

# Cooperative Task Spaces for Multi-Arm Manipulation Control based on Similarity Transformations

Journal Title  
XX(X):1–25  
©The Author(s) 0000  
Reprints and permission:  
sagepub.co.uk/journalsPermissions.nav  
DOI: 10.1177/ToBeAssigned  
www.sagepub.com/

SAGE

Tobias Löw<sup>1 2</sup>, Cem Bilaloglu<sup>1 2</sup>, Sylvain Calinon<sup>1 2</sup>

## Abstract

Many tasks in human environments require collaborative behavior between multiple kinematic chains, either to provide additional support for carrying big and bulky objects or to enable the dexterity that is required for in-hand manipulation. Since these complex systems often have a very high number of degrees of freedom, coordinating their movements is notoriously difficult to model. In this article, we present the derivation of the theoretical foundations for cooperative task spaces of multi-arm robotic systems based on geometric primitives defined using conformal geometric algebra. Based on the similarity transformations of these cooperative geometric primitives, we derive an abstraction of complex robotic systems that enables representing these systems in a way that directly corresponds to single-arm systems. By deriving the associated analytic and geometric Jacobian matrices, we then show the straightforward integration of our approach into classical control techniques rooted in operational space control. We demonstrate this using bimanual manipulators, humanoids and multi-fingered hands in optimal control experiments for reaching desired geometric primitives and in teleoperation experiments using differential kinematics control. We then discuss how the geometric primitives naturally embed nullspace structures into the controllers that can be exploited for introducing secondary control objectives. This work represents the theoretical foundations of this cooperative manipulation control framework, and thus the experiments are presented in an abstract way, while giving pointers towards potential future applications.

## Keywords

Conformal Geometric Algebra

## 1 Introduction

Humans excel at dexterous manipulation using the redundancy of their arms and hands to achieve complex tasks where cooperative behaviors between the different kinematic chains is required. Many environments are built around these advanced manipulation capabilities, and robots that are increasingly operating within them. Yet, it still remains challenging for robots to achieve dexterous manipulation that is on par with humans (Kadalagere Sampath, N. Wang, H. Wu, and Yang 2023). Nevertheless, robots are now expected to handle various tasks that require them to interact with objects in a way that goes beyond traditional parallel jaw grippers or suction cups that are often featured on single arm manipulators. These types of end-effectors are limiting the robot manipulation capabilities, since many objects are often more complex and require different strategies. For instance, big and bulky objects may call for dual-arm manipulators for picking and placing. This kind of coordination between two arms is vital when grasping, lifting, and transferring objects of varying shape, weight, and fragility. This is similarly true for intricate tasks such as in-hand manipulating and multi-fingered grasping using robotics hands. The common ground here is coordinating the movements of two or more parallel kinematic chains to interact with the environment during manipulation tasks.

However, many manipulation tasks do not require exploiting the full redundancy of the system. This is because the kinematic chains are not acting independently but instead are coordinated to achieve a shared objective. As

a result, the effective behavior of the system often lies on a lower-dimensional manifold embedded within the full configuration space. We refer to these manifolds exhibiting specific geometric structures as cooperative task spaces. Recognizing and leveraging this structure not only reduces the dimensionality of the control or planning problems but also improves interpretability. Moreover, these geometric structures naturally give rise to geometric nullspaces, which can be exploited for secondary objectives, such as regulating contact forces, without interfering with the primary task objective, as we previously demonstrated in (Bilaloglu, Löw, and Calinon 2025).

Existing work using dual quaternions (Adorno, Fraisse, and Druon 2010) and geometric algebra (Löw and Calinon 2024) has begun to explore cooperative task spaces, particularly in the context of dual-arm manipulation. However, the notion of cooperation extends well beyond dual-arm settings. Many real-world scenarios involving multiple kinematic chains, such as torso-arm-hand coordination or multi-fingered in-hand manipulation, can benefit from a unified geometric framework that generalizes cooperative control.

<sup>1</sup>Idiap Research Institute, Martigny, Switzerland

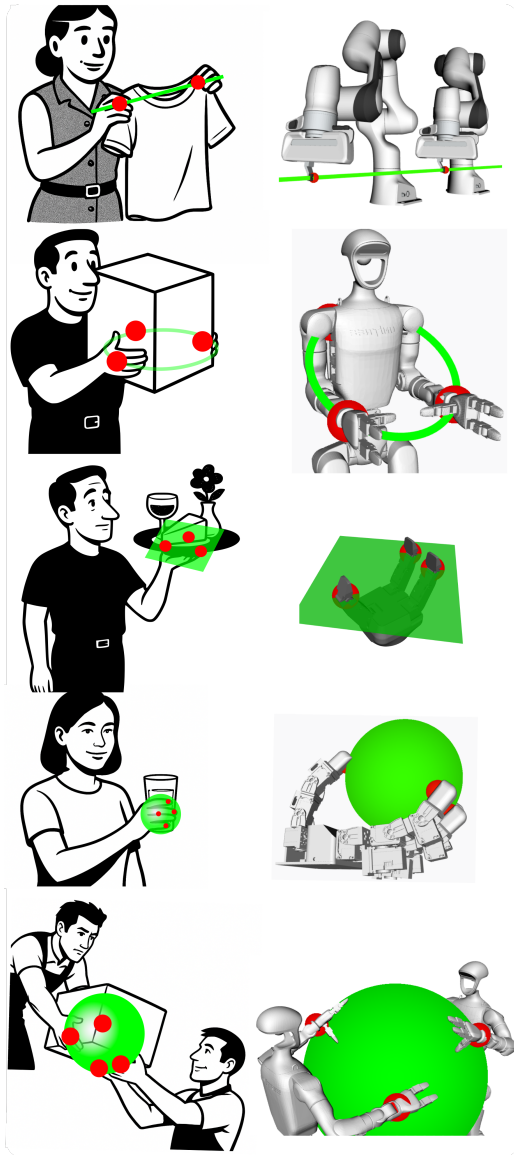
<sup>2</sup>EPFL, Lausanne, Switzerland

## Corresponding author:

Tobias Löw, Idiap Research Institute, EPFL, Switzerland.

Email: tobias.loew@epfl.ch

To illustrate this, we provide a non-exhaustive collection of robotic systems and manipulation challenges that can be abstracted through cooperative subspaces corresponding to geometric primitives, as shown in Figure 1. The geometric primitives only depend on the number of kinematic chains that are involved in the task. Hence, they model the cooperative behavior, and are not chosen based on an approximation of the object shape. Our overarching goal is to generalize operational space control (Khatib 1987), to systems with multiple kinematic chains by representing cooperative task spaces as geometric primitives within the control architecture.



**Figure 1.** Overview of various daily-life tasks and how they can be modeled using the cooperative geometric primitives. Note that the geometric primitives do not approximate the shape of the objects, but instead model the cooperative behavior of the kinematic chains. The corresponding reference points are shown in red. *Left:* Human activities with an overlay of the proposed geometric primitive to capture the task constraints. *Right:* Corresponding robotic system with the cooperative geometric primitive achieving the same behavior.

In this article, we address the challenge of cooperative manipulation control for robotic systems involving multiple parallel kinematic chains. We do so by combining the various ideas around cooperative control, grasping nullspaces and geometric primitives, into a single, mathematically coherent framework by introducing cooperative geometric primitives and integrating them into geometrically consistent control schemes. For this, we leverage conformal geometric algebra (CGA) to find algebraic objects that represent geometric primitives that go beyond keypoints. In prior work, we have shown how these primitives define geometric nullspaces and how they can be used in optimal control (L w and Calinon 2023). In the same optimal control framework, we have then extended the cooperative dual-task space using CGA for bimanual robots (L w and Calinon 2024). Here, we further extend this idea, by exploring how cooperative geometric primitives enable decentralized control strategies that harmonize task execution, disturbance rejection, and inter-agent coordination. We show that by exploiting CGA, different systems can be modelled in a uniform manner that allows informed collaboration strategies based on the geometric primitives and conformal transformations that can be expressed in this algebra. Our contributions therefore are

- A geometric representation for cooperative behaviors of multiple parallel kinematic chains using geometric primitives, enabling intuitive modeling of coordinated manipulation tasks;
- Introduction of similarity transformations of geometric primitives, extending classical rigid body transformations from single end-effector control to multi-chain cooperative manipulation;
- A unified mathematical framework for cooperative control based on similarity transformations, that can be used with optimal control or differential kinematics formulations;
- Exploitation of geometric nullspaces inherent in the cooperative task space for achieving secondary control objectives, such as contact force regulation or redundancy resolution, without affecting primary task execution;
- Demonstration of the versatility and scalability of the proposed approach on complex robotic systems, including humanoid platforms and multi-fingered hands, through tasks involving cooperative reaching and teleoperation.

### 1.1 Related Work

A common example for highlighting the challenges of cooperative manipulation is the collective transport of big and bulky objects that cannot be handled by a single robot. This task is very challenging, since multiple robots interact with the same object via local control actions (Erhart, Sieber, and Hirche 2013), which requires them to cooperate in order to achieve a common, global goal. Traditional approaches often use a leader-follower strategy, where the different roles are predefined, and the agents communicate explicitly. However, these strategies often fail due to uncertainties stemming from the task parameters, environmental dynamics, or the team composition (Farivarnejad and Berman 2022). Solving these

issues requires strategies that do not rely on centralized coordination, but instead use the manipulated object as a physical channel for implicit coordination. By sensing the exerted forces on the shared payload, the robots can decompose the motion into an allowed task space motion and a nullspace motion that is used rejecting perturbations (Carey and Werfel 2024). This strategy minimizes force conflicts, and enables cooperative manipulation even under strict communication constraints. Recent advances in cooperative control favor frameworks that further emphasize distributed optimization and adaptability. They optimize the individual performance of the robots in the system that suffers reduced operation capabilities due to the physical connection through grasping (He, M. Wu, and Liu 2022). Furthermore, the optimization can include objectives for enhancing the manipulability, avoiding obstacles, and reducing internal forces caused by robot motion errors due to coupling (He, M. Wu, and Liu 2020). Hence, there is an inherent trade-off between physical coupling (e.g. rigid grasps) and operational flexibility, which makes it important to deal with disturbances to avoid unexpected behaviors (Aladele, De Cos, Dimarogonas, and Hutchinson 2022). Learning-based approaches try to simplify the problem of coordinating multiple agents by introducing actions through shared latent spaces, which reduces the sample complexity of high-dimensional tasks like dual-arm manipulation (Aljalbout and Karl 2023). Alternatively, bi-manual end-effector poses are learned from demonstrations using task-parameterized dynamical systems (Silverio, Roza, Calinon, and Caldwell 2015).

Dynamic Movement Primitives (DMPs) have emerged as a powerful way to encode complex trajectories while preserving stability guarantees. A comprehensive tutorial survey frames DMPs as a bridge between model-based control and data-driven learning, highlighting their suitability for multi-robot settings where each agent can generate locally consistent motions that nevertheless respect a globally defined attractor (Saveriano, Abu-Dakka, Kramberger, and Peternel 2023). Building on this foundation, it has been demonstrated that DMPs can be combined with velocity scaling to personalise human-robot collaborative transport. By modulating the DMP speed according to the measured interaction forces, the system achieves smooth, safe hand-overs even when the human operator varies the payload weight or trajectory on-the-fly (Franceschi, Bussolan, Pomponi, Avram, Baraldo, and Valente 2025). Other work has leveraged human motion prediction based on DMPs to enable seamless collaborative object transfer, showing that anticipating the human partner's trajectory improves coordination and reduces interaction forces (Sidiropoulos, Karayiannidis, and Doulgieri 2019). The sensed forces on the shared object become the scaling signal for the DMPs, thereby turning the payload itself into a communication medium that implicitly synchronises the robots' motion. While many cooperative manipulation studies assume rigid payloads, real-world logistics increasingly involve deformable or soft objects (e.g., bags of grain, flexible containers). Movement-Primitive Diffusion introduced a generative framework that learns gentle manipulation policies for deformable items directly from demonstrations (Scheikl, Schreiber, Haas, Freymuth,

Neumann, Lioutikov, and Mathis-Ullrich 2024). The diffusion model produces smooth DMP parameters that respect material compliance, effectively reducing internal stresses that would otherwise arise from hard-grasping strategies. This approach complements the earlier discussion on internal-force minimisation (He, M. Wu, and Liu 2020) by providing a data-driven means to shape the primitive itself, rather than relying solely on post-hoc optimisation. Moreover, the diffusion-generated primitives can be shared across robots, enabling distributed execution without explicit communication—each robot samples the same stochastic primitive conditioned on its local perception of the object's deformation.

Some approaches are inspired by the formation control that is often used in the area of autonomous ground vehicles (Spletzer, Das, Fierro, Taylor, Kumar, and Ostrowski 2001), and are extended to manipulation, where the agents are treated as part of a cohesive geometric formation (Sieber, Deroo, and Hirche 2013). In this setting, one of the agents can be a human that is guiding the robots, which in turn try to maintain desired geometry w.r.t. each other, while reducing internal stresses (Sieber and Hirche 2019). It can then be shown that, the internal interactions forces in cooperative manipulation that arise from a formation of robots, lie in the null space of the grasp matrix, i.e. the dynamics of the object are not affected by them (De Pascali, Erhart, Zaccarian, Francesco, and Hirche 2022). This insight also bridges the connection to literature on grasping and in-hand manipulation, where the problem setting is similar, in the sense that multiple parallel kinematic chains are tasks to cooperatively manipulate a shared object. Both scenarios have a high kinematic redundancy can be exploited (Yao and Billard 2023). We can therefore also mirror control strategies for robotic grip adjustment, to enable adaptive force modulation for cooperative robotic manipulation in response to environmental feedback. Here, compliant control behaviors are achieved through impedance control schemes on the object level (Pfanne, Chalon, Stulp, Ritter, and Albu-Sch ffer 2020). Dexterity in contact-rich manipulation is often limited, because in contrast to humans, typically grasping models for robotic hands only use single contact points at the fingertips, instead of the whole surface (Bircher, Morgan, and Dollar 2021). Smart gripper design can alleviate this issue to a certain extent, where the geometry is optimized to achieve more dexterous manipulation capabilities, such as fixing the movement of the object's pose along sphere (Patel and Dollar 2021). This highlights the importance of geometric considerations for encoding spatial relationships and constraints and the role of geometric primitives.

Geometric primitives also play an important role in recent advances on general manipulation control, currently mainly in the form of object-centric keypoints. The goal here is to reduce complex tasks across a category of objects, where shape variations are allowed, to geometric constraints on keypoints (W. Gao and Tedrake 2021). When used in closed-loop manipulation control for achieving contact-rich tasks, this keypoint-based representation leads to automatic generalization to a category of objects by incorporating an object-centric action representation (Manuelli, W. Gao, Florence, and Tedrake 2022). The simplicity of the representation via keypoints has also been

proven to have advantages for visual imitation learning. Here, task representations are extracted from demonstrations and decomposed into keypoint-based geometric constraints (J. Gao, Tao, Jaquier, and Asfour 2023). This approach has been extended to bimanual manipulation and was shown to generalize to novel scenes in-category objects, due to the object-centric, embodiment-independent, and viewpoint-invariant representation (J. Gao, Jin, Krebs, Jaquier, and Asfour 2024). Expressing task constraints via geometric constraints on keypoints also facilitates the connection to vision-language models, such that tasks can be defined using language instructions (Huang, C. Wang, Li, Zhang, and Fei-Fei 2025).

## 2 Mathematical Background

### 2.1 Geometric Algebra

In this article, we use the specific variant known as conformal geometric algebra (CGA)  $\mathbb{G}_{4,1}$ . CGA extends the Euclidean space  $\mathbb{R}^3$ , characterized by the three basis vectors  $e_1, e_2, e_3$ , by two additional basis vectors  $e_4$  and  $e_5$ , where  $e_4^2 = 1$  and  $e_5^2 = -1$ . The conformal model is then found by a change of basis that introduces the basis vectors  $e_0 = \frac{1}{2}(e_5 - e_4)$  and  $e_\infty = e_4 + e_5$ , which can be understood as a point at the origin and one at infinity. CGA contains both geometric primitives and conformal transformations in the same algebra. We use these properties in our method to define control schemes based on cooperative geometric primitives and their similarity transformations. In this section, we introduce the necessary mathematical background, and we will use the following notation throughout the paper:  $x$  to denote scalars,  $\mathbf{x}$  for vectors,  $\mathbf{X}$  for matrices,  $X$  for multivectors and  $\mathcal{X}$  for matrices of multivectors.

**2.1.1 Geometric Primitives** Euclidean points  $\mathbf{x}$  are embedded in CGA by using the conformal embedding

$$P(\mathbf{x}) = e_0 + \mathbf{x} + \frac{1}{2}\mathbf{x}^2 e_\infty. \quad (1)$$

These conformal points form the basic building blocks for geometric primitives that can be represented in the algebra. Note that, this nonlinear embedding turns flat Euclidean space into a parabolic space. Furthermore, this embedding is similar to how we traditionally embed vectors in  $\mathbb{R}^3$  into  $\mathbb{R}^4$  when using homogeneous coordinates.

In general, geometric primitives, such as lines, circles and spheres, can be constructed from conformal points using the outer product, i.e.

$$X = \bigwedge_{i=1}^n P_i. \quad (2)$$

In the above equation, depending on the number of points  $n$  and the presence of the point at infinity  $e_\infty$ , different geometric primitives can be constructed. For example, one can construct a line from two points passing through it and the point at infinity. A circle can be constructed from any three points lying on its orbit. Similar constructions can be used for other geometric primitives, such as planes and spheres. The geometric primitives are subspaces of the algebra, defining a nullspace under the outer product

$$\text{NO}_G(\mathbf{X}) = \{\mathbf{x} \in \mathbb{R}^3 : P(\mathbf{x}) \wedge \mathbf{X} = 0\}. \quad (3)$$

This means that the set of conformalized Euclidean points that result in zero using the outer product forms the geometric primitives (Perwass 2009). We have shown previously how this outer product formulation can be formulated within an optimal control framework to define reaching tasks involving the geometric primitives (L w and Calinon 2023).

As part of the geometric algebra, the geometric primitives can be used in algebraic expressions that have geometrically meaningful interpretations. For example, the projection of a point  $P$  to another geometric primitive  $X$  is achieved by the general formula

$$P' = (P \cdot X)X^{-1}. \quad (4)$$

Using what is known as the *meet* operator, it is also possible to calculate intersections between any two geometric primitives

$$Y = (X_1^* \wedge X_2^*)^*, \quad (5)$$

where the  $*$  operator denotes the dual of a multivector, which in CGA amounts to multiplication by the pseudoscalar  $I = e_{0123\infty}$ . Here, it is not required to additionally consider edge cases. For example, in the case of a line and a circle, it is not necessary to check whether the line is tangential, intersecting the circle twice or not at all. Equation (5) will always return a meaningful geometric primitive that conveys the information of these different cases.

**2.1.2 Transformation Groups in CGA** For the purpose of robot kinematics and dynamics, we are interested in a representation of the special Euclidean group  $\mathbf{SE}(3)$ , i.e. the group of rotations and translations. In CGA it can be identified as the *motor group*  $\mathcal{M}$ , which is a representation of  $\text{Spin}(3) \ltimes \mathbb{R}^3$ , a double-cover of  $\mathbf{SE}(3)$ , meaning that  $V_M, -V_M \in \mathcal{M}$  represent the same transformation. The motor group is a six-dimensional manifold that is also a Lie group with the group constraint  $V_M V_M^{-1} = 1$ . The corresponding Lie algebra is the bivector algebra  $\mathbb{B}_\mathcal{M} = \{e_{23}, e_{13}, e_{12}, e_{1\infty}, e_{2\infty}, e_{3\infty}\}$ . Elements of the Lie algebra  $B \in \mathbb{B}_\mathcal{M}$  can be mapped to group elements  $V_M \in \mathcal{M}$  via a surjective map called the exponential map  $\exp : \mathbb{B} \rightarrow \mathcal{M}$ . Accordingly, its inverse operation for projecting group elements to the Lie algebra is named the logarithmic map  $\log : \mathcal{M} \rightarrow \mathbb{B}$ . Motors can be used to transform any multivector within the algebra, i.e. they can be used to transform geometric primitives. This adjoint operation is formulated as the product

$$X' = V_M X \tilde{V}_M, \quad (6)$$

where  $\tilde{V}_M$  denotes the reverse of a multivector.

In addition to rigid body transformations, CGA also contains elements, called dilators, that cause uniform scaling. These dilators form a group under the geometric product that we denote as  $\mathcal{D}$ . If we combine rigid body transformations with uniform scaling using the semi-direct product, we obtain the similarity transformation group  $\mathcal{S}$ , i.e.  $\mathcal{S} = \mathcal{M} \ltimes \mathcal{D}$ . Note that, we omit reflections from this group definition, since we want to preserve the handedness. *Similarity transformations*  $V_S \in \mathcal{S}$  transform geometric primitives using the same Equation (6) as motors. In this article, we use similarity transformations to find



control laws for systems of multiple parallel kinematic chains.

For conciseness, we omit here the full definition of the groups. The equations for exponential and logarithmic maps of the different transformation groups in CGA can be found in Appendix A, along with further explanations of these transformations and their relationship to the matrix Lie algebras.

## 2.2 Robot Modeling using CGA

Using motors, the forward kinematics given the current joint configuration  $\mathbf{q}$  of serial kinematic chains can be found via the product of joint motors, i.e.

$$V_M(\mathbf{q}) = \prod_{j=1}^n V_{M,j}(q_j) = \prod_{j=1}^n \exp(q_j B_j), \quad (7)$$

where  $B_j$  is the bivector describing essentially the screw axis of the  $j$ -th joint. We described the derivation of the analytic  $\mathcal{J}_q^A$  and geometric  $\mathcal{J}_q^G$  Jacobians in (L w and Calinon 2023). Expressed w.r.t. the end-effector motor, their relationship can be found as

$$\mathcal{J}_q^G = -2\widetilde{V_{Mq}}\mathcal{J}_q^A. \quad (8)$$

## 2.3 Geometric Algebra for Optimal Control

In (L w and Calinon 2023), we presented an approach for optimal control using CGA that used the outer product for distance computations between geometric primitives. For a conformal point  $P(\mathbf{q})$ , this distance can be formulated as the outer product of the target geometric primitive and the point

$$d(\mathbf{q}) = \|X \wedge P(\mathbf{q})\|_2^2, \quad (9)$$

where  $P(\mathbf{q}) = V_M(\mathbf{q})e_0\widetilde{V_M}(\mathbf{q})$  is the current end-effector point. By definition, the outer product results in zero if the point is contained within the geometric primitive. Hence, the geometric primitives  $X$  determine a geometric nullspace for the controller. We showed that this nullspace automatically leads to compliant behavior when moving tangentially to the geometric primitive, and stiff behavior when moving orthogonally. In contrast, our control formulation in this article is not based on the outer of geometric primitives. Instead, we formulate the multi-arm manipulation control based on similarity transformations of cooperative geometric primitives. While this formulation differs, the insights about the geometric nullspaces remain valid, as we will explain in more detail in this article.

## 2.4 Geometric Nullspace for Impedance Control

Nullspaces offer an intuitive way to formulate hierarchical control objectives, i.e. to introduce secondary objectives that do not disturb the tracking of the primary one. We have previously exploited the geometric nullspace formed by a line primitive to separate the objectives of tracking a line, while exerting a force along that line in a tactile control application (Bilaloglu, L w, and Calinon 2025). This separation was achieved purely geometrically without the need to derive an appropriate precision matrix and

transforming it to the correct reference frame. Hence, the geometric nullspaces are entirely coordinate-free and thus greatly simplify the definition of hierarchical control objectives.

## 2.5 Cooperative Dual Task Space

In this section, we briefly introduce the cooperative dual-task space using CGA that we presented in (L w and Calinon 2024) as an extension of (Adorno, Fraise, and Druon 2010). The formulation of the CDTs uses two motors to express the collaborative behaviour of the two manipulators. The motors represent a relative and an absolute pose. We first find the relative motor as

$$V_{M_r}(\mathbf{q}_1, \mathbf{q}_2) = \widetilde{V_{M_2}}(\mathbf{q}_2)V_{M_1}(\mathbf{q}_1), \quad (10)$$

and similarly, the absolute motor can be found as

$$V_{M_a}(\mathbf{q}_1, \mathbf{q}_2) = V_{M_2}(\mathbf{q}_2) \exp\left(\frac{1}{2} \log\left(V_{M_r}(\mathbf{q}_1, \mathbf{q}_2)\right)\right), \quad (11)$$

where  $\mathbf{q}_1$  and  $\mathbf{q}_2$  denote the joint configurations of the first and second manipulator, respectively.  $V_{M_1}$  and  $V_{M_2}$  are the corresponding end-effector motors found using the forward kinematics formula from Equation (7). For brevity, we omit the derivation of the corresponding analytic Jacobians  $\mathcal{J}_r^A(\mathbf{q}_1, \mathbf{q}_2)$  and  $\mathcal{J}_a^A(\mathbf{q}_1, \mathbf{q}_2)$  and instead refer readers to (L w and Calinon 2024) for details.

Our extension of the CDTs presented a geometric primitive, called the cooperative pointpair, that corresponds to both end-effector positions simultaneously. This cooperative pointpair is defined as the outer product of the two end-effector points, i.e.

$$P_{cdts} = V_{M_1}(\mathbf{q}_1)e_0\widetilde{V_{M_1}}(\mathbf{q}_1) \wedge V_{M_2}(\mathbf{q}_2)e_0\widetilde{V_{M_2}}(\mathbf{q}_2). \quad (12)$$

This cooperative pointpair is the cornerstone of the idea to formulate collaborative geometric primitives for robotic systems involving more than two kinematic chains.

## 3 Method

Geometric algebra enables the direct representation of geometric primitives within the algebra. Here, we show how to include these geometric primitives in the control objective by deriving control strategies based on similarity transformations.

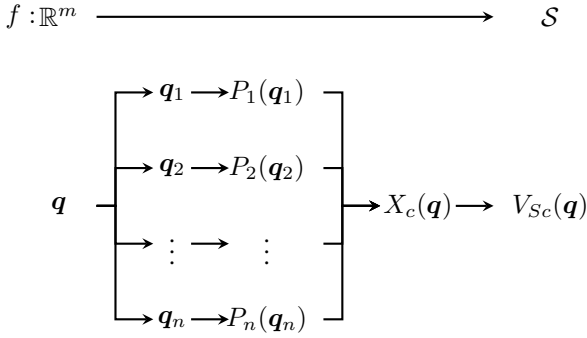
### 3.1 Cooperative Task Space Modeling

Suppose we have a robotic system that consists of  $n$  parallel kinematic chains. Each of these kinematic chains has  $m_i$  degrees of freedom, such that the cooperative system has  $m = m_1 + \dots + m_n$  degrees of freedom. Hence, the joint space configuration of the cooperative system  $\mathbf{q} \in \mathbb{R}^m$  is composed as

$$\mathbf{q} = [\mathbf{q}_1^\top \quad \mathbf{q}_2^\top \quad \dots \quad \mathbf{q}_n^\top]^\top. \quad (13)$$

The approach for modeling this cooperative system can then be described as a function  $f: \mathbb{R}^m \rightarrow \mathcal{S}$  that maps a given joint angle configuration  $\mathbf{q} \in \mathbb{R}^m$  to an element of the group of similarity transformations  $V_{Sc}(\mathbf{q}) \in \mathcal{S}$ . We call  $V_{Sc}(\mathbf{q})$

the cooperative similarity transformation. We schematically depict the function  $f$  in Figure 2. The calculation relies on two intermediary representations. The first one are the points  $P_i(q_i)$  that stand for the end-effector points of the individual kinematic chains. We use these points to calculate a cooperative geometric primitive  $X_c(q)$  that we then use to calculate  $V_{Sc}(q)$ . We will give more details on the involved mathematics in the following sections. Instead, in this section, we focus on the high-level concept given by the modeling of cooperative task spaces via the cooperative similarity transformation.



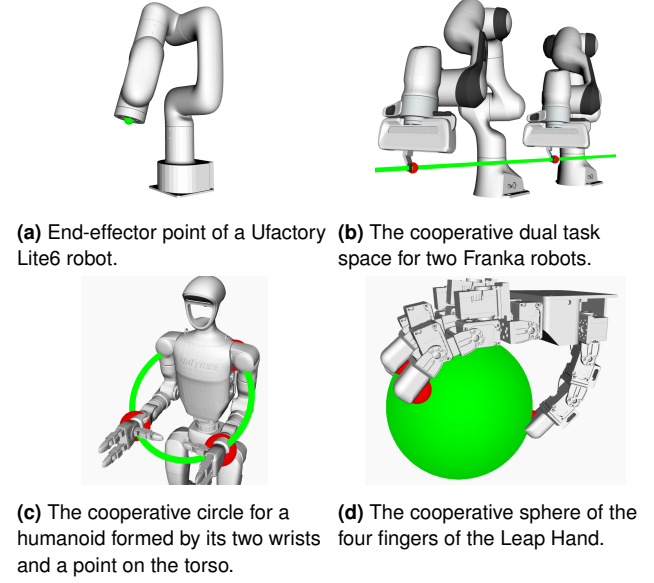
**Figure 2.** Forward kinematics function for cooperative systems using similarity transformations.

Usually, the forward kinematics of a single kinematic chain are described by the function  $g : \mathbb{R}^m \rightarrow SE(3)$ , where  $SE(3)$  is the Lie group of special Euclidean transformations, i.e. rotations and translations. This group is typically represented by homogeneous transformation matrices  $T \in \mathbb{R}^{4 \times 4}$ , making it a matrix Lie group. The matrix obtained by the function  $g$  is usually called the end-effector pose of the robot and is used in various control algorithms, such as impedance/admittance control. We claim that the function  $f$  that we propose in this article extends the function  $g$  in terms of how it is used in control and optimization algorithms to cooperative systems. More concretely, the function  $g$  can be considered as special case of the function  $f$  in the case of  $n = 1$ , since  $SE(3) \subset \mathcal{S}$ . This equivalence of functions  $f$  and  $g$  is ensured by them having the same mathematical properties. Namely,  $f$  is continuous, smooth, non-linear, non-injective and non-surjective, but especially  $f$  is well-defined. This last property means that for every joint angle configuration  $q$  there exists a unique cooperative similarity transformation  $V_{Sc}(q)$ . We highlight this property, because it means we can use the cooperative similarity transformation, and by extension the cooperative geometric primitive, to obtain a new joint configuration (via e.g. inverse kinematics). As the joint configuration  $q$  contains all kinematic chains without assigning them any specific priority, our modeling approach yields control commands for all kinematic chains simultaneously. It also means that the cooperative geometric primitive is not a constraint that is imposed on the system, but on the contrary, is the *state* of the system.

### 3.2 Cooperative Geometric Primitives

We assume that the task-space modeling of the robotic system is done via one of the geometric primitives of CGA. We call this primitive the cooperative geometric primitive

and denote it as  $X_c(q)$ .  $X_c(q)$  extends the ideas around the



**Figure 3.** Cooperative geometric primitives for multiple kinematic chains. The red points indicate the reference points on the robots that are considered for the cooperative geometric primitive, which is then shown in green.

cooperative pointpair that was shown in Equation (12) to  $n$  parallel kinematic chains. We use the properties of the outer product of points to find the cooperative geometric primitive  $X_c(q)$  formed by the end points of those kinematic chains

$$X_c(q) = \bigwedge_{i=1}^n P_i(q) = \bigwedge_{i=1}^n V_{M,i}(q_i) e_0 \tilde{V}_{M,i}(q_i), \quad (14)$$

where the conformal points  $P_i$  correspond to the end points of the kinematic chains. The trivial case of one kinematic chain is a point. Two kinematic chains give a pointpair, three a circle, and four a sphere. Examples of the cooperative geometric primitives for varying numbers of kinematic chains can be seen in Figure 3. Here, we use three manipulators as a proxy for a three-fingered hand to highlight the mathematical equivalence between multiple robot manipulators and robotic hands with multiple fingers. Note that, instead of a pointpair, we could also use a line and instead of a circle, a plane. Both would only require multiplication with the point at infinity and would lead to a relaxed control law that allowed more degrees of freedom. In general, this construction of cooperative geometric primitives holds for geometric algebras of different dimensions, i.e. for any number of  $n$ . In our case, using  $\mathbb{G}_{4,1}$ , which is a 5-dimensional algebra, the highest dimensional geometric primitive is a sphere. A sphere is constructed from  $n = 4$  points, i.e. we limit our view in this section on a maximum of four parallel kinematic chains. In theory, however, using higher dimensional geometric algebras, the derived formulas could be extended to  $4 + n$  parallel kinematic chains. We will go into more detail on this point in the discussion in Section 5.6.

The analytic Jacobian for the cooperative geometric primitive  $\mathcal{J}_c^A(q)$  can be found as the derivative of  $X_c(q)$

w.r.t. the joint angles, i.e.

$$\mathcal{J}_c^A(q) = \frac{\partial}{\partial q} X_c(q) = [\mathcal{J}_{c,1}^A(q) \quad \dots \quad \mathcal{J}_{c,n}^A(q)], \quad (15)$$

where

$$\mathcal{J}_{c,i}^A(q) = P_1 \wedge \mathcal{J}_{P,i} \wedge \dots \wedge P_n, \quad (16)$$

and

$$\mathcal{J}_{P,i} = \mathcal{J}_i^A e_0 \widetilde{M}_i + V_{M,i} e_0 \widetilde{\mathcal{J}}_i^A. \quad (17)$$

The cooperative geometric primitives are an intermediate representation that describe the inherent task space state of the system. In some sense, they represent the cooperative task space geometry for multiple parallel kinematic chains. The actual representation that we derive in this work is based on similarity transformations derived from the geometric primitives. As we show in the following section, these similarity transformations can then be seen as a direct extension of the end-effector pose of a single kinematic chain in classical control and optimization.

### 3.3 Similarity Transformations

Similarity transformations form a Lie group  $\mathcal{S}$  that is a seven dimensional manifold. In the context of this work, they can be used for transforming the cooperative geometric primitives. We define the canonical decomposition of a general similarity transform versor to be

$$V_S = V_T V_R V_D. \quad (18)$$

Elements of the associated Lie algebra  $\mathbb{B}_S$  can be found via the logarithmic map

$$\begin{aligned} B_S &= \log(V_S) \\ &= \log(V_T) + \log(V_R) + \log(V_D). \end{aligned} \quad (19)$$

More details on the transformation groups and their logarithmic maps can be found in Section 2.1.2 and Appendix A. In the same way the Lie algebra of quaternions has been used for interpolation between group elements in applications such as spherical linear interpolation, we can use elements  $B_S \in \mathbb{B}_S$  for interpolating a similarity transformation between two geometric primitives. We show this interpolation in Figure 4.

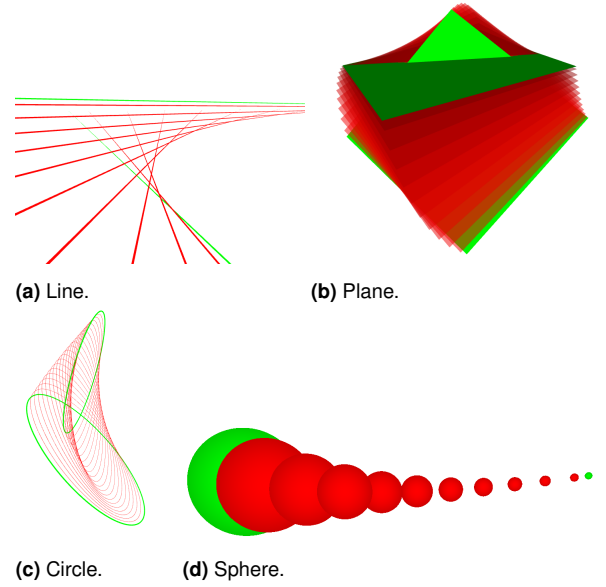
Given the two geometric primitives,  $X_1$  and  $X_2$ , we now derive the similarity transformation  $V_S(X_1, X_2)$  that fulfills

$$X_2 = V_S(X_1, X_2) X_1 \widetilde{V}_S(X_1, X_2). \quad (20)$$

Its bivector  $B_S(X_1, X_2)$  found from the logarithmic map can then be used in the optimization problem given in Equation (39). A general formula to find a versor  $V_C(X_1, X_2)$  that transforms  $X_1$  to  $X_2$  was derived in J. Lasenby, Hadfield, and A. Lasenby 2019

$$V_C(X_1, X_2) = c^{-1} (1 + X_2 X_1), \quad (21)$$

where  $c$  is a normalization constant that depends on the geometric primitives. However, using this general formula,  $V_C(X_1, X_2) \in \mathcal{C}$ , i.e. the resulting versor will be an element of the conformal transformation group  $\mathcal{C}$ , not of the similarity transformation group  $\mathcal{S}$ . In order to restrict the versor to  $\mathcal{S}$ , we present the necessary derivations that



**Figure 4.** Interpolation of different geometric primitives using similarity transformations.

give a similarity transformation between pairs of geometric primitives according to Equation (20). Note that, this clearly shows that the transformation between primitives is not unique, as was already stated in J. Lasenby, Hadfield, and A. Lasenby 2019.

**3.3.1 Point to Point** The simplest case is transforming one point to another. Since points neither have an orientation nor a size, the rotor  $V_R$  and dilator  $V_D$  simplify to the identity transformation, i.e.  $V_R = V_D = 1$ . Regarding the translation versor, given the two points  $P_1$  and  $P_2$ , it is found as

$$\begin{aligned} V_S(P_1, P_2) &= V_T(P_1, P_2) \\ &= \exp((P_2 - P_1) \wedge e_\infty). \end{aligned} \quad (22)$$

**3.3.2 Line to Line** Lines have a flat geometry, i.e. they are part of Euclidean geometry, which means a transformation between two lines can be described purely by rotations and translations. Hence, the versor resulting from Equation (21) will only contain a translation and a rotation, i.e.  $V_S(L_1, L_2) \in \mathcal{M}$ .

**3.3.3 Plane to Plane** Using the same argumentation as for the case of line to line transformations, the transformation from one plane to another can be found using Equation (21). The resulting versor will only contain a translation and a rotation, i.e.  $V_S(E_1, E_2) \in \mathcal{M}$ .

**3.3.4 Circle to Circle** Circles require all three versors in the similarity transformation: translations, rotations and dilations. We can directly compute the dilation versor based on the radii  $r_1$  and  $r_2$  of  $X_1$  and  $X_2$

$$V_D(X_1, X_2) = \exp(r_2 r_1^{-1} e_{0\infty}), \quad (23)$$

where that radius of a circle can be computed as

$$r = \left\| (X \cdot e_\infty) X^{-1} \right\|. \quad (24)$$

Note that the inner expression takes the form of the general subspace projection that we showed in Equation (4). It can

thus be interpreted as projecting infinity to the subspace of the circle and then evaluating the resulting norm.

Next, we compute the rotation versor between the circles by using the normals  $N_1$  and  $N_2$  of the planes that the circles lie. These planes can be found via  $E = C \wedge e_\infty$  and the normal of plane  $E$  is computed as

$$N = E^* - \frac{1}{2}(E^* \cdot e_0)e_\infty. \quad (25)$$

We then insert there normals in Equation (21), which yields

$$\begin{aligned} V_R(X_1, X_2) &= c^{-1}(1 + N_1 N_2) \\ &= c^{-1}\left(1 + (N_2 \cdot N_1) - (N_2 \wedge N_2)\right). \end{aligned} \quad (26)$$

Note that this is the geometric algebra variant of the standard expression for computing a unit quaternion between unit vectors.

Lastly, we compute the translation versor via the center points of the circles. The center point of circles can be found as

$$P(X) = X e_\infty X, \quad (27)$$

which in this case amounts to reflecting infinity in the circle or sphere. Given the center points, the translation versor can then be computed as was indicated for points in Equation (22). Note that, for computing the center of the cooperative circle, we first apply the transformation given by  $V_R(X_1, X_2)V_D(X_1, X_2)$ , because this translation changes the center point of the circle.

**3.3.5 Sphere to Sphere** For spheres, the rotation versor is the identity, because spheres do not have an intrinsic orientation. The dilation and translation versors can then be calculated using the same strategy as for the circles. The computation of the dilation versor follows Equation (23), where the radius of a sphere is also computed in the same way as for the circle, i.e. using Equation (24). We then use Equation (27) to compute their center points and afterwards Equation (22) to obtain the translation versor.

### 3.4 Cooperative Task Space Kinematics

First, we define the cooperative similarity transformation  $V_{Sc}(\mathbf{q})$ . Similarly to how an end-effector pose represents the transformation of the coordinate system at the origin to the end-effector, it is expressed as the similarity transformation w.r.t. a unit geometric primitive at the origin. We denote this primitive as  $X_u$ . In section 3.3, we have already presented how to find similarity transformations between two geometric primitives of the same kind. Accordingly, the versor  $V_{Sc}(\mathbf{q})$  can be found following the explanations around Equation (18) as

$$V_{Sc}(\mathbf{q}) = V_S(X_u, X_c(\mathbf{q})). \quad (28)$$

This versor then fulfills  $X_c(\mathbf{q}) = V_{Sc}(\mathbf{q})X_u\tilde{V}_{Sc}(\mathbf{q})$ . It is a direct equivalent to the end-effector motor found by the forward kinematics in Equation (7), albeit it represents a cooperation between multiple kinematic chains and not just a single one. Hence, the control laws that we propose, based on this cooperative similarity transformation, yield joint space commands for all kinematic chains simultaneously, without adopting principles such as leader/follower. As

mentioned before, the cooperative geometric primitives are an intermediate representation, whereas the cooperative similarity transformations are the actual representation that the rest of our formulations are based on. Therefore, in the following sections, we will only mention the similarity transformations and only assume the cooperative geometric primitives implicitly as the task space geometries. Since these task space geometries are more of an abstract concept rather than a physical representation, we collect the different task space geometries in Table 1. This table also shows the corresponding controllable bivector space, depending on the number of kinematic chains, and formulated w.r.t. the cooperative similarity transformation, not the origin. Note that the corresponding controllable bivector spaces all have a dimension that is strictly smaller than 7, which means that all systems have a geometric nullspace.

Most commonly, optimization solvers and control algorithms make use of one of various Jacobians that can be found for the cooperative similarity versor  $V_{Sc}(\mathbf{q})$  and its bivector  $B_{Sc}(\mathbf{q}) = \log(V_{Sc}(\mathbf{q}))$ . The usual kinematic modeling of robots makes a distinction between the analytic and the geometric Jacobian. Hence, we derive the analytic similarity Jacobian  $\mathcal{J}_S^A(\mathbf{q})$ , and the geometric similarity Jacobian  $\mathcal{J}_S^G(\mathbf{q})$ . We also show the bivector similarity Jacobian  $\mathcal{J}_S^B(\mathbf{q})$  that relates changes in the joint positions  $\mathbf{q}$  to changes in the similarity bivector  $B_{Sc}(\mathbf{q})$ . These Jacobians can then be used to find the gradient, approximate the Hessian, or compute a control signal, depending on the chosen control approach.

As per the usual definition, the analytic similarity Jacobian can be found from the partial derivatives of the cooperative similarity versor w.r.t.  $\mathbf{q}$ , i.e.

$$\begin{aligned} \mathcal{J}_{Sc}^A(\mathbf{q}) &= \frac{\partial}{\partial \mathbf{q}} V_{Sc}(\mathbf{q}) \\ &= \mathcal{J}_{Tc}^A(\mathbf{q})V_{Rc}(\mathbf{q})V_{Dc}(\mathbf{q}) \\ &\quad + V_{Tc}(\mathbf{q})\mathcal{J}_{Rc}^A(\mathbf{q})V_{Dc}(\mathbf{q}) \\ &\quad + V_{Tc}(\mathbf{q})V_{Rc}(\mathbf{q})\mathcal{J}_{Dc}^A(\mathbf{q}), \end{aligned} \quad (29)$$

where  $\mathcal{J}_{Tc}^A(\mathbf{q})$ ,  $\mathcal{J}_{Rc}^A(\mathbf{q})$ , and  $\mathcal{J}_{Dc}^A(\mathbf{q})$  are the Jacobians of the translation, rotation and dilation versors, respectively. We omit the full derivation here for conciseness, which can be found in Appendix B.

The geometric similarity Jacobian can be used to formulate control laws such as differential kinematics or impedance/admittance control. Via the group constraint and from the relationship between the end-effector Jacobians shown in Equation (8), it is easy to see that the following relationship holds:

$$\mathcal{J}_{Sc}^G(\mathbf{q}) = -2\tilde{V}_{Sc}(\mathbf{q})\mathcal{J}_{Sc}^A(\mathbf{q}). \quad (30)$$

Note that, similar to the traditional geometric Jacobian, the geometric similarity Jacobian is also depending on a frame of reference. The above relationship represents the geometric similarity Jacobian w.r.t. the cooperative similarity transformation, as opposed to, for example, the origin. The equivalent end-effector Jacobian from classical single kinematic chain modeling is sometimes referred to as the *body Jacobian*.

Lastly, the bivector similarity Jacobian  $\mathcal{J}_S^B(\mathbf{q})$  is found as the partial derivatives of a cooperative similarity bivector

**Table 1.** Description of the cooperative similarity task spaces for 1, 2, 3, and 4 parallel kinematic chains. In the case of 2 and 3 kinematic chains, two different cooperative geometric primitives are possible by including the basis vector  $e_\infty$  via outer product. This changes the controllable bivector space to be smaller, which essentially means that the system is less constrained and gain additional degrees of freedom.

Joint Space	Cooperative Geometric Primitive		Controllable Bivector Space
$\mathbf{q} = [\mathbf{q}_1^\top]^\top$	Point	$X_c(\mathbf{q}) = P(\mathbf{q})$	$e_{1\infty}, e_{2\infty}, e_{3\infty}$
$\mathbf{q} = [\mathbf{q}_1^\top \quad \mathbf{q}_2^\top]^\top$	Point Pair	$X_c(\mathbf{q}) = P(\mathbf{q}_1) \wedge P(\mathbf{q}_2)$	$e_{12}, e_{23}, e_{0\infty}, e_{1\infty}, e_{2\infty}, e_{3\infty}$
	Line	$X_c(\mathbf{q}) = P(\mathbf{q}_1) \wedge P(\mathbf{q}_2) \wedge e_\infty$	$e_{12}, e_{23}, e_{1\infty}, e_{3\infty}$
$\mathbf{q} = [\mathbf{q}_1^\top \quad \mathbf{q}_2^\top \quad \mathbf{q}_3^\top]^\top$	Circle	$X_c(\mathbf{q}) = P(\mathbf{q}_1) \wedge P(\mathbf{q}_2) \wedge P(\mathbf{q}_3)$	$e_{13}, e_{23}, e_{0\infty}, e_{1\infty}, e_{2\infty}, e_{3\infty}$
	Plane	$X_c(\mathbf{q}) = P(\mathbf{q}_1) \wedge P(\mathbf{q}_2) \wedge P(\mathbf{q}_3) \wedge e_\infty$	$e_{13}, e_{23}, e_{3\infty}$
$\mathbf{q} = [\mathbf{q}_1^\top \quad \mathbf{q}_2^\top \quad \mathbf{q}_3^\top \quad \mathbf{q}_4^\top]^\top$	Sphere	$X_c(\mathbf{q}) = P(\mathbf{q}_1) \wedge P(\mathbf{q}_2) \wedge P(\mathbf{q}_3) \wedge P(\mathbf{q}_4)$	$e_{0\infty}, e_{1\infty}, e_{2\infty}, e_{3\infty}$

$B_{Sc}(\mathbf{q})$  w.r.t.  $\mathbf{q}$  by applying the chain rule

$$\begin{aligned} \mathcal{J}_{Bc}(\mathbf{q}) &= \frac{\partial}{\partial \mathbf{q}} B_{Sc}(\mathbf{q}) = \frac{\partial}{\partial \mathbf{q}} \log(V_{Sc}(\mathbf{q})) \\ &= \frac{\partial}{\partial V} \log(V_{Sc}(\mathbf{q})) \frac{\partial}{\partial \mathbf{q}} V_{Sc}(\mathbf{q}) \\ &= \mathcal{J}_{S \rightarrow \mathbb{B}_S}(\mathbf{q}) \mathcal{J}_{Sc}^A(\mathbf{q}), \end{aligned} \quad (31)$$

where  $\mathcal{J}_{S \rightarrow \mathbb{B}_S}(\mathbf{q})$  is the Jacobian of the logarithmic map of the similarity transformation group. The bivector similarity Jacobian can then be used in applications such as optimization-based inverse kinematics.

### 3.5 Cooperative Task Space Dynamics

In this section, we present the derivation of the task space dynamics defined by the cooperative similarity transformations. We start from the well-known joint space dynamics

$$\mathbf{M}(\mathbf{q})\ddot{\mathbf{q}} + \mathbf{C}(\mathbf{q}, \dot{\mathbf{q}})\dot{\mathbf{q}} + \mathbf{g}(\mathbf{q}) = \boldsymbol{\tau} - \boldsymbol{\tau}_{\text{ext}}, \quad (32)$$

where  $\mathbf{M}(\mathbf{q})$  is known as the inertia or generalized mass matrix,  $\mathbf{C}(\mathbf{q}, \dot{\mathbf{q}})$  is representing Coriolis/centrifugal forces,  $\mathbf{g}(\mathbf{q})$  stands for the gravitational forces,  $\boldsymbol{\tau}$  is the vector of joint torques and  $\boldsymbol{\tau}_{\text{ext}}$  are the external torques. Analogously to the classical formulation of the task space dynamics using the geometric Jacobian  $\mathcal{J}^G(\mathbf{q})$  of a single kinematic chain, we can use the geometric similarity Jacobian  $\mathcal{J}_{Sc}^G(\mathbf{q})$  to derive the dynamics of the cooperative task space. Recall that the geometric similarity Jacobian defines the relationship between the joint and task space velocities, i.e.

$$\mathcal{V}_S = \mathcal{J}_{Sc}^G(\mathbf{q})\dot{\mathbf{q}}, \quad (33)$$

where  $\mathcal{V}_S \in \mathbb{B}_S$  is a twist-like quantity. Taking the derivative w.r.t. time yields

$$\dot{\mathcal{V}}_S = \mathcal{J}_{Sc}^G(\mathbf{q})\ddot{\mathbf{q}} + \dot{\mathcal{J}}_{Sc}^G(\mathbf{q}, \dot{\mathbf{q}})\dot{\mathbf{q}}. \quad (34)$$

Using this result and following the derivations from the classical operational space formulation, we find the similarity task space dynamics as

$$\mathbf{M}_{Sc}(\mathbf{q})\dot{\mathcal{V}}_S + \mathbf{C}_{Sc}(\mathbf{q}, \dot{\mathcal{V}}_S)\dot{\mathcal{V}}_S + \mathbf{g}_{Sc}(\mathbf{q}) = \mathcal{W}_S, \quad (35)$$

where the similarity inertia matrix is found as

$$\mathbf{M}_{Sc}(\mathbf{q}) = \left( (\mathcal{J}_{Sc}^G) \mathbf{M}^{-1} (\mathcal{J}_{Sc}^G)^\top \right)^{-1}, \quad (36)$$

the Coriolis/centrifugal forces as

$$\mathbf{C}_{Sc}(\mathbf{q}, \dot{\mathbf{q}}) = \mathbf{M}_{Sc} \left( \mathcal{J}_{Sc}^G \mathbf{M}^{-1} \mathbf{C} - \dot{\mathcal{J}}_{Sc}^G \dot{\mathbf{q}} \right), \quad (37)$$

and the gravitational forces as

$$\mathbf{g}_{Sc}(\mathbf{q}) = \mathbf{M}_{Sc} \mathcal{J}_{Sc}^G \mathbf{M}^{-1} \mathbf{g}. \quad (38)$$

### 3.6 Cooperative Similarity Control

We base the formulation of the control objective for cooperative manipulation control on similarity transformations. We express the control objective as an optimization problem

$$\min \left\| \log \left( \tilde{V}_{Sc}(\mathbf{q}) V_{Sd} \right) \right\|, \quad (39)$$

where we use the logarithmic map of the versor representation, as explained in Section 2.1.2. The similarity transformation  $V_{Sd}$  is a desired similarity transformation that can be found using the unit geometric primitive

$$V_{Sd} = V_S(X_u, X_d), \quad (40)$$

where  $X_d$  is a desired geometric primitive. Note that the desired similarity transformation can also be set directly without the desired geometric primitive, but latter facilitates the process, since it can be visualized. For any pair of geometric primitives  $X_c(\mathbf{q})$  and  $X_d$ , there is at least one similarity transformation  $V_S$  that transforms  $X_c(\mathbf{q})$  into  $X_d$ . Solving the optimization problem in Equation 39 essentially means making  $V_S$  the identity transformation. Here, we assume that  $X_c(\mathbf{q})$  and  $X_d$  are of the same type. In general, this does not need to be the case, but is outside the scope of this work. We add more insights on mixing geometric primitives in the discussion section.

The versor from Equation (39) can be written as

$$V_S(X_c(\mathbf{q}), X_d) = V_{Scd}(\mathbf{q}) = \tilde{V}_{Sc}(\mathbf{q}) V_{Sd}, \quad (41)$$

which is equivalent to the formulation of inverse kinematics for a single kinematic chain based on the end-effector motor. Note that, instead of using Equation (41), the versor  $V_S(X_c(\mathbf{q}), X_d)$  can also directly be found following the explanations in Section 3.3. From Equation (41), we can then find the bivector  $B_{Scd}(\mathbf{q})$  via the logarithmic map that represents the task-space control signal, similar to a twist,

$$B_{Scd}(\mathbf{q}) = \log(V_{Scd}(\mathbf{q})). \quad (42)$$

Note that, in general, this bivector can be seven-dimensional, since the similarity transformation group is a seven-dimensional manifold.

Here, we do not propose any particular solver for this optimization problem, but focus on the general modeling. In practice, this optimization problem can be solved using state-of-the-art methods from the literature around optimal control, impedance/admittance control or simply using differential kinematics. To account for this generality of the problem formulation, we give all the mathematical details that are necessary for these different control paradigms. In the following, we explain how this control problem can be embedded into existing control and optimization frameworks, in particular impedance control and optimal control.

### 3.7 Inverse Kinematics

We formulate the optimization problem from Equation (39) such that it becomes an inverse kinematics problem

$$\mathbf{q}^* = \arg \min_{\mathbf{q}} \left\| \mathbf{e}(\mathbf{q}) \right\|_2^2, \quad (43)$$

where the error vector  $\mathbf{e}(\mathbf{q})$  is

$$\mathbf{e}(\mathbf{q}) = \log \left( \tilde{V}_{Sc}(\mathbf{q}) V_{Sd} \right). \quad (44)$$

To illustrate how this inverse kinematics problem can be solved using optimization-based approaches, we present a simple Gauss-Newton step

$$\mathbf{q}_{k+1} = \mathbf{q}_k - \alpha \mathbf{H}(\mathbf{q}_k) \mathbf{e}(\mathbf{q}_k), \quad (45)$$

that starts from an initial guess  $\mathbf{q}_0$  and iterates until convergence. The step size  $\alpha$  and can be found using line search. The matrix  $\mathbf{H}(\mathbf{q})$  is the Hessian matrix of Equation (43). It can be approximated as

$$\mathbf{H}(\mathbf{q}) = \left( \mathcal{J}_{Be}^\top(\mathbf{q}_k) \mathcal{J}_{Be}(\mathbf{q}_k) \right)^{-1} \mathcal{J}_{Be}^\top(\mathbf{q}_k), \quad (46)$$

where the Jacobian  $\mathcal{J}_{Be}(\mathbf{q}_k)$  is found as

$$\mathcal{J}_{Be}(\mathbf{q}_k) = \mathcal{J}_{S \rightarrow \mathbb{B}_S}(\mathbf{q}) \tilde{\mathcal{J}}_{Sc}^A(\mathbf{q}) V_{Sd}, \quad (47)$$

and follows the same derivation as the cooperative bivector Jacobian in Equation (31).

Instead of using the optimization-based inverse kinematics, we can also use a task space velocity based on the differential kinematics. For this we recall Equation (33) that relates the joint velocity  $\dot{\mathbf{q}}$  to the similarity twist  $\mathcal{V}_S$  via the geometric similarity Jacobian. Hence, in order to obtain a joint-space command by using differential kinematics, we invert the geometric similarity Jacobian

$$\dot{\mathbf{q}} = \left( \mathcal{J}_{Sc}^G(\mathbf{q}) \right)^{-1} B_{S,i}. \quad (48)$$

The control scheme is formulated w.r.t. to the cooperative similarity transformation, i.e. the local frame. Note that this control formulation is identical to the usual way of doing differential kinematics. The only difference is that a cooperative task space of multiple kinematic chains is modeled, as opposed to a single one. Therefore, given a similarity twist, this control law will find joint velocities for all kinematic chains that are part of the cooperative system.

### 3.8 Impedance Control

The traditional control methods require a twist as a task-space control command. Here, the cooperative similarity bivector  $B_{Scd}(\mathbf{q})$  from Equation (42) can be used to define a corresponding similarity twist  $\mathcal{V}_S$ .

We derive an impedance control law to regulate the system to a desired similarity transformation  $V_{Sd}$ . Assuming  $\dot{V}_{Sd} = 0$ , we find the error dynamics as

$$\begin{aligned} B_{S,e} &= \log \left( \tilde{V}_{Sc}(\mathbf{q}) V_{Sd} \right), \\ \dot{B}_{S,e} &= -\frac{1}{2} \mathcal{V}_{Sc}, \end{aligned} \quad (49)$$

where  $\mathcal{V}_{Sc}$  is the similarity twist from Equation (33). These error dynamics can then be used to define a simple control law for the desired similarity task space acceleration  $\dot{\mathcal{V}}_{Sd}(\mathbf{q})$

$$\dot{\mathcal{V}}_{Sd}(\mathbf{q}) = \mathbf{K} B_{S,e} + \mathbf{D} \dot{B}_{S,e}, \quad (50)$$

where the matrices  $\mathbf{K}, \mathbf{D} \in \mathbb{R}^{7 \times 7}$  are the stiffness and damping gains, respectively. Using the similarity task space dynamics from Equation (41), we then compute the desired similarity wrench  $\mathcal{W}_{Sd}(\mathbf{q})$

$$\mathcal{W}_{Sd}(\mathbf{q}) = \mathbf{M}_{Sc}(\mathbf{q}) \dot{\mathcal{V}}_{Sd}(\mathbf{q}) + \mathbf{C}_{Sc}(\mathbf{q}, \dot{\mathbf{q}}) + \mathbf{g}_{Sc}(\mathbf{q}), \quad (51)$$

which is then used to compute a joint torque control command  $\boldsymbol{\tau}(\mathbf{q})$

$$\boldsymbol{\tau}(\mathbf{q}) = \left( \mathcal{J}_{Sc}^G(\mathbf{q}) \right)^\top \mathcal{W}_{Sd}(\mathbf{q}). \quad (52)$$

The derivation of closely related control schemes, such as admittance control, is very similar and could be used to find desired joint positions or velocities for robotic systems lacking torque control.

### 3.9 Optimal Control

A general optimal control problem can be written as

$$\begin{aligned} \{\mathbf{u}_k^*\}_{k=0}^{n-1} &= \arg \min_{\{\mathbf{u}_k\}_{k=0}^{n-1}} \sum_{k=0}^{n-1} l(\mathbf{x}_k, \mathbf{u}_k) + l_f(\mathbf{x}_n), \\ \text{s.t. } \mathbf{x}_{k+1} &= f(\mathbf{x}_k, \mathbf{u}_k), \\ \mathbf{x}_k &\in \mathcal{X}, \\ \mathbf{u}_k &\in \mathcal{U}, \end{aligned} \quad (53)$$

where  $\mathbf{x}_k$  are the states,  $\mathbf{u}_k$  the control commands,  $l(\mathbf{x}_k, \mathbf{u}_k)$  the running costs,  $l_f(\mathbf{x}_n)$  the terminal cost,  $f(\mathbf{x}_k, \mathbf{u}_k)$  the system dynamics, and  $\mathcal{X}$  and  $\mathcal{U}$  are the admissible sets for the state and control, respectively.

For simplicity, we assume linear system dynamics

$$\mathbf{x}_{k+1} = f(\mathbf{x}_k, \mathbf{u}_k) = \mathbf{A} \mathbf{x}_k + \mathbf{B} \mathbf{u}_k, \quad (54)$$

and choose second-order dynamics in the joint space, such that our state and control become

$$\mathbf{x} = \begin{bmatrix} \mathbf{q} \\ \dot{\mathbf{q}} \end{bmatrix}, \quad \mathbf{u} = \ddot{\mathbf{q}}. \quad (55)$$

Since the main purpose of this section is to demonstrate the usage of the cooperative similarity transforms in optimal



control conceptually, we limit the scope to a simple reaching problem. Hence, the running cost is independent of the state  $\mathbf{x}_k$  and only penalizes the magnitude of the control commands  $\mathbf{u}_k$ , i.e.

$$l(\mathbf{x}_k, \mathbf{u}_k) = \mathbf{u}_k^\top \mathbf{R} \mathbf{u}_k, \quad (56)$$

where  $\mathbf{R}$  is a regularization term. The terminal cost then adopts Equation (39) to minimize the distance to the desired similarity transformation  $V_{Sd}$

$$l_f(\mathbf{x}_n) = l_f(\mathbf{q}_n, \dot{\mathbf{q}}_n) = \left\| \log(\tilde{V}_{Sc}(\mathbf{q}_n) V_{Sd}) \right\|_Q^2, \quad (57)$$

where the  $Q$  is the precision matrix.

While in this formulation the system dynamics are linear, the terminal cost term contains non-linearities. Hence, this problem cannot be solved in closed-form and instead needs to be solved iteratively. Once the optimal trajectory is obtained, the resulting sequence of control commands can be used either open-loop, or in a receding horizon fashion as in model predictive control. Since we opted for second-order dynamics in the joint space, the control commands are accelerations. These can either be integrated to obtain position or velocity commands, or transformed to torque commands via inverse dynamics.

### 3.10 Geometric Nullspace

One of the features of a redundant systems is the existence of a nullspace. This comes from the fact that the linear map, given by the Jacobian of the system, does not have full column rank. The nullspace is the set of vectors that are mapped to zero, i.e. the set of joint velocities that result in no end-effector movement. This is a desirable property for redundancy resolution and hierarchical control with secondary objectives. The classic approach for projecting any velocity vector to the nullspace is using a nullspace projection operator like

$$\mathbf{N}(\mathbf{q}) = \mathbf{I} - \left( \mathcal{J}_S^G(\mathbf{q}) \right)^{-1} \mathcal{J}_S^G(\mathbf{q}), \quad (58)$$

where in our case  $\mathcal{J}_S^G(\mathbf{q})$  is the geometric similarity Jacobian from Equation (30).

In our work, we use the cooperative similarity transformations to extend the notion of end-effector poses to systems of multiple kinematic chains. The geometric nullspace is induced by the cooperative geometric primitive, since by definition of the geometric primitives, they form a nullspace under the outer product as seen from Equation (3). In our formulation of a cooperative geometric primitive, we use the end-effector points of the involved kinematic chains as its defining points. Consequently, those end-effector points can move arbitrarily along directions tangent to the cooperative primitive without altering the primitive itself. Hence, the cooperative geometric primitive directly encodes the system's geometric nullspace. Due to geometric algebra, the primitives are algebraic objects and not simply parameterizations. But the explicit parameterization, such as radius and center of a sphere, can easily be obtained from these algebraic objects. Furthermore, we can easily relate the derivatives of those parameters to the joint angles by using the analytic Jacobian of the cooperative geometric primitive

$\mathcal{J}_c^A(\mathbf{q})$  that we showed in Equation (15). This enables using recent redundancy resolution methods such as Haug 2023 or Ferrentino, Savino, Franchi, and Chiacchio 2024.

### 3.11 Manipulability Analysis

For a more fundamental understanding of the proposed cooperative similarity control of these complex robotic systems, we define the manipulability of the system. The cooperative similarity transformation fulfills the same role as the rigid body transformation at the end-effector, represented as a motor, for robotic arms. For traditional manipulability analysis, the geometric Jacobian related to this end-effector motor is used. Accordingly, we define the similarity manipulability ellipsoid as

$$\mathcal{M}_S(\mathbf{q}) = \mathcal{J}_S^G(\mathbf{q}) (\mathcal{J}_S^G(\mathbf{q}))^\top, \quad (59)$$

which further highlights the parallels of the system modeling between single kinematic chains using motors and multiple kinematic chains using the cooperative similarity transformation.

The similarity manipulability ellipsoid  $\mathcal{M}_S(\mathbf{q}) \in \mathbb{R}^{7 \times 7}$  can be understood as a direct equivalent to the traditional velocity manipulability ellipsoid. Hence, it informs about the system's capability to move the cooperative geometric primitive, which is the same as the traditional manipulability's information about the robot's ability to move its end-effector. In addition, the cooperative similarity manipulability also contains a dimension that expresses the ability to dilate the cooperative geometric primitive. This dimension becomes of particular interest, when we look at the inverse manipulability  $\mathcal{M}_S^{-1}(\mathbf{q})$ . This inverse represents the force manipulability and thus expresses the system's ability to enact forces in different directions. For the dilation, this expresses the ability to apply force that increase or decrease the size of the cooperative geometric primitive. For example, in the case of three manipulators, this would give a measure of the radial force for changing the radius of the cooperative circle. This is important information for tasks such as carrying big and bulky objects, where the three manipulators are required to apply force to an object while carrying it. The cooperative force manipulability can then be used to optimize for the ideal configuration to enable the maximum force on the dilation. Similarly, this manipulability ellipsoid could then also be used to implement standard techniques for avoiding the singularities discussed in Section 5.3.

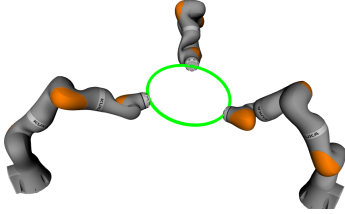
## 4 Results

We have presented a purely geometric framework for the cooperative control of multiple kinematic chains. Accordingly, the examples are designed to illustrate the underlying mathematical concepts developed in this work through kinematic simulations. We show the cooperative geometric primitives both in optimal control and in teleoperation scenarios. To maintain generality, the examples are intentionally presented in an abstract form, without specific applications. Instead, we only refer to potential applications. The presented results are implemented using

our open-source software framework *gafro*<sup>\*</sup>. Additional material and videos of the teleoperation examples can be found on our website<sup>†</sup>.

#### 4.1 Impedance Control

In this section, we demonstrate the operational space control for the similarity task space that we derived in Section 2.4. As the setup, we choose three Kuka IIWA14 robots, as shown in Figure 5, giving the full system 21 degrees of freedom. By considering the three end-effector points, the cooperative geometric primitive takes the form of a circle. The control law follows the impedance control in the similarity task space according to Equations (50), (51), and (52). The final control command will therefore be a torque command. To account for this, we are using a simulation environment that is capable of dynamics simulation using the forward dynamics. For the stiffness and damping parameters we set the gain matrices as  $\mathbf{K} = \text{diag}([1.0 \ 1.0 \ 1.0 \ 7.5 \ 7.5 \ 7.5 \ 7.5])$  and  $\mathbf{D} = \text{diag}([5.0 \ 5.0 \ 5.0 \ 5.0 \ 5.0 \ 5.0 \ 5.0])$ , respectively.

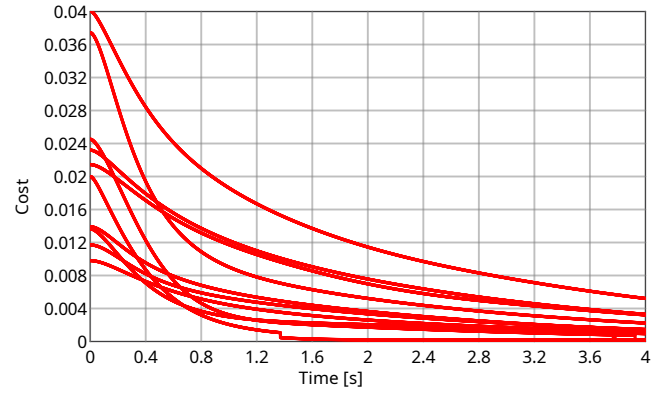


**Figure 5.** Three Kuka IIWA14 robots arranged in a circle. The green circle shows the cooperative geometric primitive for the current joint configuration.

For evaluating the similarity task space impedance controller, we uniformly sample ten different random initial joint configurations around a nominal initial joint configuration  $\mathbf{q}_0 = [\mathbf{q}_{1,0}^\top, \mathbf{q}_{2,0}^\top, \mathbf{q}_{3,0}^\top]^\top$ , with  $\mathbf{q}_{1,0} = \mathbf{q}_{2,0} = \mathbf{q}_{3,0} = [0 \ -0.7854 \ 0 \ 1.3962 \ 0 \ 0.6109 \ 0]^\top$ . We then simulate the system behavior for the given controller and task parameters during 4s. We show the corresponding cost values over time in Figure 6 (i.e., the norm of the bivector from Equation (49)). It can clearly be seen that the similarity impedance controller minimizes the error and regulates the system towards the target cooperative similarity transformation. Note that the response of the system could be further improved by tuning the stiffness and damping parameters.

#### 4.2 Optimal Control Illustrations

In this section, we show several simulated examples of the optimal control problem formulated in Equation (53). As previously described, the problem is formulated as a reaching problem. Accordingly, these results extend reaching a desired pose with a single manipulator to a cooperative system. We use the terminal cost given by Equation (57) and choose a desired similarity transformation for the cooperative geometric primitive that corresponds to the number of kinematic chains, which can be seen from Table 1. We choose to solve the optimization problem with the iterative linear quadratic regulator (iLQR) algorithm. We set



**Figure 6.** Impedance control response.

the number of timesteps to  $n = 250$ , and use  $\Delta t = 0.001$ . Hence, the planning horizon corresponds to 2.5s. Since the problem setup is very simple, typically the solver converges in five or less iterations. Convergence means that the error bivector from the terminal cost in Equation (57) essentially becomes zero, which means that the final state has reached the desired similarity transformation.

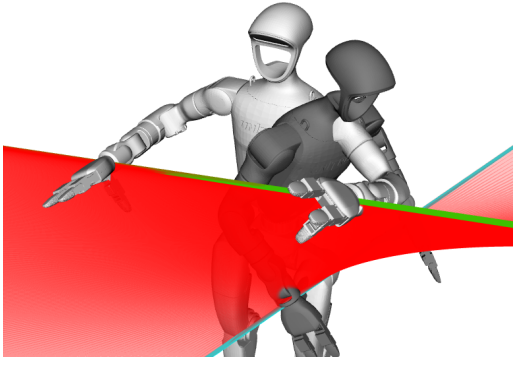
**4.2.1 Line Reaching** In our first example, we examine the simplest case: two points representing the cooperative task space of two independent kinematic chains, such as a bi-manual platform or the arms of a humanoid. In this scenario, we use the Unitree G1 robot model. We neglect the legs in this example and instead focus on the two arms, starting from the waist joints. Since the arms each have seven degrees of freedom and the waist has three, the total system has 17 degrees of freedom. The reference points to construct the line according to Equation (14) are located at the wrists. By using a line instead of a point pair to model the cooperative behavior of the two arms, we remove constraints on the motion and allow for the arms to move freely along the line, which essentially defines a geometric nullspace. Here, we illustrate a reaching motion from one line primitive to another in Figure 7a with a humanoid model. In Figure 1, we have shown the example of cloth folding as a potential application for a dual arm system modeled by a cooperative line. In Figure 7b we show the corresponding task space bivector command that was found by solving the optimal control problem.

**4.2.2 Circle Reaching** In the second example, we consider three points that define the cooperative task space of three independent kinematic chains. In CGA, three non-collinear points uniquely define a circle primitive. Like a line, a circle introduces a null space, allowing the three points to move freely along it. We demonstrate transitions from one circle to another using three different setups: a trio of manipulators, the torso and arms of a humanoid robot, and a collaborative reaching scenario involving both a manipulator and a humanoid.

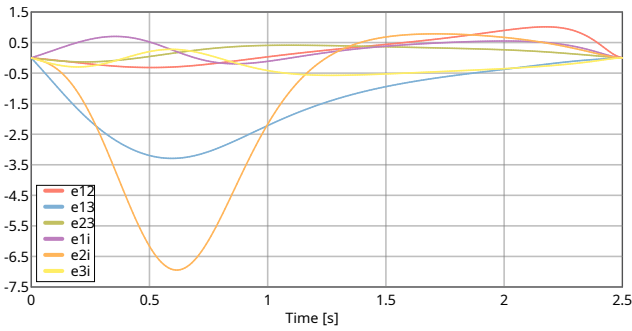
The first scenario of three manipulators, which is shown in Figure 8a, is the most generic one. It can be seen as a general

<sup>\*</sup><https://gitlab.com/gafro>

<sup>†</sup>[https://geometric-algebra.tobiloew.ch/cooperative\\_geometric\\_primitives/](https://geometric-algebra.tobiloew.ch/cooperative_geometric_primitives/)



(a) Task space trajectory of the cooperative line primitive that corresponds to the optimal joint trajectory that was found for reaching the target similarity transformation. The points used for modeling the cooperative line are located at the wrists. The initial configuration is shown in white and the final one in gray. The green line is the initial cooperative line, the turquoise one the final, and in red we show its trajectory in between.



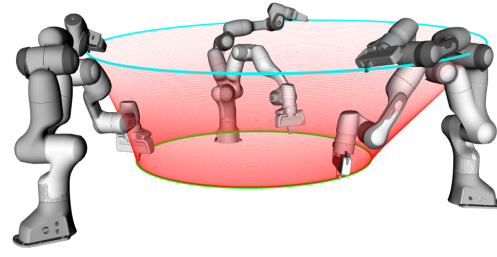
(b) Trajectory of the task space bivector command.

**Figure 7.** Humanoid reaching for a desired cooperative line. Including the joints at the waist, this system has 17 degrees of freedom. 7a shows the task space trajectory and 7b the corresponding similarity bivector command.

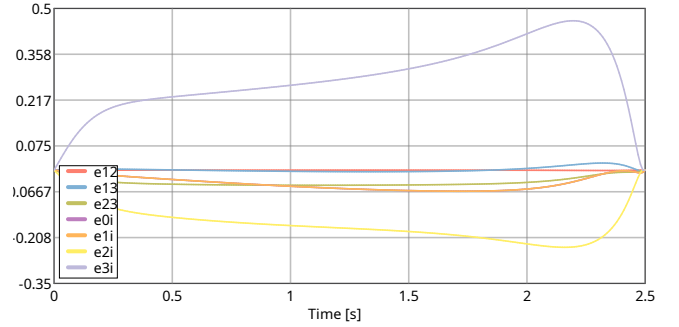
proxy for tasks involving the lifting and manipulation of big and bulky objects. It is also the same scenario that we used as an example for the impedance controller in Section 4.1. We use three Franka robots in this example, which means that the system has 21 degrees of freedom. Notice how the turquoise target circle is larger than the green initial circle. Consequently, the dilation bivector command  $e_{0\infty}$  needs to be non-zero in order for the cooperative system to reach the target. This can be seen in the corresponding Figure 8b, where we plot the bivector command over the planning horizon.

We show the second scenario in Figure 9a with the bivector command trajectory in Figure 9b. Here, we use a single humanoid and choose a point on the torso and its two arms as the three points that define the circle. Since we use the same robot model, i.e. the Unitree G1, as in Section 4.2.1, the system also has 17 degrees of freedom. This example illustrates how a cooperative circle can be used to model a single humanoid carrying a big and bulky object in arms, while pressing the object against its torso to provide additional support. We show an illustration of this task for a human in Figure 1.

The last scenario for the cooperative circle features a manipulator and a humanoid that are collaborating for the reaching of the desired circle, as shown in Figure 10a.

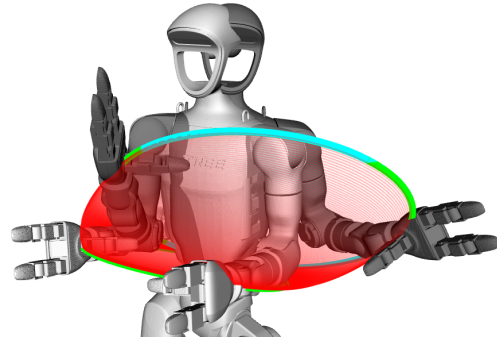


(a) The initial configuration is shown in white and the final one in gray. The green circle is the initial cooperative circle, the turquoise one the final, and in red we show its trajectory in between.

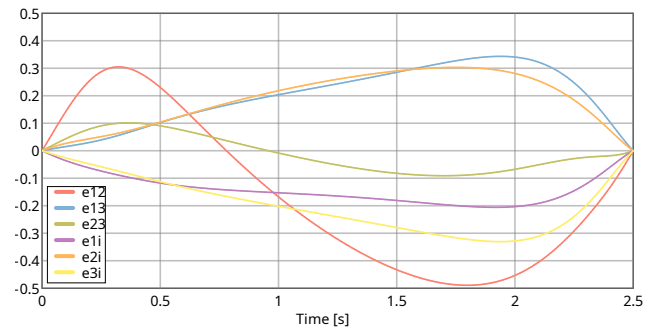


(b) Trajectory of the task space bivector command.

**Figure 8.** Three Franka robots reaching for a cooperative circle. Since each robot has seven degrees of freedom, the complete system has 21 degrees of freedom.



(a) The initial configuration is shown in white and the final one in gray. The green circle is the initial cooperative circle, the turquoise one the final, and in red we show its trajectory in between.

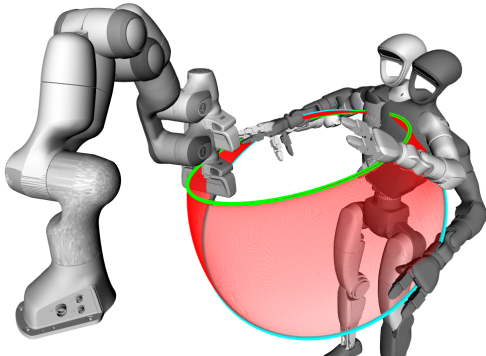


(b) Trajectory of the task space bivector command.

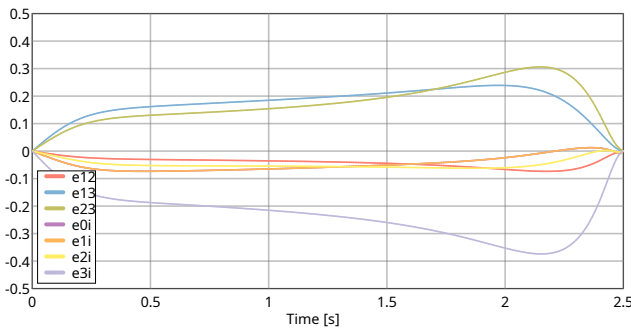
**Figure 9.** Humanoid reaching for a cooperative circle by using a point on its torso.

Although we use a humanoid in this example, it could be

easily replaced by a human in order to model a human-robot collaboration scenario. Since this extension provides an interesting opportunity for future work, we discuss it further in Section 5.5. The bivector command trajectory can be seen in Figure 10b.



(a) The initial configuration is shown in white and the final one in gray. The green circle is the initial cooperative circle, the turquoise one the final, and in red we show its trajectory in between.

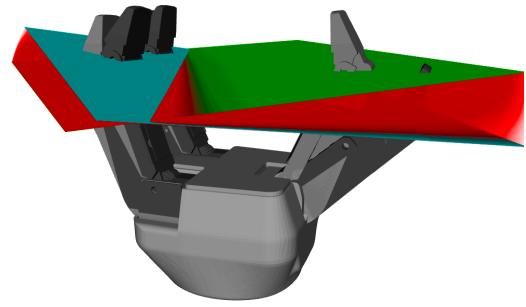


(b) Trajectory of the task space bivector command.

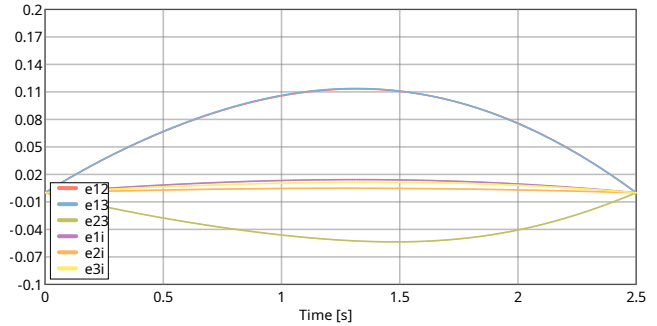
**Figure 10.** Collaboration between a manipulator and a humanoid modeled by a cooperative circle.

**4.2.3 Plane Reaching** Similar to the line, an infinite plane in CGA can be constructed by combining three points with a point at infinity. As with other primitives in CGA, a plane represents a geometric null space. Therefore, moving the three points within the same plane results in an equivalent control objective. We demonstrate this behavior using a three-fingered robotic hand, as shown in Figure 11a. In Figure 1, we have used the example of a human carrying a plate as a potential application for this scenario. Since a plane is a flat geometric primitive, the dilation component is zero, as can be seen from the bivector command trajectory in Figure 11b.

**4.2.4 Sphere Reaching** As the last example, we considered the sphere primitive constructed using four points, which is the limit that we can reach with CGA. We demonstrate reaching from one sphere to another using a four-fingered hand in Figure 12a, and a collaborative scenario involving two humanoids in Figure 13a. The corresponding bivector command trajectories can be found in Figures 12b and 13b, respectively. The cooperative sphere of the four-fingered hand is a general example for grasping using a robotic hand and can thus be applied to a wide range of applications. We also show an example of this in Figure 1. Similarly, in the scenario with the humanoids, one of the



(a) The initial configuration is shown in white and the final one in gray. The green plane is the initial cooperative plane, the turquoise one the final, and in red we show its trajectory in between.



(b) Trajectory of the task space bivector command.

**Figure 11.** Three-fingered hand reaching for a cooperative plane.

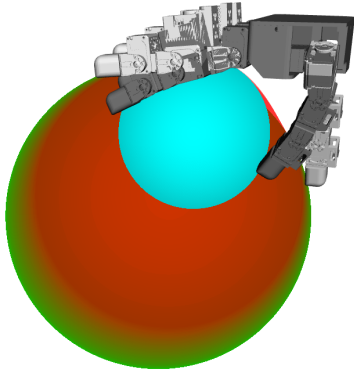
humanoids could be replaced by the user in order to model human-robot collaboration. Generally, this example illustrates the collective transport of big and bulky objects using two independent humanoids, as depicted in Figure 1.

**4.2.5 Line Constraint** The cooperative geometric primitives can not only serve as task objectives but also as constraints for secondary tasks. In the previous examples, we demonstrated reaching problems in which the cooperative primitive transitions from one configuration to another. Here, we illustrate how the cooperative line primitive can be used to impose geometric constraints on the motion of a dual-arm system, while simultaneously executing individual end-effector tasks.

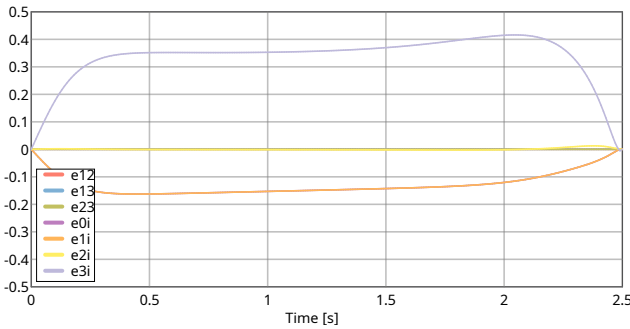
We use the Unitree G1 humanoid robot with its two arms. Together with the waist joints, it provides 17 degrees of freedom. The cooperative line primitive is formed by the two wrist points. In the first scenario, shown in Figure 14a, we task the right hand to reach a target point, while the left hand remains stationary. This represents an asymmetric reaching task in which only one arm is actively controlled, while the other provides a fixed reference. The cooperative line connecting the two wrists therefore changes freely as the right hand moves towards its target, which can be observed from the trajectory of the line parameters in Figure 14b.

In the second scenario, we introduce an additional constraint on the cooperative line, which must now remain horizontal throughout the motion. As shown in Figure 15a, the right hand again reaches a target point, but now the left hand must move in coordination to maintain the horizontal orientation of the line primitive. This coordinated motion ensures that the line parameter corresponding to the





(a) The initial configuration is shown in white and the final one in gray. The green sphere is the initial cooperative sphere, the turquoise one the final, and in red we show its trajectory in between.



(b) Trajectory of the task space bivector command.

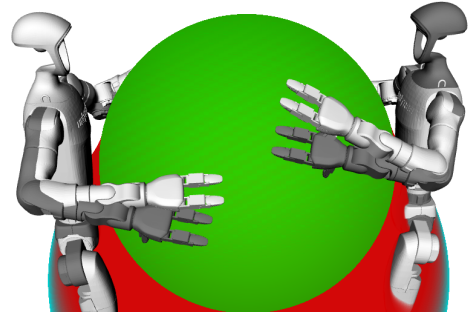
**Figure 12.** Four-fingered hand reaching for a cooperative sphere. The hand has 16 degrees of freedom.

vertical component remains constant, which is reflected in Figure 15b.

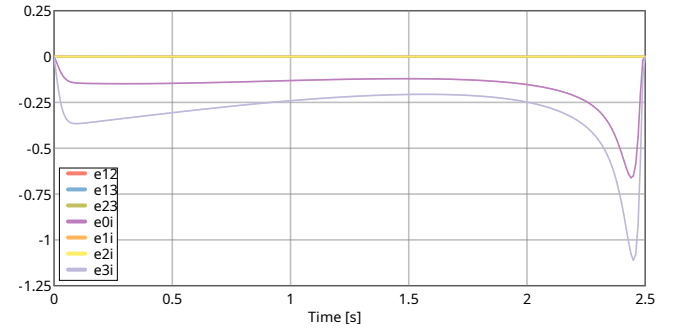
This example demonstrates the versatility of the cooperative geometric primitive framework for encoding both task objectives and geometric constraints. By specifying constraints on the cooperative primitive, such as maintaining a horizontal orientation, the framework automatically generates coordinated motion across multiple kinematic chains. This capability is particularly useful for tasks like carrying objects with specific orientation requirements, or maintaining formation constraints during collaborative manipulation.

### 4.3 Geometric Nullspace

In this section, we provide a simple example of how the geometric nullspaces can be used in practice to define secondary control objectives. We arrange three Franka robots in circle, yielding a system with 21 degrees of freedom and cooperative geometric primitive that is a circle. The primary objective in this setup is leaving the circle unchanged, i.e. remaining at the same cooperative similarity transformation. The geometric nullspace allows each of robots to move freely on the circle. Hence, we define a secondary objective that tasks one of the manipulators to reach a different point on the circle. Since this secondary task only involves one kinematic chain, we can easily find a desired velocity  $\dot{q}_{0,d}$  that would move the robot's end-effector towards the goal using classical methods. If we applied  $\dot{q}_{0,d}$  directly, the end-effector point would not respect the primary objective of



(a) The initial configuration is shown in white and the final one in gray. The green sphere is the initial cooperative sphere, the turquoise one the final, and in red we show its trajectory in between.



(b) Trajectory of the task space bivector command.

**Figure 13.** Two humanoids reaching for a cooperative sphere. The combined system has 34 degrees of freedom, including both humanoids' waist joints.

staying on the circle. We show this scenario in Figure 16. It can clearly be seen that the distance between the initial and current similarity transformations is non-zero during the execution of the task.

Instead, we project the desired velocity by using the nullspace projector  $N(q)$  from Equation (58)

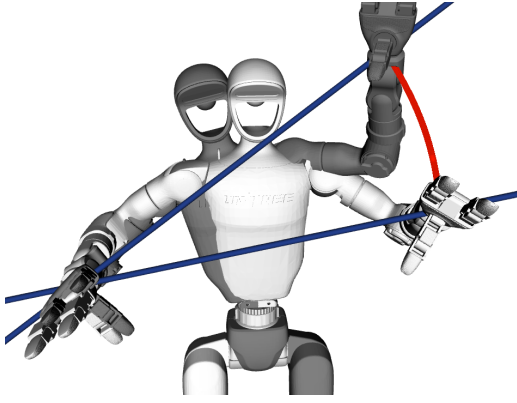
$$\dot{q}_{d,p} = N(q) \begin{bmatrix} \dot{q}_{0,d} \\ 0 \\ 0 \end{bmatrix}. \quad (60)$$

Using  $\dot{q}_{d,p}$ , the first robot still reaches the target point, while staying on the circle. We show this in Figure 17. It can be seen that, by executing the motion in the nullspace of the cooperative task space, which corresponds to the cooperative geometric primitive, i.e. the circle, the first robot follows the circle. Therefore, the similarity distance is zero during the task.

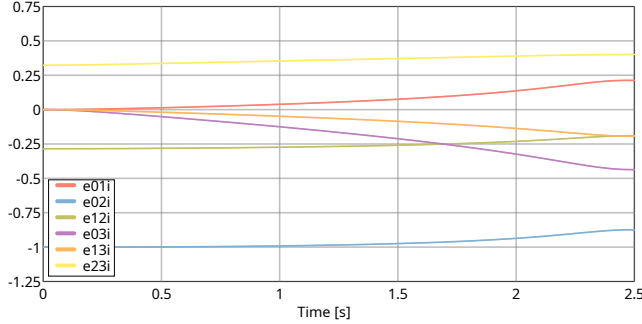
### 4.4 Geometric Singularity

In addition to the classical kinematic singularities that affect individual manipulators, cooperative systems exhibit geometric singularities that arise from the degeneration of the cooperative geometric primitive itself. These singularities occur when the geometric primitive loses its defining characteristics, such as when three points forming a circle become collinear, or when a circle's radius approaches zero or infinity.

To illustrate this phenomenon, we consider the same setup as in Section 4.2.2 (i.e., three Franka robots with a total of



(a) The initial configuration of the robot is shown in white and the final one in gray. The right hand moves (shown with red trajectory) to reach a target point while the left hand remains stationary. The blue line represents the cooperative line primitive connecting the two wrists.



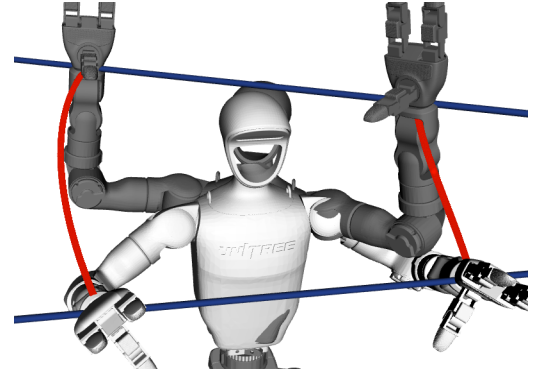
(b) Trajectory of the line parameters over time. The parameters change as the right hand moves while the left hand remains fixed, showing unconstrained evolution of the cooperative line primitive.

**Figure 14.** Asymmetric reaching task with a cooperative line primitive. The right hand reaches a target point while the left hand stays in place, allowing the cooperative line to change freely.

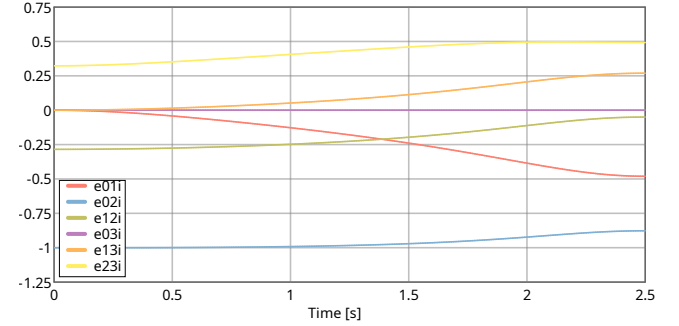
21 degrees of freedom), where the cooperative geometric primitive is a circle defined by the three end-effector points. We task the system with reaching a target configuration that is a geometric singularity, as shown in Figure 18. In this configuration, the three end-effector points approach a collinear arrangement, causing the circle to degenerate. Note that we are not controlling the cooperative circle here, but instead track its evolution while one end-effector reaches the line formed by the other two.

The trajectory of the circle parameters over time is shown in Figure 19a. As the system approaches the singular configuration, several of the circle parameters drop rapidly towards zero, indicating the degeneration of the circle primitive. Correspondingly, the bivector trajectory in Figure 19b shows a drastic change in the cooperative similarity transformation. These large components reflect that lines in CGA can essentially be seen as circles with infinite radius (i.e. the similarity transformation becomes infinite).

The effect of the geometric singularity on the system's manipulability is depicted in Figure 19c. As expected, several components of the task space manipulability decrease significantly when approaching the singularity, with some dropping to zero values. This loss of manipulability indicates that certain directions in the cooperative task space become



(a) The initial configuration of the robot is shown in white and the final one in gray. Both hands move (shown with red trajectories) in a coordinated manner to maintain the horizontal orientation of the cooperative line primitive while the right hand reaches the target.



(b) Trajectory of the line parameters over time. Compared to the unconstrained case, the value of  $e_{03i}$  remains constant, reflecting the horizontal orientation constraint on the cooperative line primitive.

**Figure 15.** Constrained reaching task with a cooperative line primitive. The right hand reaches a target point while the cooperative line is constrained to remain horizontal, requiring coordinated motion of both arms.

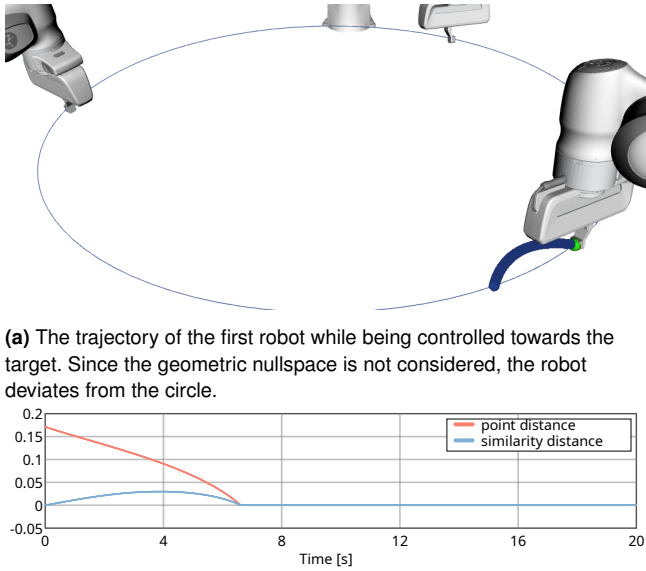
difficult or impossible to control, which is the property of a singular configuration.

This example demonstrates that geometric singularities present an important consideration for the control of cooperative systems modeled by geometric primitives. Unlike classical kinematic singularities that depend solely on individual joint configurations, geometric singularities are intrinsic to the cooperative task space representation and can occur even when all individual manipulators are far from their kinematic singularities. Strategies for singularity avoidance in this context would need to monitor the geometric properties of the cooperative primitive and potentially modify the desired trajectory or task space representation when approaching degenerate configurations.

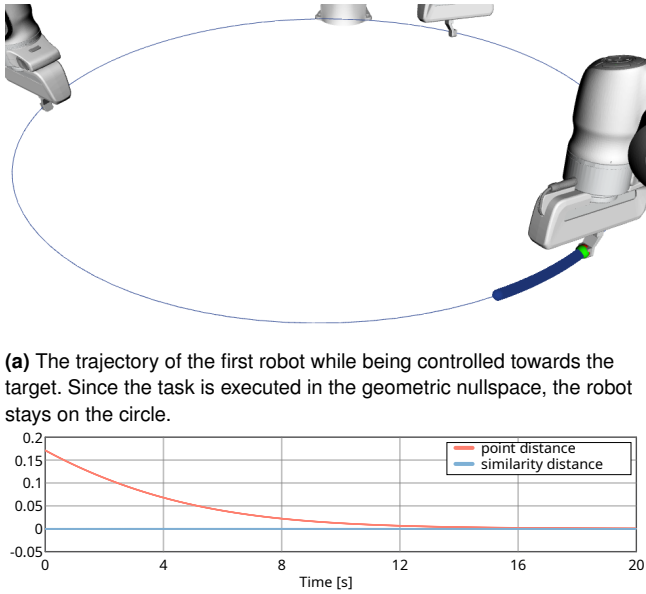
#### 4.5 Teleoperation Demonstration

In this section, we present teleoperation for controlling complex robotic systems with a high number of degrees of freedom. The purpose of the example is to demonstrate the simplicity of the cooperative geometric primitives, and how using the cooperative geometric primitives for modeling the cooperative task space of multiple parallel kinematic chains constrains the degrees of freedom and thus greatly facilitates the control of these complex systems. While we simulate the robotic systems, the input device is real hardware.



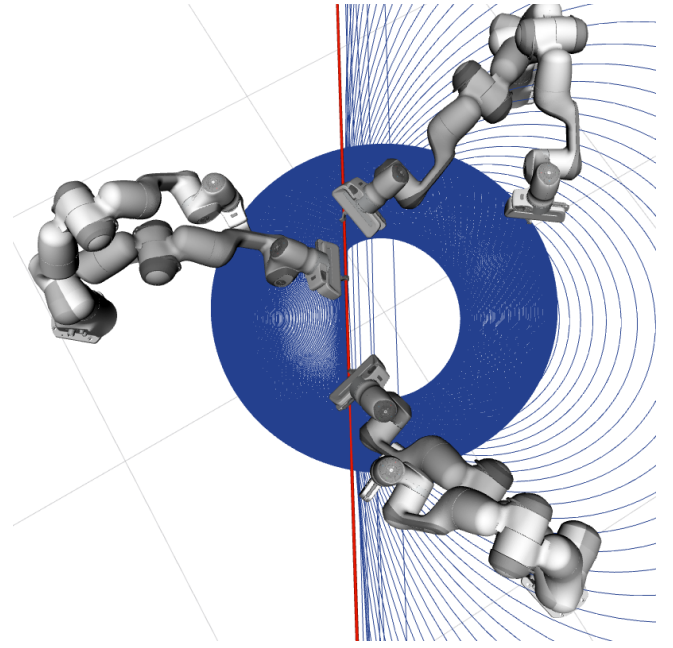


**Figure 16.** Cooperative task space of three Franka robots. The first robot is controlled towards the target point (green). The control is not considering the geometric nullspace, i.e. the cooperative circle.



**Figure 17.** Cooperative task space of three Franka robots. The first robot is controlled towards the target point (green), while considering the geometric nullspace, i.e. the cooperative circle. The primary objective is keeping the cooperative similarity transformation unchanged, which amounts to staying on the circle.

We use a readily available six degree of freedom device that is commonly used in CAD applications as a single input



**Figure 18.** Configuration of the three Franka robots approaching a geometric singularity. The end-effector points approach a collinear arrangement, causing the cooperative circle primitive (shown in blue) to degenerate to a line (shown in red). The initial robot configurations are shown with lighter color.

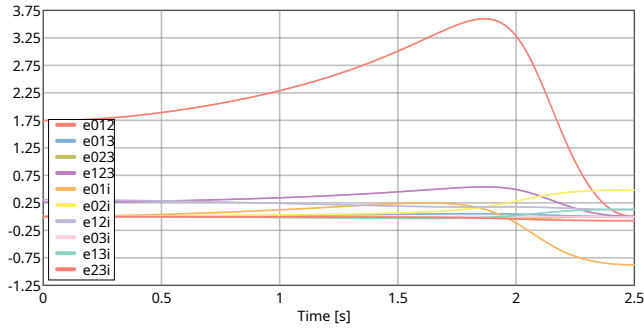
device to obtain the commands for the teleoperation. We then map the six axes of the input device to a Lie algebra element of the similarity transformation group, i.e. a bivector, that then represents a task space command. The mapping of the input device's axes to the similarity transformations and the corresponding motion of the robotic hand is shown in Figure 20. We then use the differential kinematics from Equation (48) to obtain a joint velocity command  $\dot{q}$ . This command is used in a first-order kinematics simulation, i.e. we obtain the new joint position as

$$q_{t+1} = q_t + \Delta t \dot{q}, \quad (61)$$

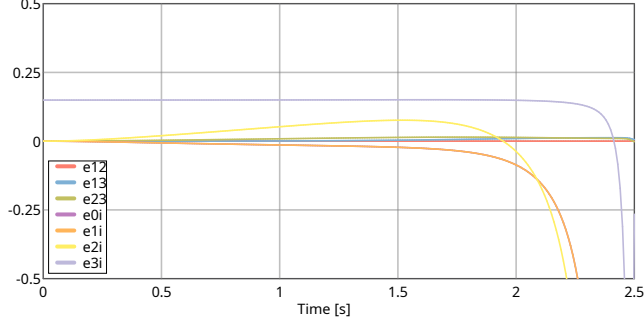
where the time step  $\Delta t$  is chosen to be 0.01s.

We want to point out that the sphere in Figure 20 is only visualizing the current cooperative sphere. It is, however, not used for control, in the sense that the sphere is moved and then tracked by the fingers. Instead, the controller from Equation (48) causes a joint movement that then results in the sphere changing accordingly. Since the cooperative similarity transformation and the related geometric similarity Jacobian capture all involved kinematic chains directly, all fingers are controlled simultaneously. We want to further demonstrate this important point by using the pure dilation bivector command as an example. A pure dilation bivector means that the bivector command is  $B_S = \log(d)e_{0\infty}$ , where  $d \in \mathbb{R}^+$  is the scaling factor. More information on the uniform scaling group can be found in Appendix A.3. It can be inferred that  $d = 1$  leaves the hand position unchanged,  $d \in (0, 1)$  will close the hand, and  $d > 1$  will open it. We can therefore entirely control the opening and closing of the hand using a single scalar. We show the effects of this pure dilation command in Figure 21.

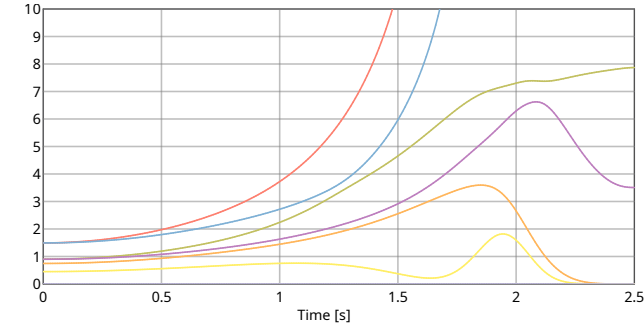
Here, we have shown the example for teleoperating the Leap hand, which has 16 degrees of freedom. The same



(a) Trajectory of the circle primitive parameters over time. The parameters drop rapidly to zero as the circle degenerates when the three points approach collinearity.



(b) Trajectory of the similarity transformation bivector. Several components diverge, reflecting the large transformation for a line, i.e. a circle with infinite radius.



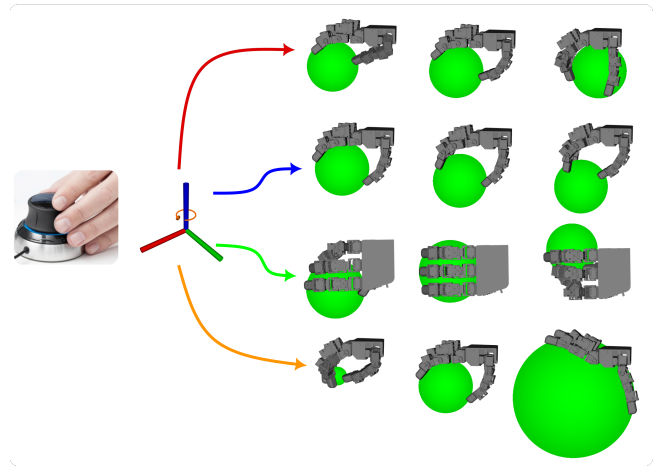
(c) Eigenvalues of the task space manipulability over time. Several components drop to zero values, indicating loss of controllability in certain task space directions due to the geometric singularity.

**Figure 19.** Effect of geometric singularity in a cooperative system of three Franka robots. The system passes through a configuration where the circle primitive degenerates as the three end-effector points become nearly collinear.

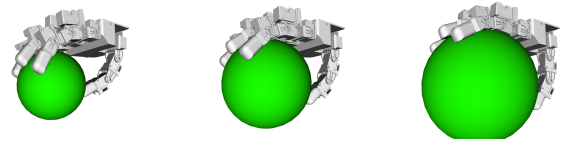
principles also apply to other systems. We show in the accompanying video how we teleoperate a system of three manipulators having 21 degrees, and a system of two humanoids having 34 degrees of freedom. In all cases, we use the simple six-axes device as an input, which shows how the proposed modeling approach greatly simplifies the control of complex robotic systems.

## 5 Discussion

Our unified control framework based on cooperative similarity transformations generalizes the rigid body transformation of a single robot's end-effector to the cooperative control of multiple kinematic chains. The experimental results demonstrated this approach across a variety of robotic systems, ranging from a bi-manual setup to



**Figure 20.** Teleoperation of an anthropomorphic hand. The axes of teleoperation device on the left, are mapped to the different similarity transformations as shown in the center. The three principal axes map to the corresponding translations, while the rotation around the  $z$ -axis is mapped to the dilation. Rotations are not required, since a sphere is invariant to rotations around its center. The resulting motion of the robotic hand is then depicted on the right.



**Figure 21.** Example of the effect of a pure dilation command. We use  $\Delta t = 0.25s$ , with two dilation commands  $B_{S1} = -e_{0\infty}$  and  $B_{S2} = e_{0\infty}$  that correspond to  $d = 0.3679$  and  $d = 2.718$ . The original hand position is shown in the middle. The left image shows the updated hand position after we apply the dilation command  $B_{S1}$  and the right image shows the updated hand position after we apply the dilation command  $B_{S2}$ .

scenarios involving collaborating humanoids. Given that the framework is purely geometric, and considering the variety of systems explored, the kinematic simulations show both the strengths and limitations of our method, which are further discussed in the following subsections.

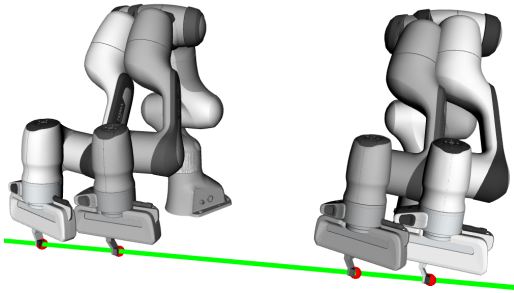
### 5.1 Connection to Traditional Single-Arm Control

Throughout this article, we have mentioned several times that the cooperative similarity transformation and its Jacobians fulfill the same role as the rigid body transformation to the robot end-effector that is used in single-arm systems. Since similarity transformations form a seven-dimensional manifold, as opposed to the six-dimensional one of rigid body transformations, the modeling using the cooperative similarity transformations only introduces one additional dimension to achieve cooperative control behaviors of highly complex robotic systems. Although the involved mathematics that we derived in this article might seem unfamiliar, all the findings in the literature for traditional control methods of single-arm systems remain valid and are directly applicable to the scenarios shown in this article. We have shown this via differential kinematics control in

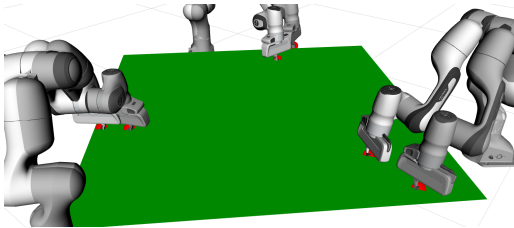
the teleoperation experiments in Section 4.5. This further highlights how the cooperative modeling greatly facilitates the control of complex robotic systems.

## 5.2 Geometric Nullspace

The systems considered in our experiments range from a bimanual setup with 14 degrees of freedom (DoF) (see Figure 22a) to collaborating humanoids with 34 DoF in total. The resulting systems are highly redundant, and thus are inherently challenging to control. Even when restricted to the task space, the dimensionality remains high, i.e. ranging from 12 to 24 DoF. Notably, encoding the cooperative task space using geometric primitives significantly reduces the effective dimensionality of the problem. The similarity transformations associated with these primitives have at most 7 DoF, accounting for translation, rotation, and uniform scaling. Depending on the symmetry of the primitive, the effective number of DoF can be even lower. For instance, a sphere (see Figure 12a) admits only 4 DoF, as rotation does not alter its configuration. As can be inferred from the teleoperation experiment given in Figure 20, this reduction in dimensionality is critical: no teleoperation device exists for 28-DoF systems, nor can a human or an existing algorithm feasibly control such a high-dimensional system directly. In contrast, using our proposed method for controlling the translation and dilation of a target sphere is straightforward.



(a) The line nullspace formed by two Franka robots. Each robot's end-effector is free to move along the line unrestricted. Secondary control objectives tangential to the line will not change the line.



(b) The plane nullspace formed by three Franka robots. Each robot's end-effector is free to move in the plane unrestricted. Secondary control objectives tangential to the plane will not change the plane.

**Figure 22.** Geometric nullspaces that admit the introduction of secondary control objectives. Perturbations that are orthogonal to the geometric primitives will be rejected. The definition of these geometric nullspaces is coordinate-free.

A second key aspect of the redundancy resolution arises from the use of geometric nullspaces. Geometric primitives naturally define subspaces of the underlying algebra, giving rise to geometric nullspace formulations for control. This structure ensures that the controller is inherently stiff when

the system deviates from the primitive, and compliant when moving along it, as we previously showed in the context of optimal control for a single robotic arm L w and Calinon 2023. This principle extends directly to the cooperative geometric primitives introduced in this work, as illustrated in Figures 22a and 22b.

The geometric nullspaces induced by these primitives enable a decoupling of orthogonal control objectives in a coordinate-free manner, while eliminating the need for basis-specific representations or manual tuning of off-diagonal matrix terms. For example, aligning the end-effector with a target line does not constrain motion along the line, enabling the controlled application of contact forces in that direction. We previously demonstrated this feature with a single robot in an impedance control application on curved surfaces Bilaloglu, L w, and Calinon 2025. As shown in Figure 22a, this property naturally extends to cooperative settings, such as coordinated object lifting.

Similarly, in the case of cooperative reaching toward a plane, optimization yields the closest configuration due to the presence of a geometric nullspace, as shown in Figure 22b. Although not illustrated here for brevity, this principle generalizes to other geometric primitives such as circles and spheres. For example, when four robotic arms cooperatively maintain a spherical constraint, they can move compliantly along the tangent directions of the sphere's surface without violating the constraint, while the controller remains stiff in directions orthogonal to the sphere.

## 5.3 Singularities of Cooperative Primitives

Singularities in kinematic chains typically result in the loss of one or more degrees of freedom. These configurations are characterized by a degenerate Jacobian matrix that loses full rank. Similarly, singularities can arise in cooperative geometric primitives when the primitive is no longer uniquely defined. To illustrate this, consider the example of the circle primitive. If all three end-effector points lie on a straight line, the defining circle degenerates into a line. In the conformal geometric model, lines are interpreted as circles with infinite radius. As a result, the cooperative primitive becomes ill-defined, and its behavior cannot be uniquely determined.

These degeneracies produce effects akin to conventional singularities: it becomes ambiguous which end-effector should move, and the associated geometric Jacobian grows unbounded as the three points approach co-linearity. While such configurations are rare and typically do not disrupt control in practice, we included manipulability matrices for the geometric primitives in Section 3.11 to provide tools for analyzing and avoiding these cases. Just as manipulability is used to avoid singularities in single-robot systems, these matrices can be employed alongside standard techniques for singularity-avoiding cooperative control.

**5.3.1 Conditions for Well-Definedness** The cooperative geometric primitive  $X_c(q)$  defined in Equation (14) is well-defined if and only if the outer product of the defining points yields a non-degenerate geometric entity. This requirement imposes geometric constraints on the relative positions of the end-effector points. Specifically:

- **Pointpair/Line:** Two points must be distinct, i.e.,  $P_1 \neq P_2$ . When the points coincide, the pointpair degenerates to a single point.
- **Circle/Plane:** Three points must be non-collinear. When the three points lie on a straight line, the circle degenerates to a line (or equivalently, a circle with infinite radius in CGA).
- **Sphere:** Four points must be non-coplanar. When all four points lie in the same plane, the sphere degenerates to a plane.

As the system approaches these degenerate configurations, the cooperative geometric primitive becomes increasingly ill-defined. In the limiting case, the primitive cannot uniquely determine the cooperative similarity transformation  $V_{Sc}(\mathbf{q})$ , and the control objective loses its geometric meaning.

**5.3.2 Rank Deficiency and Jacobian Conditioning** The cooperative Jacobian matrices derived in Section 3.4 maintain sufficient rank as long as the cooperative primitive remains well-defined and the individual manipulator Jacobians are not singular. However, near degenerate configurations, the cooperative Jacobian exhibits poor conditioning and may lose rank.

Consider the example of a cooperative circle approaching collinearity as shown in Section 4.4. As the three defining points align, the radius of the circle grows unbounded, and certain components of the similarity transformation become ambiguous. Consequently, the geometric similarity Jacobian  $\mathcal{J}_{Sc}^G(\mathbf{q})$  grows unbounded in specific directions, similar to the behavior observed near kinematic singularities in single-arm systems.

The manipulability analysis presented in Section 3.11 provides a quantitative measure for detecting such configurations. The manipulability ellipsoid  $\mathcal{M}_S(\mathbf{q})$  becomes increasingly elongated as degeneracy is approached, with one or more eigenvalues tending toward zero. This behavior can be exploited to implement singularity avoidance strategies, analogous to those used in traditional manipulator control.

**5.3.3 Practical Implications and Mitigation Strategies** In practice, the limitations discussed above can be mitigated through singularity avoidance strategies using the manipulability measure from Section 3.11 to detect and avoid degenerate configurations, similar to standard techniques in single-arm control. While these challenges are inherent to the cooperative primitive framework, they are not fundamentally different from the analogous challenges faced in traditional multi-robot coordination or single-arm control near singularities. The key distinction is that the cooperative geometric primitive provides an explicit, coordinate-free characterization of the degenerate configurations, making them easier to detect and avoid.

## 5.4 Extension to Arbitrary Contact Points on the Robot Body

In this article, we presented cooperative geometric primitives with respect to end-effector points. While this is a reasonable and fairly standard assumption, since the end-effector typically has the highest manipulability and can be equipped with a tool or sensor, there are many scenarios in which it

is desirable to define cooperative geometric primitives with respect to arbitrary points on the robot's body.

For example, when carrying a large and bulky object, humans often use additional contact points such as the torso to help distribute the load and improve stability. We illustrated a preliminary example of this idea in Figure 9a, where the torso of a humanoid robot was used to define the third point of a cooperative circle.

In the general case, however, this requires to define a distance function from the robot surface to the geometric primitives and use its Jacobian to guide the optimization. Accordingly, the optimization can select these points dynamically, depending on the task requirements.

## 5.5 Extension to Human-Robot Collaboration

In Figures 10a and 13a, we presented experiments involving humanoid robots. With a slight change in perspective, these humanoids can be interpreted as abstractions of humans in a human-robot collaboration setting. This observation suggests that cooperative geometric primitives could also be applied to model scenarios involving human-robot collaboration.

The primary distinction between a human and a humanoid robot lies in controllability: the human's motion cannot be directly controlled. As a result, the system must adopt a leader-follower control paradigm, where the human acts as the leader and the robot as the follower. The robot then adapts its end-effector motion in response to the human's hand movements, in order to maintain a desired cooperative primitive, such as a circle or a sphere, defined by the relative positions of the human hand and the robot end-effector(s). Due to the benefits of using the cooperative geometric primitives, the resulting control scheme would remain geometrically consistent.

## 5.6 Extension to $n$ Kinematic Chains

While we presented control strategies for up to four parallel kinematic chains, this practical limitation only stems from our choice of algebra, i.e. conformal geometric algebra  $\mathbb{G}_{4,1}$ . Here, CGA is the smallest algebra that allows for the representation of geometric primitives capturing the collaborative behaviour of up to four parallel kinematic chains. This choice is justified, since three and four-fingered hands are common and able to secure a grasp by removing all DoFs of a target object. Hence, we want to point out that this limitation is only of practical nature, but not of theoretical one. More concretely, using a different choice of the underlying quadratic space  $(\mathbb{R}^{p,q,r}, g(\mathbf{x}, \mathbf{x}))$ , the corresponding geometric algebra  $\mathbb{G}_{p,q,r}$  would allow for the representation of more parallel kinematic chains, such as in the case of five-fingered hands. The presented results based on the derivation of the collaborative geometric primitives would remain valid.

## 5.7 Different Types of Geometric Primitives

In this work, we described the general cooperative similarity task spaces for multiple parallel kinematic chains. We wanted to emphasize the usage of the cooperative similarity transformation as an equivalent to the end-effector pose. Thus, in our control and optimization examples, we always assumed the target geometric primitives to be of the same

type as the task space geometry, as listed in Table 1. In general, however, it is possible to formulate task objectives and constraints involving different types of geometric primitives. In principle, this could be approached via the outer product minimization that we presented for a single manipulator in L w and Calinon 2023, where cooperative geometric primitive could then be used to reach a desired point. Geometric relationships, such as intersections between primitives, can also be fully described in the algebra, and thus can be used in optimization problems. Note that these relationships are well-defined, i.e. they will always yield a mathematically valid result even if the primitives do not intersect.

### 5.8 Numerical Analysis

A critical numerical consideration in the implementation of our framework concerns the conditioning of the logarithmic map used to extract similarity bivectors from versors. The logarithmic maps for rotors, translators, dilators, and similarity transformations presented in Appendix A all involve trigonometric functions in their denominators. The primary source of numerical instability arises when the rotation angle is a whole multiple of  $\pi$ , which manifests as the scalar part of the rotor  $\langle V_R \rangle_0$  approaching  $\pm 1$ . In these configurations, the denominator  $\sin(\cos^{-1}(\langle V_R \rangle_0))$  in Equation (63) becomes zero, leading to division by zero. These singular cases are straightforward to detect by monitoring the scalar component  $|\langle V_R \rangle_0|$  of the rotor before computing the logarithm. Note that rotors are essentially identical to quaternions, and therefore the same challenges and solutions to the conditioning of the logarithmic map apply. Similar considerations apply to the other transformation groups, though the rotation component typically presents the most significant numerical challenge.

### 5.9 Computational Efficiency

A similarity bivector  $B_S \in \mathbb{B}_S$  requires storage of 7 scalar components, while the corresponding similarity transformation versor  $V_S \in \mathcal{S}$  requires 12 components (1 scalar, 7 bivector, and 4 quadvector components). The geometric product of two similarity transformations requires 112 floating-point multiplications and 100 additions, according to the geometric product rules where certain blade combinations are known to result in zero. For a single manipulator, a homogeneous transformation matrix in  $SE(3)$  maps the end effector pose to the target pose through a single matrix product. Exploiting the zeros in the last row of the homogeneous matrices, this multiplication requires about 39 multiplications and 34 additions. Consequently, for three or more independent kinematic chains, the total number of operations required by similarity transforms or by separate rigid body transformations similar in terms of computational complexity. Considering the fact that the similarity transformations encode up to four kinematic chains, this computational overhead is acceptable, since it in turns enables more efficient constraint evaluation on all kinematic chains simultaneously. An in-depth comparison of dual quaternions (i.e. the rigid transformation part of the similarity transformations) versus transformation matrices can be found in Dantam 2021.

### 5.10 Practical Considerations

In this article, the results that we have shown based on our mathematical formulations are all in simulation. Our emphasis is on the theoretical derivations and implications of this novel modeling approach for cooperative systems of multiple parallel kinematic chains. However, we presented all the necessary tools for using our approach in practice. The three different control approaches shown in Section 4 all yield valid joint space commands that can be used on real systems. Of course, the gain parameters, such as the stiffness and damping, would need to be tuned for a real system. Apart from that, a potential issue are non-zero joint velocities despite reaching the target. This is an inherent issue of redundant systems that can be solved using various methods that add dissipative terms in the nullspace while preserving passivity.

## 6 Conclusion

In this work, we introduced a framework for modeling the cooperative task space of multiple parallel kinematic chains through the integration of cooperative geometric primitives and similarity transformations. By deriving the mathematical foundations of this approach, we demonstrated that cooperative geometric primitives offer a powerful abstraction for simplifying the representation and analysis of complex robotic systems, particularly those with a high number of degrees of freedom. This abstraction reduces the inherent modeling complexity by encapsulating intricate kinematic interactions into unified geometric constructs, enabling intuitive and scalable coordination strategies. In this article, we focused on the mathematical derivation of the cooperative task spaces based on the geometric primitives and presented their application in abstract experiments in order to preserve generality. Hence, this work offers various interesting extension opportunities that could be addressed in future work.

Our formulation of the cooperative similarity transformation presents a direct link to classical control methods. Based on the cooperative similarity transformation, we derived an analytic and geometric Jacobian that can be used in standard control techniques. We demonstrated this using optimal control and teleoperation with differential kinematics as examples. Even though the controlled systems have a high number of degrees of freedom, the controller based on the similarity transformation is almost identical to single-arm controllers, and only adds a single dimension for controlling the dilation. Combined with the concept of geometric nullspaces, which decouple orthogonal control objectives in a coordinate-free manner, we have derived the theory of a versatile control framework that leverages the algebraic properties of geometric primitives. Our experiments have shown the general applicability of this control formulation. Actual real-world tasks, however, will require more complex modeling approaches, where different geometric primitives are combined in a single objective function in order to represent the task constraints. We will address this in future work by focusing on the applications, while using the findings of this article as the theoretical foundations. These applications could include the modeling of human-robot collaboration tasks, where the cooperative geometric primitives are



used to implement geometrically consistent leader-follower paradigms that exploit the inherent nullspaces.

## Acknowledgements

## Declaration of conflicting interests

The author(s) declared no potential conflicts of interest with respect to the research, authorship, and/or publication of this article.

## Funding

This work was supported by the State Secretariat for Education, Research and Innovation in Switzerland for participation in the European Commission's Horizon Europe Program through the INTELLIMAN project (<https://intelliman-project.eu/>), HORIZON-CL4-Digital-Emerging Grant 101070136) and the SESTOSENZO project (<http://sestosenzo.eu/>, HORIZON-CL4-Digital-Emerging Grant 101070310).

## References

- Adorno, Bruno Vilhena, Philippe Fraise, and Sébastien Druon (Oct. 2010). "Dual Position Control Strategies Using the Cooperative Dual Task-Space Framework". In: *IEEE International Conference on Intelligent Robots and Systems (IROS)*, pp. 3955–3960. DOI: 10.1109/IROS.2010.5650218.
- Aladele, Victor, Carlos R. De Cos, Dimos V. Dimarogonas, and Seth Hutchinson (Dec. 2022). "An Adaptive Cooperative Manipulation Control Framework for Multi-Agent Disturbance Rejection". In: *IEEE Conference on Decision and Control (CDC)*, pp. 100–106. DOI: 10.1109/CDC51059.2022.9992478.
- Aljalbout, Elie and Maximilian Karl (June 2023). "CLAS: Coordinating Multi-Robot Manipulation with Central Latent Action Spaces". In: *Proceedings of The 5th Annual Learning for Dynamics and Control Conference*. Vol. 211. PMLR, pp. 1152–1166.
- Bilaloglu, Cem, Tobias Löw, and Sylvain Calinon (2025). "Tactile Ergodic Coverage on Curved Surfaces". In: *IEEE Transactions on Robotics* 41, pp. 1421–1435. DOI: 10.1109/TRO.2025.3532513.
- Bircher, Walter G., Andrew S. Morgan, and Aaron M. Dollar (May 2021). "Complex Manipulation with a Simple Robotic Hand through Contact Breaking and Caging". In: *Science Robotics* 6.54, eabd2666. DOI: 10.1126/scirobotics.abd2666.
- Carey, Nicole E. and Justin Werfel (2024). "A Force-Mediated Controller for Cooperative Object Manipulation with Independent Autonomous Robots". In: *Distributed Autonomous Robotic Systems*. Ed. by Julien Bourgeois et al. Vol. 28. Cham: Springer Nature Switzerland, pp. 140–155. DOI: 10.1007/978-3-031-51497-5\_11.
- Dantam, Neil T (Sept. 2021). "Robust and Efficient Forward, Differential, and Inverse Kinematics Using Dual Quaternions". In: *The International Journal of Robotics Research* 40.10-11, pp. 1087–1105. DOI: 10.1177/0278364920931948.
- De Pascali, Luca, Sebastian Erhart, Luca Zaccarian, Biral Francesco, and Sandra Hirche (Feb. 2022). "A Decoupling Scheme for Force Control in Cooperative Multi-Robot Manipulation Tasks". In: *2022 IEEE 17th International Conference on Advanced Motion Control (AMC)*, pp. 243–249. DOI: 10.1109/AMC51637.2022.9729263.
- Erhart, Sebastian, Dominik Sieber, and Sandra Hirche (Nov. 2013). "An Impedance-Based Control Architecture for Multi-Robot Cooperative Dual-Arm Mobile Manipulation". In: *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 315–322. DOI: 10.1109/IROS.2013.6696370.
- Farivarnejad, Hamed and Spring Berman (May 2022). "Multirobot Control Strategies for Collective Transport". In: *Annual Review of Control, Robotics, and Autonomous Systems* 5.1, pp. 205–219. DOI: 10.1146/annurev-control-042920-095844.
- Ferrentino, Enrico, Heitor J. Savino, Antonio Franchi, and Pasquale Chiacchio (Oct. 2024). "A Dynamic Programming Framework for Optimal Planning of Redundant Robots Along Prescribed Paths With Kineto-Dynamic Constraints". In: *IEEE Transactions on Automation Science and Engineering* 21.4, pp. 6744–6757. DOI: 10.1109/TASE.2023.3330371.
- Franceschi, Paolo, Andrea Bussolan, Vincenzo Pomponi, Oliver Avram, Stefano Baraldo, and Anna Valente (June 2025). *Human-Robot Collaborative Transport Personalization via Dynamic Movement Primitives and Velocity Scaling*. DOI: 10.48550/arXiv.2506.09697.
- Gao, Jianfeng, Xiaoshu Jin, Franziska Krebs, Noémie Jaquier, and Tamim Asfour (May 2024). "Bi-KVIL: Keypoints-based Visual Imitation Learning of Bimanual Manipulation Tasks". In: *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 16850–16857. DOI: 10.1109/ICRA57147.2024.10610763.
- Gao, Jianfeng, Zhi Tao, Noémie Jaquier, and Tamim Asfour (Oct. 2023). "K-VIL: Keypoints-Based Visual Imitation Learning". In: *IEEE Transactions on Robotics* 39.5, pp. 3888–3908. DOI: 10.1109/TRO.2023.3286074.
- Gao, Wei and Russ Tedrake (Apr. 2021). "kPAM 2.0: Feedback Control for Category-Level Robotic Manipulation". In: *IEEE Robotics and Automation Letters* 6.2, pp. 2962–2969. DOI: 10.1109/LRA.2021.3062315.
- Haug, Edward J. (Dec. 2023). "Redundant Serial Manipulator Inverse Position Kinematics and Dynamics". In: *Journal of Mechanisms and Robotics* 16.081008. DOI: 10.1115/1.4064047.
- He, Yanhao, Min Wu, and Steven Liu (Jan. 2020). "An Optimisation-Based Distributed Cooperative Control for Multi-Robot Manipulation with Obstacle Avoidance". In: *IFAC-PapersOnLine*. 21st IFAC World Congress 53.2, pp. 9859–9864. DOI: 10.1016/j.ifacol.2020.12.2691.
- (May 2022). "A Distributed Optimal Control Framework for Multi-Robot Cooperative Manipulation in Dynamic Environments". In: *Journal of Intelligent & Robotic Systems* 105.1, p. 8. DOI: 10.1007/s10846-022-01621-4.
- Huang, Wenlong, Chen Wang, Yunzhu Li, Ruohan Zhang, and Li Fei-Fei (Jan. 2025). "ReKep: Spatio-Temporal Reasoning of Relational Keypoint Constraints for Robotic Manipulation". In: *Proceedings of The 8th Conference on Robot Learning*. PMLR, pp. 4573–4602.
- Kadalagere Sampath, Suhas, Ning Wang, Hao Wu, and Chenguang Yang (2023). "Review on Human-like Robot Manipulation Using Dexterous Hands". In: *Cognitive Computation and Systems* 5.1, pp. 14–29. DOI: 10.1049/ccs2.12073.
- Khatib, O. (Feb. 1987). "A Unified Approach for Motion and Force Control of Robot Manipulators: The Operational Space Formulation". In: *IEEE Journal on Robotics and Automation* 3.1, pp. 43–53. DOI: 10.1109/JRA.1987.1087068.
- Lasenby, J., H. Hadfield, and A. Lasenby (Oct. 2019). "Calculating the Rotor between Conformal Objects". In: *Advances in Applied Clifford Algebras* 29.5, p. 102. DOI: 10.1007/s00006-019-1014-8.
- Löw, Tobias and Sylvain Calinon (Oct. 2023). "Geometric Algebra for Optimal Control With Applications in Manipulation Tasks". In: *IEEE Transactions on Robotics* 39.5, pp. 3586–3600. DOI: 10.1109/TRO.2023.3277282.
- (May 2024). "Extending the Cooperative Dual-Task Space in Conformal Geometric Algebra". In: *2024 IEEE International Conference on Robotics and Automation (ICRA)*. Yokohama, Japan: IEEE, pp. 14882–14887. DOI: 10.1109/ICRA57147.2024.10610558.
- Manuelli, Lucas, Wei Gao, Peter Florence, and Russ Tedrake (2022). "KPAM: KeyPoint Affordances for Category-Level Robotic Manipulation". In: *Robotics Research*. Ed. by Tamim Asfour, Eiichi Yoshida, Jaeheung Park, Henrik Christensen, and Oussama Khatib. Cham: Springer International Publishing, pp. 132–157. DOI: 10.1007/978-3-030-95459-8\_9.
- Patel, Vatsal V. and Aaron M. Dollar (Sept. 2021). "Robot Hand Based on a Spherical Parallel Mechanism for Within-Hand Rotations about a Fixed Point". In: *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. Prague, Czech Republic: IEEE, pp. 709–716. DOI: 10.1109/IROS51168.2021.9636704.
- Perwass, Christian (2009). *Geometric Algebra with Applications in Engineering*. Geometry and Computing 4. Berlin: Springer.



- Pfanne, Martin, Maxime Chalon, Freek Stulp, Helge Ritter, and Alin Albu-Sch ffer (Apr. 2020). “Object-Level Impedance Control for Dexterous In-Hand Manipulation”. In: *IEEE Robotics and Automation Letters* 5.2, pp. 2987–2994. DOI: 10.1109/LRA.2020.2974702.
- Saveriano, Matteo, Fares J Abu-Dakka, Alja  Kramberger, and Luka Peternel (Nov. 2023). “Dynamic Movement Primitives in Robotics: A Tutorial Survey”. In: *The International Journal of Robotics Research* 42.13, pp. 1133–1184. DOI: 10.1177/02783649231201196.
- Scheikl, Paul Maria, Nicolas Schreiber, Christoph Haas, Niklas Freymuth, Gerhard Neumann, Rudolf Lioutikov, and Franziska Mathis-Ullrich (June 2024). “Movement Primitive Diffusion: Learning Gentle Robotic Manipulation of Deformable Objects”. In: *IEEE Robotics and Automation Letters* 9.6, pp. 5338–5345. DOI: 10.1109/LRA.2024.3382529.
- Sidiropoulos, Antonis, Yiannis Karayiannidis, and Zoe Doulgeri (June 2019). “Human-Robot Collaborative Object Transfer Using Human Motion Prediction Based on Dynamic Movement Primitives”. In: *2019 18th European Control Conference (ECC)*, pp. 2583–2588. DOI: 10.23919/ECC.2019.8796249.
- Sieber, Dominik, Frederik Deroo, and Sandra Hirche (Nov. 2013). “Formation-Based Approach for Multi-Robot Cooperative Manipulation Based on Optimal Control Design”. In: *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*. Tokyo: IEEE, pp. 5227–5233. DOI: 10.1109/IROS.2013.6697112.
- Sieber, Dominik and Sandra Hirche (July 2019). “Human-Guided Multirobot Cooperative Manipulation”. In: *IEEE Transactions on Control Systems Technology* 27.4, pp. 1492–1509. DOI: 10.1109/TCST.2018.2813323.
- Silverio, Joao, Leonel Roza, Sylvain Calinon, and Darwin G. Caldwell (Sept. 2015). “Learning Bimanual End-Effector Poses from Demonstrations Using Task-Parameterized Dynamical Systems”. In: *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. Hamburg: IEEE, pp. 464–470. DOI: 10.1109/IROS.2015.7353413.
- Spletzer, J., A.K. Das, R. Fierro, C.J. Taylor, V. Kumar, and J.P. Ostrowski (Oct. 2001). “Cooperative Localization and Control for Multi-Robot Manipulation”. In: *Proceedings 2001 IEEE/RSJ International Conference on Intelligent Robots and Systems. Expanding the Societal Role of Robotics in the Next Millennium (Cat. No.01CH37180)*. Vol. 2, 631–636 vol.2. DOI: 10.1109/IROS.2001.976240.
- Yao, Kunpeng and Aude Billard (2023). “Exploiting Kinematic Redundancy for Robotic Grasping of Multiple Objects”. In: 39.3, pp. 1982–2002. DOI: 10.1109/TRO.2023.3253249.

## A Transformation Groups in CGA

The geometric product of  $n$  vectors  $x_i \in \mathbb{G}_{4,1}$ , i.e.  $V = \prod_{i=1}^n x_i$ , forms the set of versors that combined with the geometric product form the Clifford group  $\Gamma(4,1)$ . The restrictions to  $VV^{-1} = \pm 1$  or  $VV^{-1} = 1$ , yield the groups  $Pin(4,1)$  and  $Spin(4,1)$ , respectively. Note that these groups are double covers of the matrix Lie groups  $O(4,1)$  and  $SO(4,1)$ , i.e. the orthogonal and special orthogonal groups. In general, this means that CGA allows for the representation of conformal transformations in  $\mathbb{R}^3$  as a product of vectors instead of highly non-linear matrix expressions, since  $O(4,1)$  is isomorphic to the group of conformal transformations  $Conf(3)$ .

In the following, we list the different transformations groups of CGA. We focus on the subgroups of  $Spin^+(4,1)$ , i.e. we include only versors that are comprised of an even number of reflections in hyperplanes. This means that we omit pure reflections, since we are only interested in transformations that preserve the handedness.

### A.1 Rotation Group

The group of rotations in three-dimensional Euclidean space is usually represented by the special orthogonal group

$SO(3)$ , i.e. a matrix Lie group. The group  $Spin(3)$  is its double-cover and can be represented as unit quaternions. In CGA, it is the rotors that form an isomorphic group to unit quaternions and we denote them here as  $\mathcal{R}$ . Their Lie algebra is the bivector algebra  $\mathbb{B}_R = \text{span}\{e_{12}, e_{13}, e_{23}\}$ . Given the elements  $V_R \in \mathcal{R}$  and  $B_R \in \mathbb{B}_R$ , the exponential map  $\exp_{\mathcal{R}} : \mathbb{B}_R \rightarrow \mathcal{R}$  and its inverse the logarithmic map  $\log_{\mathcal{R}} : \mathcal{R} \rightarrow \mathbb{B}_R$  are

$$\begin{aligned} V_R &= \exp(B_R) \\ &= \cos\left(\frac{1}{2}\|B_R\|\right) \\ &\quad - \sin\left(\frac{1}{2}\|B_R\|\right)\|B_R\|^{-1}B_R, \end{aligned} \quad (62)$$

and

$$B_R = \log(V_R) = \frac{-2 \cos^{-1}(\langle V_R \rangle_0)}{\sin(\cos^{-1}(\langle V_R \rangle_0))} \langle V_R \rangle_2. \quad (63)$$

### A.2 Translation Group

The translation group of  $\mathbb{R}^3$  is the Euclidean space itself under the addition operation, i.e.  $(\mathbb{R}^3, +)$ , which is often shortened to simply  $\mathbb{R}^3$ . In CGA, this group can be represented in versor form. Here, we denote the translation group containing all translator versor in CGA as  $\mathcal{T}$ . Note that, unlike the rotation group in CGA, the group  $\mathcal{T}$  is not a double-cover of  $(\mathbb{R}^3, +)$ , since  $(\mathbb{R}^3, +)$  is already a simply-connected group. The Lie algebra of the group  $\mathcal{T}$  is the bivector algebra  $\mathbb{B}_T = \text{span}\{e_{1\infty}, e_{2\infty}, e_{3\infty}\}$ . Given the elements  $V_T \in \mathcal{T}$  and  $B_T \in \mathbb{B}_T$ , the exponential map  $\exp_{\mathcal{T}} : \mathbb{B}_T \rightarrow \mathcal{T}$  and its inverse the logarithmic map  $\log_{\mathcal{T}} : \mathcal{T} \rightarrow \mathbb{B}_T$  are

$$V_T = \exp(B_T) = 1 - \frac{1}{2}B_T, \quad (64)$$

and

$$B_T = \log(V_T) = -2\langle V_T \rangle_2. \quad (65)$$

In general, the translation bivector  $B_T$  can be found from a Euclidean vector  $t \in \mathbb{R}^3$  as

$$B_T = t \wedge e_{\infty}. \quad (66)$$

### A.3 Uniform Scaling Group

Uniform scaling is a transformation that preserves geometric similarity, i.e. the shape, proportions, angles and orientation as well as parallelism and collinearity are preserved while distances are changed by an isotropic scaling factor. Here, we restrict uniform scaling to positive scalars  $\mathbb{R}^+$  to preserve the handedness as well. In CGA, the versor achieving this is called a dilator  $V_D$  and consequently the set of all dilators is the dilation group  $\mathcal{D}$ , with its corresponding bivector Lie algebra  $\mathbb{B}_D = \text{span}\{e_{0\infty}\}$ . Given the elements  $V_D \in \mathcal{D}$  and  $B_D \in \mathbb{B}_D$ , the exponential map  $\exp_{\mathcal{D}} : \mathbb{B}_D \rightarrow \mathcal{D}$  and its inverse the logarithmic map  $\log_{\mathcal{D}} : \mathcal{D} \rightarrow \mathbb{B}_D$  are

$$\begin{aligned} V_D &= \exp(B_D) \\ &= \cosh\left(\frac{1}{2}\|B_D\|\right) - \sinh\left(\frac{1}{2}\|B_D\|\right)e_{0\infty}, \end{aligned} \quad (67)$$

and

$$B_D = \log(V_D) = 2 \cosh^{-1}(\langle V_D \rangle_0) e_{0\infty}, \quad (68)$$

where the bivector  $B_D$  then relates to the scaling factor  $d \in \mathbb{R}^+$  via

$$B_D = \log(d) e_{0\infty}. \quad (69)$$

Note that, the scaling is always with respect to the origin.

#### A.4 Rigid Transformation Group

The group of rigid transformations in Euclidean space is the most commonly used group in robotics. Traditionally, it is represented by the matrix Lie group  $\mathbf{SE}(3)$  called the special Euclidean group. Alternative representations, such as dual quaternions, are representations of  $\text{Spin}(3) \ltimes \mathbb{R}^3$ , which is the double-cover of  $\mathbf{SE}(3)$ . Here, we denote this group as  $\mathcal{M}$  and usually call its elements motors  $V_M$ . The group is found as  $\mathcal{M} = \mathcal{R} \ltimes \mathcal{T}$ , and we define the canonical decomposition of an element  $V_M \in \mathcal{M}$  as

$$V_M = V_T V_R. \quad (70)$$

The Lie algebra of  $\mathcal{M}$  is the bivector algebra  $\mathbb{B}_M = \text{span}\{e_{12}, e_{13}, e_{23}, e_{1\infty}, e_{2\infty}, e_{3\infty}\}$  and an element  $B_M \in \mathbb{B}_M$  is decomposed as

$$B_M = B_T + B_R. \quad (71)$$

Consequently, given the elements  $V_M \in \mathcal{M}$  and  $B_M \in \mathbb{B}_M$ , the exponential map  $\exp_{\mathcal{M}} : \mathbb{B}_M \rightarrow \mathcal{M}$  and its inverse the logarithmic map  $\log_{\mathcal{M}} : \mathcal{M} \rightarrow \mathbb{B}_M$  are

$$V_M = \exp(B_M) = \exp(B_T) \exp(B_R), \quad (72)$$

and

$$B_M = \log(V_M) = \log(V_T) + \log(V_R). \quad (73)$$

#### A.5 Similarity Group

The Lie group of direct similarity transformations combines translations, rotations and uniform scaling. In terms of the matrix Lie groups it can be found as the semi-direct product of the special Euclidean group and positive scalars, i.e.  $\mathbf{SIM}(3) = \mathbf{SE}(3) \ltimes \mathbb{R}^+$ . Its double-cover representation using the spin groups found in geometric algebras therefore is  $(\text{Spin}(3) \ltimes \mathbb{R}^3) \ltimes \mathbb{R}^+$ . Here, we denote the versor representation of the similarity group in CGA as  $\mathcal{S} = \mathcal{M} \ltimes \mathcal{D} = (\mathcal{R} \ltimes \mathcal{T}) \ltimes \mathcal{D}$  and we define the canonical decomposition of an element  $V_S \in \mathcal{S}$  as

$$V_S = V_T V_R V_D. \quad (74)$$

The Lie algebra of  $\mathcal{S}$  is the bivector algebra  $\mathbb{B}_S = \text{span}\{e_{12}, e_{13}, e_{23}, e_{0\infty}, e_{1\infty}, e_{2\infty}, e_{3\infty}\}$  and an element  $B_S \in \mathbb{B}_S$  is decomposed as

$$B_S = B_T + B_R + B_D. \quad (75)$$

Consequently, given the elements  $V_S \in \mathcal{S}$  and  $B_S \in \mathbb{B}_S$ , the exponential map  $\exp_{\mathcal{S}} : \mathbb{B}_S \rightarrow \mathcal{S}$  and its inverse the logarithmic map  $\log_{\mathcal{S}} : \mathcal{S} \rightarrow \mathbb{B}_S$  are

$$V_S = \exp(B_S) = \exp(B_T) \exp(B_R) \exp(B_D), \quad (76)$$

and

$$B_S = \log(V_S) = \log(V_T) + \log(V_R) + \log(V_D). \quad (77)$$

## B Jacobians

### B.1 Normalizing a Multivector

Given a multivector  $X$ , where  $X \tilde{X} \in \mathbb{R}$ , it can be normalized as

$$\bar{X} = X |X \tilde{X}|^{-\frac{1}{2}}. \quad (78)$$

For a multivector valued function  $X = F(\mathbf{x})$ , we can then find its derivative as

$$\begin{aligned} \frac{\partial}{\partial \mathbf{x}} \bar{F}(\mathbf{x}) &= |X \tilde{X}|^{-\frac{1}{2}} \mathcal{J}_F \\ &\quad - \frac{1}{2} |X \tilde{X}|^{-\frac{3}{2}} \left( X (\mathcal{J}_F \tilde{X} + X \tilde{\mathcal{J}}_F) \right). \end{aligned} \quad (79)$$

### B.2 Inverting a Multivector

Given a multivector  $X$ , where  $X \tilde{X} \in \mathbb{R}$ , its inverse can be found as

$$X^{-1} = \tilde{X} (X \tilde{X})^{-1}. \quad (80)$$

For a multivector valued function  $X = F(\mathbf{x})$ , we can then find its derivative as

$$\begin{aligned} \frac{\partial}{\partial \mathbf{x}} F^{-1}(\mathbf{x}) &= (X \tilde{X})^{-1} \tilde{\mathcal{J}}_F \\ &\quad - 2 (X \tilde{X})^{-2} \tilde{X} \langle X \tilde{\mathcal{J}}_F \rangle_0. \end{aligned} \quad (81)$$

### B.3 Analytic Circle Similarity Jacobian

Here, we derive the analytic similarity Jacobian  $\mathcal{J}_{S,c}^A(\mathbf{q})$ , given the cooperative geometric primitive  $X_c(\mathbf{q})$  and the corresponding Jacobian  $\mathcal{J}_c^A(\mathbf{q})$  from Equation (15). As shown in Equation (29), we can decompose the analytic similarity into the Jacobians for the translator  $\mathcal{J}_T^A(\mathbf{q})$ , rotor  $\mathcal{J}_R^A(\mathbf{q})$ , and dilator  $\mathcal{J}_D^A(\mathbf{q})$ .

The analytic dilator Jacobian can be found as

$$\mathcal{J}_D^A(\mathbf{q}) = -\frac{1}{4} (P_{\infty} \tilde{P}_{\infty})^{-1} \langle \mathcal{J}_{\infty} \tilde{P}_{\infty} + P_{\infty} \tilde{\mathcal{J}}_{\infty} \rangle_0 V_D, \quad (82)$$

where  $P_{\infty}$  is the projection of infinity from Equation (24)

$$P_{\infty} = (X \cdot e_{\infty}) X^{-1}, \quad (83)$$

and  $\mathcal{J}_{\infty}$  its Jacobian

$$\mathcal{J}_{\infty} = (e_{\infty} \cdot \mathcal{J}_c^A) C^{-1} + (e_{\infty} \cdot C) \hat{\mathcal{J}}_c^A. \quad (84)$$

The analytic rotor Jacobian can be found as

$$\begin{aligned} \mathcal{J}_R^A(\mathbf{q}) &= |R \tilde{R}|^{-\frac{1}{2}} \mathcal{J}_R \\ &\quad - \frac{1}{2} |R \tilde{R}|^{-\frac{3}{2}} R (R \tilde{\mathcal{J}}_R + \mathcal{J}_R \tilde{R}), \end{aligned} \quad (85)$$

where  $R$  is the rotor  $V_R(\mathbf{q})$  before normalization according to Equation (26) and  $\mathcal{J}_R$  the corresponding Jacobian

$$\mathcal{J}_R = e_3 \cdot \bar{\mathcal{J}}_N - e_3 \wedge \bar{\mathcal{J}}_N, \quad (86)$$

where  $\bar{\mathcal{J}}_N$  is the normalized Jacobian of the plane normal found in Equation (25)

$$\begin{aligned} \bar{\mathcal{J}}_N &= |N \tilde{N}|^{-\frac{1}{2}} \mathcal{J}_N \\ &\quad - \frac{1}{2} |N \tilde{N}|^{-\frac{3}{2}} N (N \tilde{\mathcal{J}}_N + \mathcal{J}_N \tilde{N}), \end{aligned} \quad (87)$$

where

$$\mathcal{J}_N = \mathcal{J}_E^* - \frac{1}{2}(\mathbf{e}_0 \cdot \mathcal{J}_E^*)\mathbf{e}_\infty, \quad (88)$$

and

$$\mathcal{J}_E = \mathcal{J}_c^A(\mathbf{q}) \wedge \mathbf{e}_\infty. \quad (89)$$

The analytic translator Jacobian can be found as

$$\begin{aligned} \mathcal{J}_T^A = & -\frac{1}{2} \langle P_c \rangle_0^{-1} \mathcal{J}_P \\ & - \langle P_c \rangle_0^{-1} P_c (-\mathbf{e}_\infty \cdot \langle \mathcal{J}_P \rangle_0) \wedge \mathbf{e}_\infty, \end{aligned} \quad (90)$$

where  $P_c$  is the center point and  $\mathcal{J}_P$  its Jacobian

$$\mathcal{J}_P = \mathcal{J}_c^A(\mathbf{q})\mathbf{e}_\infty X_c(\mathbf{q}) + X_c(\mathbf{q})\mathbf{e}_\infty \mathcal{J}_c^A(\mathbf{q}). \quad (91)$$