interest is x_i, we saw in section 3 that:

$$x_i = y + r_e s_i$$

Taking partial derivatives gives:

$$\frac{\partial x_i}{\partial \alpha} = \frac{\partial y}{\partial \alpha}$$

The 3D point x_i can then be represented as the CGA point X_i:

$$X_i = \frac{1}{2} x_i^2 n_\infty + x_i + n_0$$

The derivative of this CGA point is then easily found:

$$\frac{\partial X_i}{\partial \alpha} = \left(\frac{\partial x_i}{\partial \alpha} \cdot x_i \right) n_\infty + \frac{\partial x_i}{\partial \alpha}$$

We then form the dual constraint sphere:

$$\Sigma_i^* = X_i - \frac{1}{2} \rho^2 n_\infty$$

which, as the radius is fixed, has partial derivative:

$$\frac{\partial \Sigma_i^*}{\partial \alpha} = \frac{\partial X_i}{\partial \alpha}$$

As we saw in the previous section, the intersection of this dual constraint sphere Σ_i^* and the dual circle C_i^* centred on the motor shaft produces a point-pair T_i that represents the two possible elbow positions for that arm:

$$\begin{aligned} C_i^* &= \left(B_i - \frac{1}{2} l^2 n_\infty \right) \wedge (I_3 (s_i \wedge e_3)) \\ T_i &= (\Sigma_i^* \wedge C_i^*)^* \end{aligned}$$

The outer product and taking the dual are both linear, which means that taking derivatives is particularly easy here:

$$\frac{\partial T_i}{\partial \alpha} = \left(\frac{\partial \Sigma_i^*}{\partial \alpha} \wedge C_i^* \right)^*$$

Of course the elbow can only actually be in one position which we can extract via projection operators:

$$P_i = \frac{1}{2} \left(1 + \frac{T_i}{\sqrt{T_i^2}} \right), \qquad \frac{\partial P_i}{\partial \alpha} = \frac{1}{2 T_i^2} \left(\sqrt{T_i^2} \frac{\partial T_i}{\partial \alpha} - T_i \frac{\frac{\partial T_i}{\partial \alpha} \cdot T_i}{\sqrt{T_i^2}} \right)$$

$$W_i = -\tilde{P}_i (T_i \cdot n_\infty) P_i$$

$$\frac{\partial W_i}{\partial \alpha} = -\frac{\partial \tilde{P}_i}{\partial \alpha} (T_i \cdot n_\infty) P_i - \tilde{P}_i \left(\frac{\partial T_i}{\partial \alpha} \cdot n_\infty \right) P_i - \tilde{P}_i (T_i \cdot n_\infty) \frac{\partial P_i}{\partial \alpha}$$

We can then convert from the CGA to the 3D vector point:

$$w_i = -\frac{\sum_{j=1}^{j=3}(W_i \cdot e_j)e_j}{W_i \cdot n_\infty}$$

$$\frac{\partial w_i}{\partial \alpha} = \frac{-\sum_{j=1}^{j=3}\left(\frac{\partial W_i}{\partial \alpha} \cdot e_j\right) e_j (W_i \cdot n_\infty) + \sum_{j=1}^{j=3}(W_i \cdot e_j)\, e_j \left(\frac{\partial W_i}{\partial \alpha} \cdot n_\infty\right)}{(W_i \cdot n_\infty)^2}$$

and use this to form the derivative of the motor angles with regard to α:

$$z_i = w_i - r_b s_i, \qquad \frac{\partial z_i}{\partial \alpha} = \frac{\partial w_i}{\partial \alpha}$$

$$\theta_i = \text{atan2}(z_i \cdot e_3, z_i \cdot s_i)$$

$$\frac{\partial \theta_i}{\partial \alpha} = \frac{z_i \cdot s_i}{|z_i \cdot s_i|} \frac{(z_i \cdot s_i)\left(\frac{\partial z_i}{\partial \alpha} \cdot e_3\right) - (z_i \cdot e_3)\left(\frac{\partial z_i}{\partial \alpha} \cdot s_i\right)}{z_i^2}$$

This finally gives us an expression for the derivative of the motor angle with respect to the α of the endpoint. Typically in engineering scenarios we would construct a matrix of the partial derivatives with respect to α_j, $j \in 1, 2, 3$, known as the Jacobian matrix:

$$J^* = \begin{bmatrix} \frac{\partial \theta_1}{\partial \alpha_1} & \frac{\partial \theta_1}{\partial \alpha_2} & \frac{\partial \theta_1}{\partial \alpha_3} \\ \frac{\partial \theta_2}{\partial \alpha_1} & \frac{\partial \theta_2}{\partial \alpha_2} & \frac{\partial \theta_2}{\partial \alpha_3} \\ \frac{\partial \theta_3}{\partial \alpha_1} & \frac{\partial \theta_3}{\partial \alpha_2} & \frac{\partial \theta_3}{\partial \alpha_3} \end{bmatrix}$$

This matrix can then be used to convert an end-point velocity vector to a set of motor velocities:

$$\begin{bmatrix} \frac{\partial \theta_1}{\partial t} \\ \frac{\partial \theta_2}{\partial t} \\ \frac{\partial \theta_3}{\partial t} \end{bmatrix} = J^* \begin{bmatrix} \frac{\partial \alpha_1}{\partial t} \\ \frac{\partial \alpha_2}{\partial t} \\ \frac{\partial \alpha_3}{\partial t} \end{bmatrix}$$

As it is the Jacobian matrix for the inverse kinematic problem, this matrix is specifically labelled the inverse Jacobian matrix.

6 The Forward Jacobian

Many problems in robotics require us to take derivatives of the forward kinematic equations. Specifically, we need to know the end-point plate velocity as a function of the motor speeds.

Our forward kinematic solution begins with calculating the position of the elbow point for a given arm i:

$$w_i = (r_b + l\cos(\theta_i))s_i + l\sin(\theta_i)e_3, \qquad \frac{\partial w_i}{\partial \theta_i} = -l\sin(\theta_i)s_i + l\cos(\theta_i)e_3$$

With the elbow point we can then calculate the pseudo-elbow point:

$$a_i = w_i - r_e s_i, \qquad \frac{\partial a_i}{\partial \theta_i} = \frac{\partial w_i}{\partial \theta_i}$$

We then convert the pseudo-elbow to a CGA point:

$$A_i = \frac{1}{2}a_i^2 n_\infty + a_i + n_0, \qquad \frac{\partial A_i}{\partial \theta_i} = \left(\frac{\partial a_i}{\partial \theta_i} \cdot a_i\right) n_\infty + \frac{\partial a_i}{\partial \theta_i}$$

The forearm length dual constraint sphere can then be constructed about the pseudo-elbow point

$$\Sigma_i^* = A_i - \frac{1}{2}\rho^2 n_\infty, \qquad \frac{\partial \Sigma_i^*}{\partial \theta_i} = \frac{\partial A_i}{\partial \theta_i}$$

The intersection of all three constraint spheres, one from each arm, produces the point pair on which the solution lies.

$$T = (\Sigma_1 \vee \Sigma_2 \vee \Sigma_3) \equiv I_5(\Sigma_1^* \wedge \Sigma_2^* \wedge \Sigma_3^*)$$

We can take derivatives of this point-pair with respect to each of the motor angles:

$$\frac{\partial T}{\partial \theta_1} = I_5\left(\frac{\partial \Sigma_1^*}{\partial \theta_1} \wedge \Sigma_2^* \wedge \Sigma_3^*\right), \qquad \frac{\partial T}{\partial \theta_2} = I_5\left(\Sigma_1^* \wedge \frac{\partial \Sigma_2^*}{\partial \theta_2} \wedge \Sigma_3^*\right)$$
$$\frac{\partial T}{\partial \theta_3} = I_5\left(\Sigma_1^* \wedge \Sigma_2^* \wedge \frac{\partial \Sigma_3^*}{\partial \theta_3}\right)$$

We can re-write these derivatives as follows:

$$\frac{\partial T}{\partial \theta_i} = (-1)^{i-1} I_5\left(\frac{\partial \Sigma_i^*}{\partial \theta_i} \wedge C^*\right), \quad \text{where } C^* = \bigwedge_{j\in 1,2,3 \;\; j\neq i} \Sigma_j^* \tag{1}$$

Practically, when we take partial derivatives with respect to one θ at a time we are effectively freezing two of the motors in position and moving the third. Geometrically, this process forces the end-point plate to move along the surface of the circle formed by the intersection of the two constraint spheres centred at the pseudo-elbow points of the frozen motors.

Figure 4 shows the geometric significance of Equation 1. To get the end-point plate position we again extract one end of the point-pair T:

$$P = \frac{1}{2}\left(1 + \frac{T}{\sqrt{T^2}}\right), \qquad \frac{\partial P}{\partial \theta_i} = \frac{1}{2T^2}\left(\sqrt{T^2}\frac{\partial T}{\partial \theta_i} - T\frac{\frac{\partial T}{\partial \theta_i} \cdot T}{\sqrt{T^2}}\right)$$

$$Y = -\tilde{P}(T \cdot n_\infty)P, \qquad \frac{\partial Y}{\partial \theta_i} = -\frac{\partial \tilde{P}}{\partial \theta_i}(T \cdot n_\infty)P - \tilde{P}\left(\frac{\partial T}{\partial \theta_i} \cdot n_\infty\right)P - \tilde{P}(T \cdot n_\infty)\frac{\partial P}{\partial \theta_i}$$

Finally we convert our end-point back to a 3D point:

$$y = \frac{-\sum_{j=1}^{j=3}(Y \cdot e_j)e_j}{Y \cdot n_\infty}$$

$$\frac{\partial y}{\partial \theta_i} = \frac{-\sum_{j=1}^{j=3}\left(\frac{\partial Y}{\partial \theta_i} \cdot e_j\right)e_j(Y \cdot n_\infty) + \sum_{j=1}^{j=3}(Y \cdot e_j)\,e_j(\frac{\partial Y}{\partial \theta_i} \cdot n_\infty)}{(Y \cdot n_\infty)^2}$$

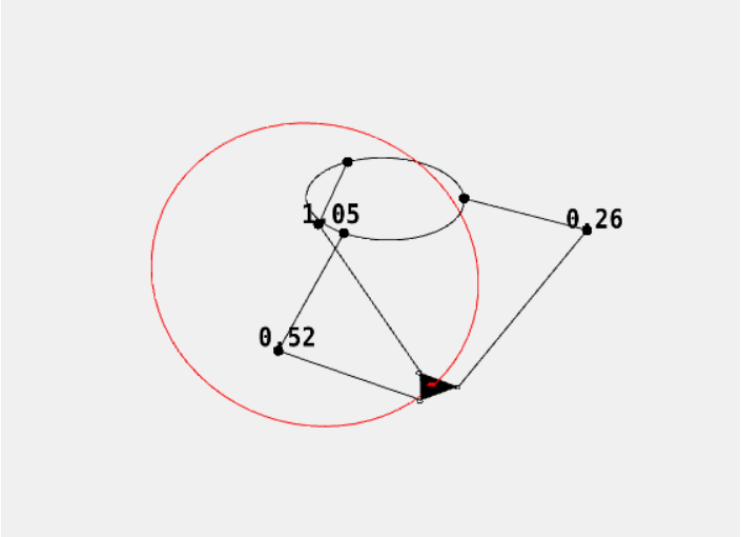

Fig. 4. With two limbs frozen the end-point plate is constrained to move such that its centre always lies on the circle (shown in red) formed from the intersection of the other two limbs' constraint spheres.

We can write the end-point plate position as:

$$y = \alpha_1 e_1 + \alpha_2 e_2 + \alpha_3 e_3, \qquad \frac{\partial y}{\partial \theta_i} = \frac{\partial \alpha_1}{\partial \theta_i} e_1 + \frac{\partial \alpha_2}{\partial \theta_i} e_2 + \frac{\partial \alpha_3}{\partial \theta_i} e_3$$

With $\frac{\partial y}{\partial \theta_i}$ we are therefore in a position to build the forward Jacobian matrix:

$$J = \begin{bmatrix} \frac{\partial y}{\partial \theta_1} \cdot e_1 & \frac{\partial y}{\partial \theta_2} \cdot e_1 & \frac{\partial y}{\partial \theta_3} \cdot e_1 \\ \frac{\partial y}{\partial \theta_1} \cdot e_2 & \frac{\partial y}{\partial \theta_2} \cdot e_2 & \frac{\partial y}{\partial \theta_3} \cdot e_2 \\ \frac{\partial y}{\partial \theta_1} \cdot e_3 & \frac{\partial y}{\partial \theta_2} \cdot e_3 & \frac{\partial y}{\partial \theta_3} \cdot e_3 \end{bmatrix}$$

The inverse Jacobian matrix and the forward Jacobian matrix are, as the names suggest, inverse to each other.

$$JJ^* = I$$

7 Simulation and Verification in Python and Unity3D

To verify our derivatives we implemented the above mathematics in the Clifford Python package [10] and tested it against a central differences approximation.

With the mathematics verified we constructed a CGA Unity3D library based on the C# output of the ganja.js [9] code generator. Figure 5 shows examples of this library being used to intersect circles, lines, spheres and planes in Unity3D. Alongside the CGA library we also constructed a physical delta robot model using Unity3D's built-in joint system and physics system, the right hand side of

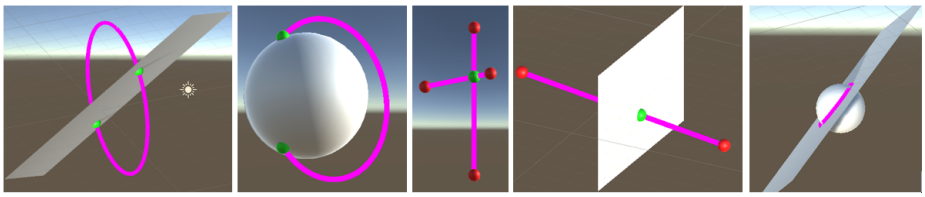

Fig. 5. Intersections of geometric primitives calculated and visualised with the Unity3D CGA library

Figure 6 shows the model in the Unity3D scene view. The simulated physical model of the robot allows us to design and tune a controller safely and for low cost without requiring us to build or buy a real Delta robot.

The CGA Unity3D library and simulated physical robot together form a test-bed for the design and tuning of control systems. On the left hand side of Figure 6 is the representation of the robot to a controller constructed using the CGA library and the mathematics described in the previous sections.

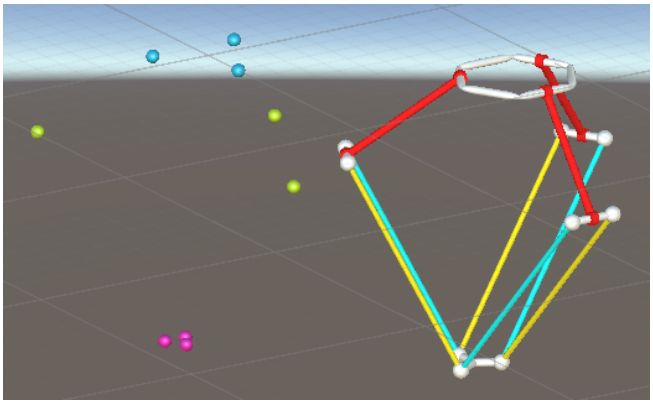

Fig. 6. Left: The internal robot geometry model of the controller. Right: The physically simulated Delta robot in Unity3D.

8 Conclusion

In this paper we have derived, step-by-step, the forward and inverse kinematic solutions for a Delta robot in CGA as well as the forward and inverse Jacobian matrices. We then implemented the mathematics in Python and C# before simulating a physical robot model in Unity3D and using our mathematics to design a basic position and velocity control system to position the end-plate of the robot.

References

1. Clavel, R. Conception d'un robot parallèle rapide à 4 degrés de liberté. Infoscience `https://infoscience.epfl.ch/record/31403` (1991) doi:10.5075/epfl-thesis-925.

2. Clavel, R., Device for the Movement and Positioning of an Element in Space, United States Patent No. 4976582, (1990)
3. Pessina, L., Reymond Clavel, creator of the Delta Robot, reflects on his career, EPFL website, https://sti.epfl.ch/reymond-clavel-creator-of-the-delta-robot-reflects-on-his-career/
4. Gallardo-Alvarado, J. Kinematic Analysis of Parallel Manipulators by Algebraic Screw Theory. (Springer International Publishing, 2016).
5. Merlet, J.-P. Parallel robots. (Kluwer Academic Publishers, 2006).
6. Lasenby, A., Lasenby, J. and Wareham, R.: A Covariant Approach to Geometry Using Geometric Algebra., CUED Technical Report CUED/F-INFENG/TR-483. Cambridge. University Engineering Department (2004).
7. Dorst, L., Fontijne, D. and Mann, S.: Geometric algebra for computer science: an object-oriented approach to geometry. 1st edn. Elsevier; Morgan Kaufmann, Amsterdam (2007).
8. Pierrot, F., Fournier, A. and Dauchex, P. Towards a fully-parallel 6 DOF robot for high-speed applications. in 1991 IEEE International Conference on Robotics and Automation Proceedings 1288–1293 vol.2 (1991).
9. De Keninck, S.: ganja.js https://github.com/enkimute/ganja.js (2017).
10. Arsenovic, A., Hadfield, H., Wieser, E., Kern, R., and The Pygae Team. Clifford (Version v1.3.0). (2020, May 29). https://github.com/pygae/clifford. Zenodo. http://doi.org/10.5281/zenodo.3865446
11. Zsombor-Murray, P. J. Descriptive Geometric Kinematic Analysis of Clavel's "Delta" Robot. Centre for Intelligent Machines, McGill University. (2004).
12. Zamora, J. and Bayro-Corrochano, E. Inverse kinematics, fixation and grasping using conformal geometric algebra. in 2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (IEEE Cat. No.04CH37566) vol. 4 3841–3846 vol.4 (2004).
13. Hildenbrand, D., Zamora, J. and Bayro-Corrochano, E. Inverse Kinematics Computation in Computer Graphics and Robotics Using Conformal Geometric Algebra. Advances in Applied Clifford Algebras 18, 699–713 (2008).
14. Aristidou, A. Tracking and modelling motion for biomechanical analysis (Doctoral thesis). https://doi.org/10.17863/CAM.13996. (2010).
15. Kleppe, A. L. and Egeland, O. Inverse Kinematics for Industrial Robots using Conformal Geometric Algebra. Modeling, Identification and Control: A Norwegian Research Bulletin 37, 63–75 (2016).
16. Kim, J. S., Jeong, J. H. and Park, J. H. Inverse kinematics and geometric singularity analysis of a 3-SPS/S redundant motion mechanism using conformal geometric algebra. Mechanism and Machine Theory 90, 23–36 (2015).
17. Fu, Z., Yang, W. and Yang, Z. Solution of Inverse Kinematics for 6R Robot Manipulators With Offset Wrist Based on Geometric Algebra. Journal of Mechanisms and Robotics 5, (2013).
18. Tichý, R. Inverse Kinematics for the Industrial Robot IRB4400 Based on Conformal Geometric Algebra. in Modelling and Simulation for Autonomous Systems (eds. Mazal, J., Fagiolini, A. and Vasik, P.) 148–161 (Springer International Publishing, 2020). https://doi.org/10.1007/978-3-030-43890-6_12.
19. Hildenbrand, D., Hrdina, J., Návrat, A. and Vašík, P. Local Controllability of Snake Robots Based on CRA, Theory and Practice. Adv. Appl. Clifford Algebras 30, 2 (2019).
20. Selig, J. M. Geometric Fundamentals of Robotics. (Springer-Verlag, 2005). https://doi.org/10.1007/b138859.